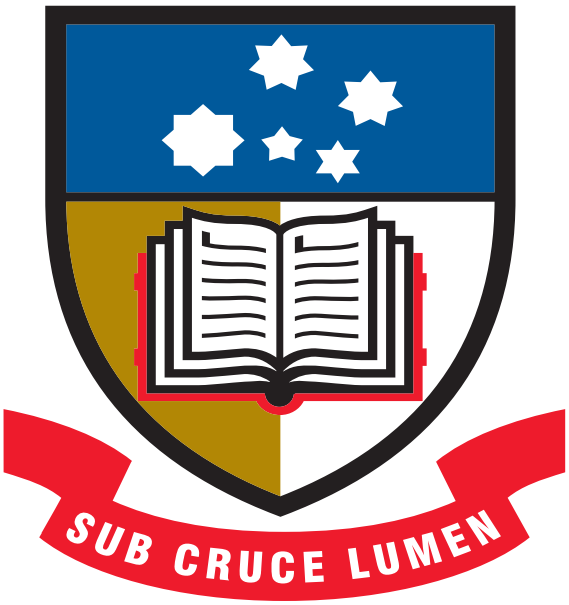
# Application of Memory-Based Approach to Multi-Objective Optimisation on Dynamic Resource-Constrained Project Scheduling with Time-varying Number of Tasks

by

**Manuel Blanco Abello, MEng. Sc. Elec. Eng.**




Submitted to the Graduate Center of the

**University of Adelaide**,

this thesis is a partial fulfillment for

the prerequisites to the degree of

**Doctor of Philosophy**

Supervisor: Prof. Zbigniew Michalewicz

Associate Supervisor: Dr. Lam Thu Bui

$13^{th}$ of June 2014

To the loving memory of my mother,

**Consorcia Blanco Abello**

and of my father,

**Florencio Raz Abello**

whose last words to me were:

"Finish your Ph.D."

. . . Done it, Dad.

# Contents

# List of Reserved Symbols

| | |
|---|---|
| $N$ | Total number of tasks |
| $\mathcal{P}^2$ | Class of bi-objective dynamic RCPS problems, each with increases and without decrease in total number of tasks and has four types of resources |
| $\mathcal{P}^3$ | Class of tri-objective dynamic RCPS problems, each with increases and without decrease in total number of tasks and has four types of resources |
| $\mathcal{C}^2$ | Class of bi-objective dynamic RCPS problems, each with fixed total number of tasks |
| $\mathcal{L}^2$ | Set of problems from $\mathcal{P}^2$ that have labels of one to 30 listed in Table 4.5 |
| $\mathcal{L}^3$ | Set of tri-objective counterparts of $\mathcal{L}^2$ |
| $\mathcal{M}$ | Class of multi-objective dynamic RCPS problems, each with increasing total number of tasks |
| $\mathcal{O}^2$ | Subset of $\mathcal{Q}^2$ |
| $\mathcal{Q}^2$ | Class of bi-objective dynamic RCPS problems, each with increases and without decrease in the total number of tasks and has two types of resources |
| $\mathcal{X}^2$ | a subset of $\mathcal{O}^2$ that is comprised of DOE-designed problems |
| BBD | Box-Behnken Design |
| CBAR | Centroid-Based Adaptation with Random Immigrants |

| | |
|---|---|
| CCD | Central Composite Design |
| EA | Evolutionary Algorithm |
| EDA | Estimation of Distribution Algorithm |
| McBAR | Mapping of Task IDs for CBAR |
| MCS | Monte Carlo Simulation |
| MOE | Military Operation Environment |
| MOO | Multi-Objective Optimisation |
| NSGA-II | Non-Dominated Sorting Genetic Algorithm-II |
| PNT | Precedence Network of Tasks |
| RCPS | Resource-Constrained Project Scheduling |
| RSM | Response Surface Methodology |
| SOSA | Sequential Order of State Alteration |
| TNIS | Task Number Increase Sequence |
| TSC | Type of sequence of changes |

# List of Tables

# List of Figures

# Abstract

Many, if not all, manufacturing processes in industry require scheduling activities; such activities are very important as often they determine the success or failure of some companies. For example, in wine production, grapes are planted, mature fruits are harvested, transported and crushed then the juice obtained is placed in tanks, which are managed during fermentation, and finally the wine is bottle. A schedule can be a solution to a problem which has several, possibly conflicting objectives, e.g. the mininisation of production costs and delays while meeting customer-imposed wine delivery times; the problem also has constraints, e.g. a bottling line cannot be used without being cleaned to process white wine when it last processed red wine. As can be expected, the problem has variables such as the number of wine bottles ordered.

The environment (e.g. wine factory) in which the schedule is implemented may change (e.g. one bottling line breaks down) whereby this schedule becomes infeasible. Consequently, there could be a need to solve a new scheduling problem to obtain a new schedule best suited to the new state of the environment. The number of variables in this new problem may be the same as that of the previous problem. A large proportion of research effort has been directed towards scheduling problems with a constant number of variables despite changes in the environments where the problems are set. However, there are important scheduling problems where the number of variables could vary. For example, in some models of job-shop scheduling problems there are occurrences of additional rush jobs and job cancellations.

This thesis deals with one particular class of scheduling problems, each being multi-objective, resource constrained, and having numbers and values of variables which vary over time. Various traditional operation research methods as well as a few Artificial Intelligence-based techniques, such as Multi-Agent Systems and Evolutionary Algorithms (EA), have been applied to solve this type of problem. In this thesis, a memory-based EA technique was applied to solve problems from the class. Being memory-based, this technique utilises the solutions to problems set in previous states of an environment in

order to solve a problem set in the current state of this environment.

The memory-based EA technique, referred to as *Centroid-Based Adaptation with Random Immigrants* (CBAR), is applicable only to solve multi-objective, resource-constrained problems with a constant number of variables. In this thesis, CBAR is extended to become applicable to solve all problems from the above-mentioned class. The result of this extension is a technique referred to as *Mapping of Task IDs for CBAR* (McBAR).

This thesis investigates the performance, the performance stability over environmental dynamics, and the efficiency of McBAR for solving various problems from the above class, legitimises the sub-algorithms that constitute McBAR and extends McBAR to become proactive (anticipative of future environmental changes).

Compared to the other techniques investigated in this thesis, results showed McBAR to have the best and most stable performance, and to be most efficient for determining solutions to problems from the above class. All of the sub-algorithms of McBAR are shown to be legitimate, while McBAR having been made proactive is shown to be beneficial in some applications.

# Copyright

## Application of Memory-Based Approach to Multi-Objective Optimisation on Dynamic Resource-Constrained Project Scheduling with Time-varying Number of Tasks



---

**Manuel Blanco Abello**
**13th of June 2014**

# Acknowledgements

Although unworthy to state the following, I have to give credence to whom it is due: First and foremost, I would like to thank the triune God, maker of heaven and earth, for all the graces He gave me before and during my study; and to the Blessed Virgin Mary and my dear friend Saint Anthony of Padua[1] for all their intersessions. Sincere gratitude goes to my late parents for their great love and labour for my education.

I never expected to be accepted as a Ph.D. student in Evolutionary Algorithm, considering that my background is in Electrical Engineering. However, Professor Zbigniew Michalewicz supported me to be accepted for Ph.D. student in the Computer Science department. My sincere thanks to him, my supervisor. I also would like to thank Dr. Lam Thu Bui for his great assistance to the publications leading to this theses as my second supervisor. In addition, it is a great honour working with him. Earnest gratitude also goes to Dr. William Winser for his tireless and patient proof-reading of this thesis. Further, thanks to Tracey Young for her kind assistance during my first year of this study. The computing facilities afforded by e-Research South Australia (ERSA) is extremely helpful for the computation of the results presented in this thesis, thanks to this institution as well.

Thanks go to my former lecturers from University of Queensland, Dr. Nicholas Shuley and Dr. David Lovell. Also, thanks go to my former lecturers from the University of San Carlos, Engr. Roger Bajarias, Engr. Tomas Javellana; further, to my lecturers on advanced Mathematics, Dr. Jose T. de Luna, the late Dr. Jose Bernaldez and to my grade school teacher, Sir Labitigan, who awaken my interest on Mathematics. Thanks also to my former classmate, Geneth Remedio, for all her financial assistance.

Several authors who were kind enough to share with me their papers/research results and provided additional explanation when I asked for: André R. Goncalves from School of Electrical and Computer Engineering, University of Campinas, Brazil; Luís Paquete from Faculdade de Economia & CSI Centro de Sistemas Inteligentes, Universidade do

[1] The final edition of this thesis was made exactly on the feast day of Saint Anthony.

Algarve, Portugal; Feili Yu from the University of Connecticut; and Gary W. Oehlert from the University of Minnesota.

# Introduction

## Chapter 1

Intelligently designed schedules are significant in a myriad of fields, such as manufacturing, air travel, and military operations [35,58,231]. With these schedules, manufacturing cost could be reduced, flight schedules could be orderly, and military objectives could be achieved. Tasks in schedules require resources, such as personnel, machines, and raw materials [58, 209]. A scheduling problem where on-going tasks are restricted to utilise a limited number of resources is referred to as a *Resource-Constrained Project Scheduling* (RCPS) problem [58]. Many problems can be transformed into the form of an RCPS, such as production sequencing, timetabling, and flight scheduling.

Many scheduling problems are multi-objective [194, 214, 241]. Suppose in creating an edifice construction schedule one objective is to reduce construction duration that could be accomplished by employing more labourers. However, due to overhead expenses (e.g. hazard fee) per labourer, the construction cost could be higher for the same total man-hours. Now, if the other objective is to reduce the construction cost, which could be accomplished by employing fewer labourers, the construction duration could be longer. Thus, in this example, the two conflicting objectives (minimisation of construction cost and duration) cannot possibly be achieved simultaneously. The problem of simultaneous optimisation of conflicting objectives is referred to as the *Multi-Objective Optimisation* (MOO) problem [106].

There are several measures of the quality of a solution to a given MOO problem [237]. One of them is the weighted sum of the problem objective values. For example, the cost and duration of a schedule being a solution to a MOO problem are multiplied by 0.3 and

0.7, respectively, and then the products are summed to obtain the measure of the quality of this schedule.

The quality of a schedule may be affected by changes in an environment on which this schedule is implemented [40]. For example, in the implementation of an air traffic schedule, bad weather or emergencies occur whereby this schedule might become infeasible to implement or the schedule quality might deteriorate to an unacceptably low level. Thus, it is necessary and/or could be beneficial to revise the schedule [40]. Many real-world RCPS problems are set in dynamic environments which may change over time the constraints, the number of variables, or the parametric values of these problems [37, 104]. For example, a military operation scheduling problem has a constraint not to bomb a civilian area. However, if after the start of implementing the problem solution/schedule, the civilians are known to have left, then this constraint can be removed. It could also transpire during the implementation that hostile combatants retreat to recuperate then attack at some time later, and that such an attack could elicit the occurrence of new military operational tasks not considered in the creation of the solution. If variables of the scheduling problem correspond to tasks then the occurrence of the new tasks can change the problem dimension. Another event in the military operation could be an increase in the number of enemy combatants. If this number is a problem parameter then the increase will change the value of this parameter.

In the context of this thesis, at any moment of change in any dynamic environment, only three types of task status are considered: finished, to-be-executed and on-going, which indicates the task status of being started. Let the *total number of tasks* be the sum of the number of tasks at any of the statuses. Further, let $\mathcal{M}$ denotes a class of RCPS problems set in dynamic environments where each problem has two or more conflicting objectives and each environment has no decrease in the total number of tasks and has at least one moment of increase in the total number of tasks, an increase that instigates change in problem dimension. This introduction summarises the research in this thesis on the creation of effective and efficient technique to determine solutions to many problems

from $\mathcal{M}$.

Section 1.1 explains the motivation of the thesis, i.e. the need to create the effective and efficient technique. Before proceeding, notations are introduced and terms are defined in Section 1.2. Then Section 1.3 declares the hypothesis and enumerates several thesis' goals whose fulfilment proves this hypothesis. The methods to accomplish these goals are then explored in Section 1.4, followed by the discussion of the major contributions of the research in Section 1.5. Section 1.6 concludes outlining the organisation of the thesis.

## 1.1 Motivation

An RCPS problem is NP-hard which is often approached using modern heuristic methods, e.g. *Evolutionary Algorithm* (EA) [123]. Consider a schedule as a solution to an RCPS problem set in the original state of a dynamic environment. One possible approach for addressing RCPS in this dynamic environment is to revise the schedule through an EA-based technique every time this environment changes [205]. Suppose that the schedules as solutions to problems set in the environment at previous states are stored in a memory and then retrieved to help the search for a schedule as the solution to a problem set in the current environmental state. This approach is referred to as a *Memory-Based Approach* (MBA), and was shown in some researches [40, 131, 185] to be indeed effective for helping the search.

In several studies MBA has been applied to solve problems set in dynamic environments [22, 223, 224]. Our previous study/research [40] successfully applied a type of memory-based EA technique, referred to as *Centroid-Based Adaptation with Random Immigrants* (CBAR), to bi-objective RCPS problems set in dynamic environments with a constant total number of tasks. However, our other work [3] demonstrated CBAR to be inapplicable for solving any problem from $\mathcal{M}$, i.e. a multi-objective RCPS problem set in a dynamic environment with increases and without decreases in the total number of tasks.

Important scheduling problems are set in dynamic environments, each with a time-varying total number of tasks. For example, in some job-shop scheduling, additional rush jobs and job cancellations occur [22]. To solve this type of problem, some Artificial Intelligence (AI) techniques, such as Multi-Agent System (MAS) [14, 109] and EA [27, 28, 127, 164], have been applied. However, the application of a memory-based EA technique to solve problems from $\mathcal{M}$ is absent in the literature. Thus, the motivation of this research is the need to create an effective and efficient memory-based EA technique to solve many problems from $\mathcal{M}$. Exploiting the initial success of CBAR and considering its inapplicability, the technique sought is an innovation of CBAR referred to as *Mapping of Task IDs for CBAR* (McBAR).

## 1.2 Notations and Definitions

The following notations and definitions will be used in this thesis.

### 1.2.1 Tasks and Resources

*Resources* are assets that can be utilised for a system to function. Examples of resources are soldiers, fuel and weapons. Let $j$ be a resource type label; $1 \leq j \leq J$, where $J$ is the number of resource types. $R_j$ be the number of items of resource type labelled $j$.

*Resource constraint* is a restriction on the number $r_{j,t}$ of being-used items of a certain resource type labelled $j$ that, at any time $t$, must not exceed a given number $R_j$ of items of the same resource type; note that unavailable items of one resource type cannot be replaced by items of other resource types, e.g. shut down fighter jets cannot be replaced by tanks and soldiers. Let the resource constraint be denoted as *rsrc* and be expressed as,

$$\forall\ 1 \leq j \leq J,\ \forall\ t \qquad\qquad r_{j,t} \leq R_j \tag{1.1}$$

*Task* is an activity to be undertaken. It is characterised by the following attributes: the

nature of activity, expected duration, number and type of resources utilised, precedence relationship to other tasks, starting time, and status as enumerated above. For example, the task of bombing an area is expected to take four hours, requires six fighter jets, must be executed after the failure of some negotiations, and is to start at dawn of the day after the failure. Let $T_n$ signify a task with ID $n$, $1 \leq n \leq N$, where $N$ is the total number of tasks.

*Task precedence relationship* is a priority in the order of tasks. For example, if task $T_2$ is to transport weapons and ammunitions, task $T_5$ is to bomb an area, task $T_{16}$ is to interrogate prisoners of war, and task $T_{17}$ is to clear the bombed area, then a task precedence relationship could be: the bombing of the area must be done after transporting weapons and ammunitions, and before clearing the bombed area and interrogating prisoners of war. Let the *precedence constraint* of any scheduling problem on tasks be denoted as *prec* and expressed as,

$$\forall\ p(i,j) \in P \qquad\qquad E_i \leq S_j \tag{1.2}$$

where $P$ is a set of pairs $p(i,j)$ of task indexes, $1 \leq i,j \leq N$, with task $T_i$ demanded in the problem to precede task $T_j$; $E_i$ is the ending time of task $T_i$, and $S_j$ is the starting time of task $T_j$.

### 1.2.2 General Static Problem

Following [156], let a multi-objective *static* RCPS problem be represented by a triple $\langle\ E \mid C \mid O\ \rangle$ where, $E$ denotes the model of a static environment in which the problem is set, $C$ denotes the model of problem constraints, and $O$ denotes the problem objectives. The elements of the triple are made slightly specific as follows. Let $E$ be a set composed of the total number of tasks $N$ and a set

$$P = \{p_1, p_2, \ldots, p_M\} \tag{1.3}$$

of $M$ entities other than $N$ that characterise the environment, i.e. $E = \{P, N\}$. Notice that in $E$, the total number $N$ of tasks is not made as an element of $P$ so as to emphasise one of the foci of the thesis on the increase of the total number of tasks in environments. The other entities in $P$ could be the cost of moving each resource from one task to another, the duration of each task, the number and types of resources utilised by each task and the landscape characteristics such as roads, topography and vegetation.

Let $C$ be a set composed of a set $Prec$ of task precedencies and a set $Rsrc$ of resource constraints whose elements are of the same form as the inequalities of 1.1 and 1.2 respectively, i.e. $C = \{Prec, Rsrc\}$. A resource constraint in $Rsrc$ could be that the number of in-use fighter jets must not exceed seven and the precedence constraint in $Prec$ could be that troop deployment must be executed after aerial bombing.

Let $O$ be a set $\{o_1, o_2, \ldots, o_K\}$ of problem objectives to be optimised, where $K$ is the number of objectives included in the model of the problem. The objectives could be to minimise fuel expenses, equipment damage, soldier fatalities, and operation execution duration, and to maximise the secured area.

The static problem is then expressed as,

$$g = \langle \{P, N\} \mid \{Prec, Rsrc\} \mid \{o_1, o_2, \ldots, o_K\} \rangle \tag{1.4}$$

Note that the terms on the right-hand side of this expression are not fully specified. Thus, $g$ is a general problem set in a static environment.

### 1.2.3 Dynamic Environment

Now, consider a *dynamic* environment where snapshots are taken at its original state and immediately after each time it changes. For example, a snapshot is taken at an original environment topography that has a valley occupied by rebels. Let a scheduling problem be formulated from this scenario. Now, suppose the next snapshot is taken immediately after the environment changes for the first time to a state that, in addition to the last state

of this environment, has a road-blocking landslide which hampers transport of logistics from a depot, can thus change the scheduling problem. Note that a snapshot of a dynamic environment is considered as a static environment. Thus, any problem set in it is referred to as a static sub-problem.

Let the order $i$ of taking a snapshot also be the *sequential order of state alteration* (SOSA) of the dynamic environment from where this snapshot is taken. For instance, a third snapshot is taken immediately after the third moment that the environment is altered, i.e. the third SOSA. Given a dynamic environment, let each zeroth snapshot and zeroth SOSA corresponds to the original state of this environment.

Let $\Gamma_i$ be a class of multi-objective static RCPS sub-problems, each set in the $i^{th}$ snapshot of a dynamic environment without decreases and with increases in the total number of tasks. Further, let each of its members be expressed as,

$$g_i = \langle\ \{P_i, N_i\}\ \mid\ \{Prec_i, Rsrc_i\}\ \mid\ \{o_1, o_2, \ldots, o_K\}\ \rangle \tag{1.5}$$

where each term in the right-hand side with subscript $i$ has the same meaning as its non-subscripted counterpart in Equation 1.4; however, this term corresponds to the $i^{th}$ snapshot.

### 1.2.4 Dynamic Problems

As stated above, the members of $\mathcal{M}$ are multi-objective dynamic RCPS problems, each set in a dynamic environment with increases and without decreases in the total number of tasks. So, let a dynamic problem $g \in \mathcal{M}$ be expressed as a sequence of multi-objective static RCPS sub-problems,

$$g = \langle g_0, g_1, \ldots, g_L \rangle \tag{1.6}$$

where $\langle \ldots \rangle$ denotes an ordered set; each sub-problem $g_l \in g$ is a member of $\Gamma_l$ and is to be found in a corresponding in-sequenced snapshot of the environment where the

problem $g$ is set; where the subscripts denote the SOSA of the environment; and $L$ is the number of moments of changes in the environment. Note that a dynamic environment can have several changes at a single moment. Let $g_l$ be referred to as a (static) sub-problem of (dynamic) problem $g$. The perception of a dynamic problem as sequence of static sub-problems is also evident in [212].

### 1.2.5 Subsets of $\mathcal{M}$

Let us now define important subclasses of $\mathcal{M}$.

- $\mathcal{P}^2$: Each member of this subclass is a sequence of sub-problems which individually has two objectives – the minimisation of schedule cost $o_1$ and duration $o_2$, i.e. $O$ in Equation 1.5 is $\{o_1, o_2\}$ for this subclass. In addition, each of these sub-problems has constraints on four types of resources which we will define in Section 4.2.

- $\mathcal{P}^3$: Each member of this subclass is a sequence of sub-problems which individually has three objectives – the last two named objectives and the objective to minimise the probability $o_3$ that a schedule becomes infeasible for violating resource constraints triggered by task duration changes, i.e. $O$ in Equation 1.5 is $\{o_1, o_2, o_3\}$ for this subclass. It is important to determine the probability, especially when the investment required for the schedule is immense [41]. The resource constraints of each of the sub-problems are the same as those of $\mathcal{P}^2$.

- $\mathcal{Q}^2$: Each member of this subclass is a sequence of sub-problems which individually has two objectives – $o_1$ and $o_2$. And, each of these sub-problems has constraints on two types of resources which we will define in Section 4.3.

Although not a sub-class of $\mathcal{M}$, this thesis considers the class $\mathcal{C}^2$ of bi-objective RCPS dynamic problems with a constant total number of tasks. The problems being associated with McBAR (described in Section 1.1) are from the following sub-classes: $\mathcal{L}^2 \subset \mathcal{P}^2$,

$\mathcal{L}^3 \subset \mathcal{P}^3$ and $\mathcal{O}^2 \subset \mathcal{Q}^2$. We will define the subclasses $\mathcal{L}^2$ and $\mathcal{L}^3$ in Section 4.2 and the subclass $\mathcal{O}^2$ in Section 4.3.

A sample notation for a dynamic problem $l^2$ from $\mathcal{L}^2$ is,

$$l^2 = \langle l_0^2, l_1^2, \ldots, l_I^2 \rangle \tag{1.7}$$

where the static sub-problem $l_i^2 \in l^2$ is a member of $\Gamma_i$; and $I$ is a given integer.

### 1.2.6 Set of Important Techniques

The problems from $\mathcal{L}^3$ are solved by an extension of McBAR; while the problems from $\mathcal{O}^2$ and $\mathcal{L}^2$ are associated with the techniques from the set,

$$\begin{aligned} \mathcal{T} \quad = \quad & \{\text{RI}, \text{NDLPOP}, \text{EDA}/\mathcal{P}^2, \text{GIBAR}, \text{CBAM}, \\ & \text{McBA}, \text{McBAR}, \text{McBAS}, \text{MedianBAR}\} \end{aligned} \tag{1.8}$$

where RI is an EA but not memory-based; NDLPOP is an EA and memory-based; EDA/$\mathcal{P}^2$ is not an EA but memory-based; GIBAR and CBAM are EAs and memory-based but do not have the core algorithm of McBAR; and the techniques at the lower row of the equation are EAs and memory-based and have the core algorithm of McBAR. Full definitions of the techniques will be given in Chapter 5, including the core algorithm of McBAR explored in Section 5.6.7.

## 1.3 Goals

The general hypothesis of the thesis is as follows:

"*McBAR is an effective and efficient technique to solve many problems from* $\mathcal{M}$."

Considering the infinite size of $\mathcal{M}$, the scope of the hypothesis does not encompass all members of $\mathcal{M}$. This hypothesis is proven by the achievement of the following goals:

1. The first goal of the thesis is to demonstrate the effectiveness of McBAR for solving *bi-objective* RCPS problems from $\mathcal{L}^2 \cup \mathcal{O}^2 \subset \mathcal{M}$ set in a dynamic environment with increases and without decrease in the total number of tasks. Associated with this goal is the desire to exhibit the dynamics of the effectiveness under the changing environment.

2. The second goal of the thesis is to show that the core algorithm of McBAR is still effective despite being applied to solve the *tri-objective* problems from $\mathcal{L}^3 \subset \mathcal{M}$ ($\mathcal{L}^3$ is defined in Section 1.2.5). Such a demonstration takes into account the dynamics of the environment in which the problems are set.

3. The third goal of the thesis is to determine the limitation on the effectiveness of McBAR for solving problems from $\mathcal{O}^2 \subset \mathcal{M}$ ($\mathcal{O}^2$ is defined in Section 1.2.5). It is of great interest to determine the limitations of any technique, since through these limitations the users of the technique can decide when to start and stop applying this technique. Thus, although the central focus of the thesis is to prove the effectiveness of McBAR, we also include in the thesis the limitation of this effectiveness.

4. The fourth goal of the thesis is to demonstrate the relevance of sub-algorithms for being the components of McBAR. Note that some components of a technique may not be relevant, i.e. their absence might still leave the technique effective. If all of the components are relevant then no computation is wasted by the technique for solving a given problem. Note that the EA-related parameters of McBAR are determined to greatly enhance the performance of McBAR. Thus, given that a system is defined as efficient if it achieves maximum performance while wasting resources the least, then this thesis' goal is also meant to indirectly prove that McBAR is efficient.

5. The fifth goal of the thesis is to directly demonstrate the efficiency of McBAR for solving problems from $\mathcal{L}^2 \subset \mathcal{M}$ ($\mathcal{L}^2$ is defined in Section 1.2.5). Some algorithms may be effective for solving some problems but could be inefficient in some aspects,

e.g. a longer computing time relative to that of other techniques for solving the same problem.

The achievement of the first, second and third thesis' goals proves and characterises the effectiveness of McBAR for solving various problems from $\mathcal{M}$. And, the attainment of the fourth and fifth thesis' goals proves that McBAR is efficient for solving many problems from $\mathcal{M}$.

## 1.4 Methodology

As implied in Section 1.1, CBAR is transformed to become McBAR. The overview of this innovation is presented before elaborating on the methodologies applied to accomplish the thesis' goals. In our previous work [3], CBAR was demonstrated to be inapplicable to solve any problem from $\mathcal{M}$, each set in an environment with increases and without decrease in the total number of tasks. This inapplicability is remedied by upgrading the algorithm of CBAR. In this upgrade, IDs of new tasks are inserted into genotypes (a string of task IDs) when the total number of tasks increases. The resulting genotypes are then evolved to obtain phenotypes as solutions to the problems from $\mathcal{L}^2 \subset \mathcal{M}$. However, it was shown in the same work, and will be demonstrated in Section 5.4, that the performance (solution-searching ability) of this upgraded CBAR for solving the problems starts to degrade when the increase occurs for the first time. To resolve the side-effect of the increase, CBAR is further upgraded. In this second upgrade, the IDs (integers) of tasks in each of the genotypes are mapped so as to minimise their change along each of the genotypes. The fully upgraded CBAR is McBAR.

Let us now present an overview on the McBAR-involving methodologies applied to achieve the above goals of the thesis.

### 1.4.1 The Effectiveness of McBAR

The first thesis' goal is achieved through the separate application of the two approaches referred to as limited and general methods. In the limited method, each technique from $\mathcal{T}$ (defined in Equation 1.8) is applied to solve each problem from $\mathcal{L}^2 \subset \mathcal{M}$. From the obtained solutions, the effectiveness of McBAR is characterised. On the other hand, the general method creates accurate empirical models of the performances of techniques from $\mathcal{T}$ through Response Surface Methodology (RSM) [144]. The models are used to accurately predict the performances of the techniques for solving problems from $\mathcal{O}^2 \subset \mathcal{M}$ (defined in Section 1.2.5). From the predicted performances the effectiveness of McBAR is also characterised. The characterisations performed by both methods take into account the dynamics of the environment in which their respective associated problems are set.

Let us now justify the use of the limited and general methods. As explained in Section 1.2.4, each problem from $\mathcal{L}^2$ is a sequence of static sub-problems $l_i^2$ (defined through Equation 1.7), each set in a snapshot taken immediately after the $i^{th}$ SOSA of a dynamical environment in which the problem is set. In the sequence, any consecutive sub-problems (e.g. $l_3^2$ and $l_4^2$) differ on some parameters and/or categories (e.g. total number of tasks, task duration and resource availability status) by a small or large magnitude. Let this magnitude be referred to as the *consecutive sub-problem attribute difference magnitude* (CSADM). The limited method is applied to characterise the performance of techniques from $\mathcal{T}$ for solving problems from $\mathcal{L}^2$, where each problem has a small or large CSADM. Note that $|\mathcal{L}^2| = 30$ and $|\mathcal{O}^2| = 506,250,000$, i.e. the limited method takes into account fewer problems than the general method.

The CSADMs of the problems taken into account by the general method (i.e. problems from $\mathcal{O}^2$) are smaller than the maximum CSADM of the problems taken into account by the limited method. Although desirable, impractically heavy computational expense is required to build empirical models which accurately predict the performance of a technique from $\mathcal{T}$ for solving problems with large CSADM. Thus, to take into account the problems with a large CSADM, the limited method is devised; and to take into account numerous

problems the general method is devised. Note that, in the preliminary stage of empirical model creation through RSM, the general method utilises the solutions to problems from $\mathcal{L}^2$, i.e. problems taken into account by the limited method. This is the other reason for devising the limited method.

### 1.4.2 Tri-Objective Problems

As presented in Section 1.2.5, each sub-problem of each problem from $\mathcal{P}^3$ has three objectives. One of these objectives is to minimise the probability *Prob* that a schedule becomes infeasible due to environmental changes. Thus, in determining a schedule as a solution to the problem, the changes are anticipated so that when they occur the probability will be minimal. However, as McBAR does not have the ability to anticipate, its core algorithm is extended to possess this ability. This extended core algorithm is denoted as McBAR-P.

McBAR-P applies the *Monte Carlo Simulation* (MCS) for the anticipation. MCS is a suitable technique for problems with large dimensions such as the sub-problems of the dynamic problems from $\mathcal{L}^3$. However, MCS is computationally expensive for solving such sub-problems. Thus, only a few dynamic problems are tackled by McBAR-P and they differ from those taken into account by the limited method only by each having the additional objective to minimise the probability *Prob*.

To achieve the second thesis' goal, the duration, financial worth and other properties of schedule $S$ as solution obtained by McBAR are compared to those of the schedule $P$ determined by McBAR-P, where schedules $S$ and $P$ are solutions to sub-problems of problems from $\mathcal{L}^2$ and $\mathcal{L}^3$, respectively, and these sub-problems differ only by the above-mentioned additional objective. This comparison is made under the dynamics of the environment in which the problems are set, to manifest the effectiveness of McBAR-P under these dynamics. Prior to the comparison, the EA-related parametric values that optimise the performance of McBAR-P for solving the tri-objective problems from $\mathcal{L}^3$ are determined.

### 1.4.3 Resiliency of Techniques

The general method explained in Section 1.4.1 creates empirical models through RSM that characterise the techniques from $\mathcal{T}$ (defined in Section 1.2.6). Our preliminary investigations showed that a single binomial (in algebraic terms) empirical model cannot accurately predict the performance, for solving problems from $\mathcal{O}^2 \subset \mathcal{M}$ ($\mathcal{O}^2$ is defined in Section 1.2.5), of the technique that corresponds to this model. A different model is then created, composed of several binomial sub-models, each accurate to predict the performance of the technique that corresponds to the composite model, but the performance for solving any problem is from a subset of $\mathcal{O}^2$ only. However, this composite model is accurate to predict the performance of the technique for solving any problem from $\mathcal{O}^2$. This type of composite model is utilised to determine the limitations on the effectiveness of the techniques from $\mathcal{T}$, i.e. to achieve the third thesis' goal.

The third thesis' goal is achieved by applying Lagrange optimisation [91] on the composite models to determine the resiliency of techniques from $\mathcal{T}$. The resiliency of a technique is a measure of the limitation on the effectiveness of a given technique from $\mathcal{T}$. It is defined as the degree of change in an environment by which the performance of the given technique, for solving problems from $\mathcal{O}^2$ set in this environment, deteriorates to a given level above the performance of the benchmark technique from $\mathcal{T}$ for solving the same problem. The benchmark technique is one that is proven to have a performance, for solving problems from $\mathcal{L}^2$, inferior to those of EA-based techniques from $\mathcal{T}$.

### 1.4.4 Component Legitimisation

As explained in Section 1.3, the confirmation of the legitimacy of the components of McBAR indirectly proves that McBAR is efficient. The concept utilised in this legitimisations is as follows. Suppose technique $A$ differs primarily from technique $B$ on some components. If, in general, $A$ performs better than $B$ for solving given problems then the components of $A$ *distinct* from $B$ are regarded as relevant to $A$ to solve these prob-

lems. To exemplify this principle, consider the following. Being a variant of EA, McBAR utilises crossover and mutation operators. To legitimise the use of these operators, the performance of McBAR is compared to that of a significantly different type of heuristic, referred to as the *Estimation of Distribution Algorithm* (EDA) [119]. Instead of using EA operators, EDA makes use of sampling and estimation of *Probability Density Functions* (PDFs) to create its next-generation population [119]. Now, if in most problems from $\mathcal{L}^2 \subset \mathcal{M}$, McBAR performs better than EDA then the EA operators used in McBAR are considered to be relevant/legitimate components of McBAR to solve problems from $\mathcal{L}^2$ (defined in Section 1.2.5). Thus, some techniques from $\mathcal{T}$ other than McBAR are crafted and their performances for solving problems from $\mathcal{L}^2$ are compared to that of McBAR to legitimise the constituting sub-algorithms of McBAR, i.e. to fulfil the fourth thesis' goal.

Note that some EA-related parametric values of McBAR are determined in order to highly enhance the performance of McBAR. These values are determined using the method of steepest ascent [31] with RSM. Thus, according to the definition of efficiency in Section 1.3, along with the high performance of McBAR and the to-be-proven legitimacy of the components of McBAR, the other purpose of the presented methods in this sub-section is to indirectly prove that McBAR is efficient.

### 1.4.5 Intelligence of Algorithms

Some machines can be perceived to possess intelligence; IBM's Watson, for example, defeated human competitors in answering questions in a TV show entitled, "Jeopardy". The apparent intelligence of Watson is primarily due to its computer program/algorithm [158]. Thus, it is supposed that its algorithm possesses a degree of intelligence. Intelligence of an algorithm was measured in various ways [121]. The measure of intelligence utilised in the thesis is the *Universal Intelligence* (UI) [121]. One of the bases by which UI measures algorithm intelligence is algorithm length, which is related to the number of Central Processing Unit (CPU) cycles required to finish executing this algorithm [121]. Another basis is on the reward (e.g. system gain) given after the execution. Guided by

the definition of efficiency in Section 1.3, the efficiency of a technique from $\mathfrak{T}$ is measured in terms of its performance and the number of CPU cycles spent by a computer in executing its algorithm intended to solve a given problem from $\mathcal{L}^2 \subset \mathcal{M}$. Advocated by the commonality between the concepts behind UI and efficiency, UI is utilised to measure the efficiency of any technique from $\mathfrak{T}$ in solving the problems from $\mathcal{L}^2$. The fifth thesis' goal is attained by using UI to compare the efficiency of McBAR to those of the other techniques from $\mathfrak{T}$.

## 1.5 Contributions

The research undertaken for the thesis yielded six major contributions; five of which correspond to the thesis' goals enumerated in Section 1.3:

1. The first major contribution is the research into the creation of McBAR. Through this research, various methods to remedy the inapplicability of CBAR to solve any problem from $\mathcal{L}^2 \subset \mathcal{M}$ are investigated. This investigation found gene-insertion into genotypes to be the most suitable among all the remedying methods, where the inserted genes correspond to new tasks in the environment in which the problem is set. The resulting deficiency in the performance of CBAR due to this insertion is analysed and that leads to its further improvements. As noted in Section 1.4, the final form of CBAR is renamed as McBAR.

2. The second major contribution refers to the first thesis' goal and is the research into the approach to show the effectiveness (measured in terms of the performance) of McBAR in solving many problems from $\mathcal{M}$. In this way various approaches for this purpose are investigated. This investigation found the limited and general methods to be the most practically appropriate approaches for the purpose. The application of these methods showed the performance of McBAR to be generally superior compared to those of techniques from $\mathfrak{T}$ different to McBAR. This superior performance is most profound when solving sub-problems – of problems from $\mathcal{L}^2 \cup$

$\mathcal{O}^2 \subset \mathcal{M}$ – that are set in snapshots taken just after the first increase in the total number of tasks in the environment in which the problems are set. This superior performance generally remains despite the changes to the environment after the first increase. Findings in the research were published in [3, 5–7].

3. The third major contribution refers to the second thesis' goal and is the research into the failure of implementing subsections of schedules due to environmental changes. As presented in Section 1.4.2, the minimisation of the probability of this failure to occur is one of the objectives of each of the tri-objective sub-problems of problems from $\mathcal{L}^3 \subset \mathcal{M}$. This research demonstrated the good performance of the core algorithm of McBAR for solving the sub-problems, where this performance is measured in terms of the duration, cost of implementation and economic relevance of the generated solutions. Findings in the research were published in [8].

4. The fourth major contribution refers to the third thesis' goal and is the research into the limitations of EA techniques. Several measures used to define the limitations of EA techniques are investigated to arrive at resiliency (defined in Section 1.4.3) as the most suitable measure to define the limitation of techniques from $\mathcal{T}$. The research finds McBAR to be the most resilient among the techniques under the dynamics of problems from $\mathcal{O}^2 \subset \mathcal{M}$. Findings in the research were published in [4–6].

5. The fifth major contribution refers to the fourth thesis' goal and is the research into the design of the algorithms of techniques from $\mathcal{T}$ other than McBAR, namely the algorithms that are useful for investigating the legitimacy of the sub-algorithms of McBAR. The research found all the sub-algorithms of McBAR to be legitimate, either using the limited or the general method described in Section 1.4.1. Findings in the research were published in [5–7].

6. The sixth major contribution refers to the fifth thesis' goal and is the research into the efficiency measure of EA algorithms. Various efficiency measures are investigated that lead to the selection of Universal Intelligence as the most suitable measure

of efficiency of techniques from $\mathcal{T}$. Among all the techniques, this research justified McBAR as the most efficient technique in solving problems from $\mathcal{L}^2 \subset \mathcal{M}$.

Several portions of the above researches undertaken for this thesis were published in three conference proceedings and four journal articles. The research that pertains to some topics in Section 5.1 is published in the following journal article:

- L. T. Bui, Z. Michalewicz, E. Parkinson, and M. B. Abello. Adaptation in Dynamic Environments: A Case Study in Mission Planning. *IEEE Transactions on Evolutionary Computation*, 16(2):190-209, 2011.

The research that leads to the topics in Chapter 6 is published in the following conference proceeding and journal article:

- M. B. Abello, L. T. Bui, and Z. Michalewicz. An adaptive approach for solving dynamic scheduling with time-varying number of tasks part – I. In *2011 IEEE Congress on Evolutionary Computation*, 1703-1710, 2011.
- M. B. Abello and Z. Michalewicz. Multi-Objective Resource-Constrained Project Scheduling with a Time-varying Number of Tasks. *The Scientific World Journal*, DOI: 10.1155/2014/420101, 2014.

Some parts of the research related to Chapter 7 is published in the following conference proceeding:

- M. B. Abello, L. T. Bui, and Z. Michalewicz. A Reactive-Proactive Approach for Solving Dynamic Scheduling with Time-varying Number of Tasks. In *2012 IEEE Congress on Evolutionary Computation*, 1-10, 2012.

The research described in Chapter 8 and Sections 9.1 to 9.3 is published in the following journal article:

- M. B. Abello and Z. Michalewicz. Implicit Memory-Based Technique in Solving Dynamic Scheduling Problems Through Response Surface Methodology - Part I: Model and Method. *International Journal of Intelligent Computing and Cybernetics*, 7(2):114-142, 2014.

- M. B. Abello and Z. Michalewicz. Implicit Memory-Based Technique in Solving Dynamic Scheduling Problems Through Response Surface Methodology - Part II: Experiments and Analysis. *International Journal of Intelligent Computing and Cybernetics*, 7(2):143-174, 2014.

The research related to Section 9.2 is not included in the thesis due to scope limitations but is published in the following conference proceeding:

- M. B. Abello, L. T. Bui, and Z. Michalewicz. An adaptive approach for solving dynamic scheduling with time-varying number of tasks part – II. In *2011 IEEE Congress on Evolutionary Computation*, 1711–1718, 2011.

## 1.6 Thesis Outline

The structure of the thesis is as follows. Chapter 2 provides background knowledge useful to understand topics utilised in the attainment of the thesis' goals. This chapter begins by exploring Evolutionary Algorithms (EA) since this is the backbone of all analysis in the thesis. Then, a particular type of EA, referred to as Genetic Programming, is presented. *Resource-Constrained Project Scheduling* (RCPS) is discussed next and is followed by the presentation about the optimisation on scheduling problems set in dynamic environments which are related to the test environments utilised in the thesis. The discussion progresses to the class of techniques referred to as *Multi-Objective Optimisation* (MOO). One of these techniques is utilised to produce solutions for the memory-based McBAR. Next, information about the *Estimation Distribution Algorithm* (EDA) is provided. The enumerated topics are helpful in understanding how the first and fourth thesis' goals are

attained through the limited method. The concepts related to the probability of schedule infeasibility are then explained. These concepts are useful for understanding the accomplishment of the second thesis' goal. *Response Surface Methodology* (RSM) is explored next. This technique is useful for building models of the performance of techniques from $\mathcal{T}$. Using these models the first and the fourth thesis' goals are achieved applying the general method, and the third thesis' goal is also realised. This chapter concludes with a discussion of *Reinforcement Learning* (RL) [190] and of the measure of intelligence derived from RL, which is the measure that is applied to achieve the fifth thesis' goal.

Chapter 3 reviews current literature related to the thesis. The reviewed topics are sequenced in an order and purpose similar to that in Chapter 2 from the section on Genetic Programming to the section on measure of intelligence. This chapter concludes with the informative review on Multi-Agent Systems as applied to scheduling.

Chapter 4 provides information on the planning approach for a dynamic RCPS environment and then explains the nature of this environment which is referred to as the *Military Mission Environment* (MME). All scheduling problems (contained in $\mathcal{M}$) considered in the thesis are set in the MME, these problems are important for the fulfilment of all the thesis' goals. In particular, this chapter describes the problems taken into account by the limited and general methods. The mathematical formulation of problems from $\mathcal{L}^2$, $\mathcal{L}^3$ and $\mathcal{O}^2 \subset \mathcal{M}$ is explored in the Appendix.

Chapter 5 explains the algorithm of CBAR then demonstrates how CBAR becomes inapplicable for solving any problem from $\mathcal{M}$ that is set in the MME. Information is then provided on how CBAR is upgraded to remedy the inapplicability, followed by an explanation of the effect of this remedy on the performance of the upgraded CBAR for solving the problems. A resolution which revises CBAR further is presented next. The final form CBAR is McBAR whose algorithm is explained last.

Chapter 5 also describes the techniques from $\mathcal{T}$ other than GIBAR (an upgrade of CBAR) and McBAR. A significant portion of this description is about EDA being applied to solve problems from $\mathcal{L}^2$. Notably, this application is absent in the literature. These

other techniques are important to confirm the legitimacy of the sub-algorithms and other characteristics of McBAR.

Chapter 6 presents the results of the investigations made through the limited method on the performance of techniques from $\mathcal{T}$ for finding solutions to all problems from $\mathcal{L}^2$. It elaborates the dynamic effects of these problems on performance, and discusses the legitimisation of the sub-algorithms of McBAR through the performance. Thus, this chapter presents the partial fulfilment of the first and fourth thesis' goals through the use of the limited method.

Chapter 7 explores the algorithm and the performance of McBAR-P (an extension of the core algorithm of McBAR) for solving the tri-objective problems from $\mathcal{L}^3 \subset \mathcal{M}$. Further, it presents the determination of the performance-enhancing EA-related parametric values of McBAR-P. Thus, this chapter demonstrates the accomplishment of the second thesis' goal.

Chapter 8 provides information on the general method. As explained in Section 1.4.1, the general method creates various composite models through RSM that characterise some techniques from $\mathcal{T}$. It explicates, using the predictions from the models, the establishment of the effectiveness, the exhibition of the dynamical performance and to the legitimisation of some components of McBAR for solving problems from $\mathcal{O}^2 \subset \mathcal{M}$. The general method is thus applied in this chapter to complete the achievement of the first and fourth thesis' goals that was partially completed in Chapter 6.

Chapter 9 elaborates the mathematical definition of resiliency and the determination of the resiliencies of some techniques from $\mathcal{T}$ using the models created in Chapter 8. The third thesis' goal is accomplished in this chapter. This chapter also investigates the efficiency of techniques from $\mathcal{T}$ for solving all problems from $\mathcal{L}^2$. The fifth thesis' goal is attained in this chapter also.

Chapter 10 provides a summary of our findings and future work on this research.

The first to the third columns from the left of Figure 1.1 contain the labels of thesis

goals, citations of contributed publications, and labels of chapters declared in Sections 1.3, 1.5 and 1.6 respectively. The arrows in this figure represent the relationships between the labelled entities and goals. For example, the relationship of goals 1 and 4 to Chapter 6 where these goals are partially accomplished is signified by two arrows from a block labelled 6 at the rightmost column to the blocks with numbers 1 and 4 at the mid-column. Another example, the block labelled [7] at the left column denotes a citation (refer to the Bibliography section) of a contributed publication which partially accomplished the goals 1 and 4.

Through both the limited and general methods and under the dynamic effects of many bi-objective problems from $\mathcal{M}$, the thesis establishes the effectiveness of McBAR for solving these problems, and legitimises the sub-algorithms of McBAR. The thesis also establishes the effectiveness of the core algorithm of McBAR for solving tri-objective problems from $\mathcal{L}^3 \subset \mathcal{M}$. Further, it proves McBAR to be the most resilient among many of the techniques from $\mathcal{T}$ under the dynamics of problems from $\mathcal{O}^2 \subset \mathcal{M}$, and demonstrates McBAR to be the most efficient and intelligent among all of the techniques from $\mathcal{T}$.

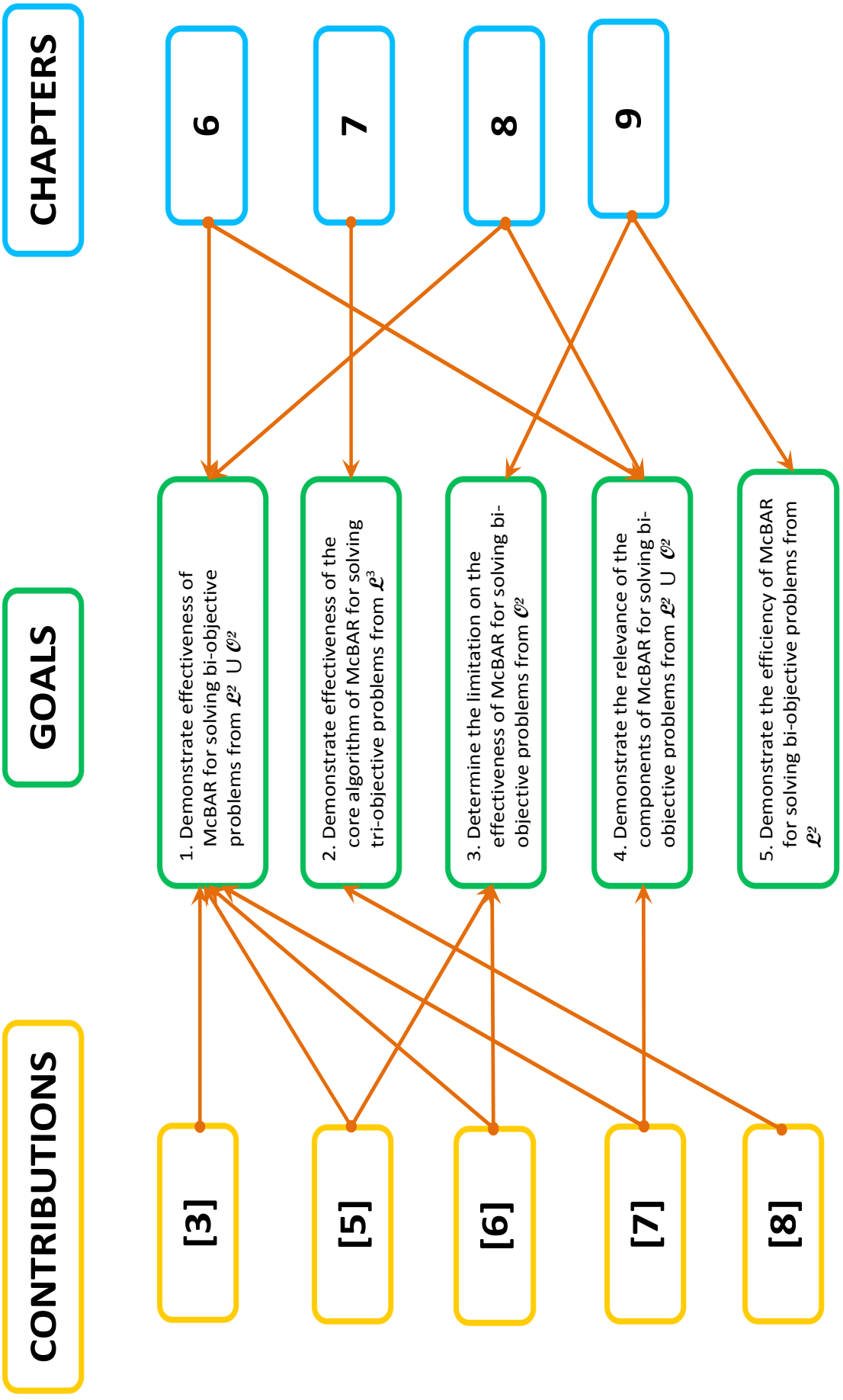



Figure 1.1: Contributions, goals and thesis structure relationships

# Background Knowledge

## Chapter 2

This chapter provides background knowledge useful for understanding the pivotal concepts for the fulfilment of the thesis' goals. Being an important component of most analysis undertaken in the thesis, the fundamental principle of *Evolutionary Algorithm* (EA) is presented in Section 2.1. A particular type of EA, referred to as Genetic Programming, is investigated in Section 2.2. Then, *Resource Constrained Project Scheduling* (RCPS) is elaborated in Section 2.3 together with the algorithm we utilised to generate initial populations for the EA-based techniques from $\mathfrak{T}$ (defined in Section 1.2.6) to evolve. This is followed in Section 2.4 by the explanation on some issues inherent in dynamic environments and approaches to solve the scheduling problems set in these environments. *Multi-Objective Optimisation* (MOO) is explored in Section 2.5 with strong emphasis on a particular multi-objective optimisation algorithm and on the popular quality measure of solutions determined by this algorithm. Section 2.6 provides information on *Estimation Distribution Algorithm* (EDA) and highlights the fundamental difference between EDA and EA. Next, the likelihood of plans or schedules to fail during execution and its specific measure is described in Section 2.7. Section 2.8 explores *Response Surface Methodology* (RSM) and explains the steps in the creation of accurate empirical models. This is followed by the discussion on Lagrange Optimisation in Section 2.9 that is used to determine, through the models, the resiliencies (defined in Section 1.4.3) of some techniques under environmental changes. This chapter concludes in Section 2.10 with the presentation of a measure of intelligence which is used to define the efficiency of techniques.

## 2.1 Evolutionary Algorithms

One of the goals of Artificial Intelligence is to create systems that can mimic some facets of human intelligence [61, 155, 160]. It may be achieved by using some ideas from nature. For example, the observation that humans have tendencies to perform actions that yield more reward than those which do not, becomes the foundation of a field in Artificial Intelligence referred to as, *Reinforcement Learning* (RL) [190]. Another example is related to Charles Darwin's theory of evolution which conjectures that individuals of species that are more fitted to their natural habitat have a greater chance that their offspring will exist into the next generation than other individuals in the same species. The field of *Evolutionary Algorithm* (EA) was founded on this theory [95]. The general algorithm of EA is illustrated in Figure 2.1. Let the population be a set of individuals of a specie and $t$ be an evolutionary generation number. The algorithm is as follows:

1. At $t = 0$, form an initial population $P(t)$ of individuals of a specie. Let its size be $S > 0$.

2. Measure fitness of each individual of $P(t)$.

3. Based on fitness of each individual in $P(t)$, select from $P(t)$ individuals of an intermediate non-empty population $Prn(t)$. Individuals of $Prn(t)$ are called parents.

4. Apply variation operators to parents in $Prn(t)$ to produce offspring that form another intermediate non-empty population $Pch(t)$. Individuals of $Pch(t)$ are called offspring. The offspring inherit characteristics or variants of characteristics of their parents in $Prn(t)$. Note that some of these offspring could be identical to their parents.

5. Measure fitness of each offspring in $Pch(t)$.

6. Based on fitness of each offspring in $Pch(t)$ and each parent in $Prn(t)$, select from these two populations a new population $P(t + 1)$ which must have a size of $S$. Population $P(t + 1)$ is considered to be the next generation where $t \rightarrow t + 1$.

7. Repeat steps 3 to 6 until a terminating condition is reached. The loop, defined by these steps, represents one evolutionary cycle/generation.

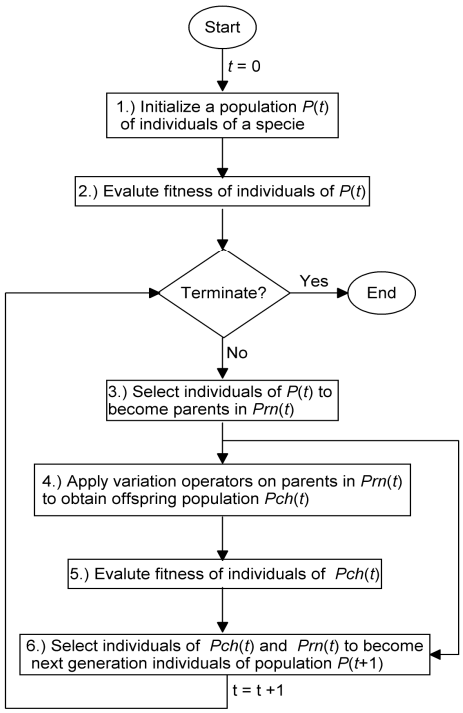



Figure 2.1: Evolutionary Algorithm

EAs were applied to solve problems in various fields, such as arts [107], sociology [17], engineering [157] and finance [39]. These EAs have their own design of the basic algorithm in Figure 2.1. The following sub-sections discuss sample designs related to the key aspects of the basic algorithm.

### 2.1.1 Individual Representation

An individual considered in step 1 of Figure 2.1 could be implemented as a string of genes which are computer programming entities, such as binary or float values, algorithms, object-oriented programming classes. Each gene is a code for an individual's characteristic/s and feature/s; and the string itself is called a genotype. A sample genotype is shown in Figure 2.2 as an ordered set of integers. Let these integers correspond to IDs of tasks in a military mission to defeat an enemy. For example, the fourth gene in the genotype

contains a value of 7 which, for example, is the ID of the task to bomb a particular enemy position using 4 fighter jets for 18 units of time.

$$G = \langle 1, 2, 3, 7, 4, 8, 9, 5, 10, 11, 6, 12 \rangle$$

Figure 2.2: Genotype as an ordered set of integers

Genotype is a code for an entity called phenotype. For example, the genotype in Figure 2.2 is a code for a schedule/phenotype illustrated in Figure 2.3. In this figure, the horizontal axis signifies time, rectangular strips represent tasks, strip width denotes task duration, strip label indicates task ID and $R_i$ represents resources of similar type utilised by tasks whose corresponding smallest enclosing rectangle is beside $R_i$. Expressed by this figure, tasks 1, 2 and 3 are started simultaneously at time 0, followed by task 7 started at the time of 14 units; tasks 4, 8 and 9 simultaneously start at time 16 units, and so on.

Usually, procedures are performed to convert genotypes into phenotypes. For example, to convert genotype $G$ above into a schedule, its elements $e$ are considered consecutively. The starting time of the task with a considered ID $e$ is set to be the earliest time later than the maximum end times of all its predecessors with IDs in $G$. Further, this starting time is such that there are enough available resources to utilise by task with ID $e$ over the entire execution period of this task.

### 2.1.2 Initial Population

Basically, before starting an evolutionary process in EA, a population is created called an initial population, e.g. $P(0)$ in step 1 of Figure 2.1. It is formed in various ways. For example, its individuals are copied from a database of results from previous runs of an algorithm [161], derived from attributes of past populations determined by an algorithm [40], or generated semi-randomly [88]. As an example of the last approach, forty ordered sets of 12 elements can be formed by the random permutations of elements of the genotype in Figure 2.2 to produce an initial population. The initial population can have a significant impact on EA's performance in determining solutions to some problems [40,131,161,224].

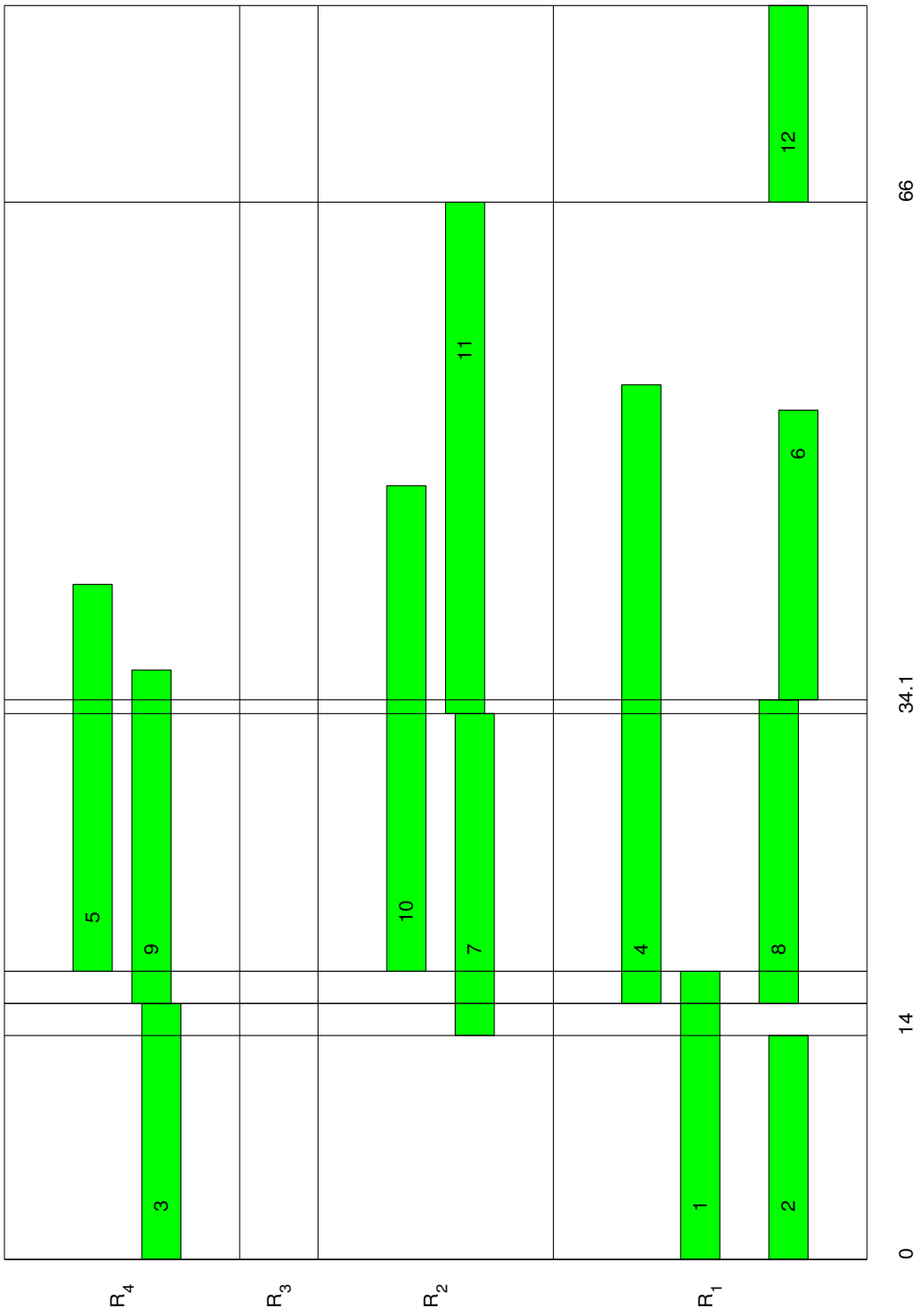



Figure 2.3: Sample Schedule

### 2.1.3 Fitness Measure

The measure in steps 2 and 5 of Figure 2.1 is a function that maps a phenotype to value/s used in the selection process in steps 3 and 6 of the same figure respectively. This measure, for example, could be on a species' ability to fit in a habitat. Based on this, the measure is referred to as the fitness measure or, simply, fitness; the mathematical function that performs the measurement is called the fitness function or objective function; and the graphical presentation of this function is called the fitness landscape. For example, the fitness function $f$ yields the cost of implementing a given schedule whose corresponding genotype is $G$ in Figure 2.2. And, it is expressed as,

$$f(G) = \sum_{i=1}^{12} \sum_{j=1}^{12} \delta_{i,j}(G) c_{i,j} \tag{2.1}$$

where $c_{i,j}$ is the cost of moving items of resource type $R_k$, indicated in Figure 2.3, from task $i$ to task $j$; $i$ and $j$ are IDs in $G$; index $k$ is dependent on the indices $i$ and $j$; and $\delta_{i,j}(G)$ is a unity if the items are required to be transferred between the tasks and zero otherwise. Note that $c_{i,j}$ varies with the number of items being moved.

### 2.1.4 Modifying Operations

Parents' genotypes are mixed, copied and/or modified to form their offspring's genotypes resulting in offspring that inherit modified or identical features of their parents. Genotype-modifying operators referred to in step 4 of Figure 2.1 take numerous forms [83,84] and are well discussed in the EA literature. The thesis focuses only on two particular modifying operators: recombination and mutation operators. We refer to a recombination operator as a crossover operator.

#### 2.1.4.1 Crossover Operator

Crossover operators mix genetic materials from both parents. For example, in Figure 2.4, both parent genotypes $PA$ and $PB$ are cut after their ninth gene. The second section $SA_2$ of parent $PA$ becomes the first section of offspring $CA$ and the first section $SB_1$ of parent $PB$ becomes the second section of the same offspring. This type of crossover involves only one randomly chosen cut point.

$$PA = \left[\overbrace{7,3,1,6,10,12,9,2,5}^{SA_1},\overbrace{11,4,8}^{SA_2}\right] \quad PB = \left[\overbrace{12,7,5,2,8,11,1,6,9}^{SB_1},\overbrace{4,10,3}^{SB_2}\right]$$
$$CA = \left[\overbrace{7,3,1,6,10,12,9,2,5}^{SA_1},\overbrace{4,10,3}^{SB_2}\right] \quad CB = \left[\overbrace{11,4,8}^{SA_2},\overbrace{12,7,5,2,8,11,1,6,9}^{SB_1}\right]$$

Figure 2.4: One-point crossover

Figure 2.5 illustrates two randomly chosen crossover points at the third and ninth genes of the parent genotypes. In this scheme, offspring $CA$ takes the first part from parent $PB$ for its first part, second part of $PA$ for its second part, and third part of $PB$ for its third part. The process is the same for offspring $CB$ produced with the inverse order: $PA \rightarrow PB \rightarrow PA$.

$$PA = \left[\overbrace{7,3,1}^{SA_1},\overbrace{6,10,12,9,2,5}^{SA_2},\overbrace{11,4,8}^{SA_3}\right] \quad PB = \left[\overbrace{12,7,5}^{SB_1},\overbrace{2,8,11,1,6,9}^{SB_2},\overbrace{4,10,3}^{SB_3}\right]$$
$$CA = \left[\overbrace{12,7,5}^{SB_1},\overbrace{6,10,12,9,2,5}^{SA_2},\overbrace{4,10,3}^{SB_3}\right] \quad CB = \left[\overbrace{7,3,1}^{SA_1},\overbrace{2,8,11,1,6,9}^{SB_2},\overbrace{11,4,8}^{SA_3}\right]$$

Figure 2.5: Two-point crossover

#### 2.1.4.2 Mutation Operators

Mutation operators modify a genotype without the use of other genotypes. This modification could be applied to several genes. One mutation scheme is called a swap mutation [174] which interchanges two genes in a genotype. Figure 2.6 illustrates a particular example of this mutation scheme. In this figure, two genes are randomly picked, say first

and last genes of the parent genotype $Pr$, and then interchanged, thereby yielding an offspring $Ch$.

$$Pr = [7, 3, 1, 12, 10, 6, 9, 2, 5, 11, 4, 8] \quad Ch = [8, 3, 1, 12, 10, 6, 9, 2, 5, 11, 4, 7]$$

Figure 2.6: Mutation of a gene

Different types of genotypes may require different mutation operators, e.g. uniform [133], swap [174] and q-gaussian [193].

### 2.1.5 Selection Criteria

The selection mechanism performed in each of steps 3 and 6 of Figure 2.1 could be implemented in various ways, e.g. roulette wheel, ranking, or tournament selection schemes [83, 84]. In the roulette wheel scheme, individuals in a population are selected with a probability proportional to their fitness. In the tournament scheme, two individuals in a population are chosen randomly, and the one with higher fitness is selected.

### 2.1.6 Termination Condition

The loop in Figure 2.1 terminates when a termination condition is met. For instance, this loop may terminate when the number of repetitions reaches a predefined limit; if maximum fitness value of phenotypes saturates as evolutionary cycles progress; or if population diversity is lost. Several other termination conditions are discussed in [181].

## 2.2 Genetic Programming (GP)

The type of EA described above processes genotypes as individuals typically expressed as strings of genes. There is another type of EA, called *Genetic Programming* (GP), which processes individuals in the form of computer programs. Each of these computer programs is usually represented as a tree whose leaves represent data, e.g. variables and

constants, and whose nodes – where the leaves originated – represent operations, e.g. arithmetic and calculus operations. The leaves and nodes are referred to as terminals and functions respectively and the set composed by them is called, *primitive set*. And the segment connecting a node to another node or to a leaf is referred to as, *link*. The measure of depth of a program-representing tree is the maximum number of links traverse from the topmost node to the node without any linked node.

A tree representation of a sample program is depicted in Figure 2.7. The sample program is constituted of the elements of the primitive set which have scalar multiplication and addition as functions, and constants $a$ and $b$, and variables $x$ and $y$ as terminals. As depicted in the figure, the terminals $x$ and $a$ are used as operands of the multiplication ($\times$) function. The result of this multiplication is added, through the addition ($+$) function, to the product ($\times$) of the terminals $y$ and $b$. This algorithm is applied to all functions and terminals in the tree whereby the end result is the output of the program expressed as,

$$\left(xa + yb\right)\left(x^2 + y^2\right) \tag{2.2}$$

Note that a tree could be composed of linked sub-trees.

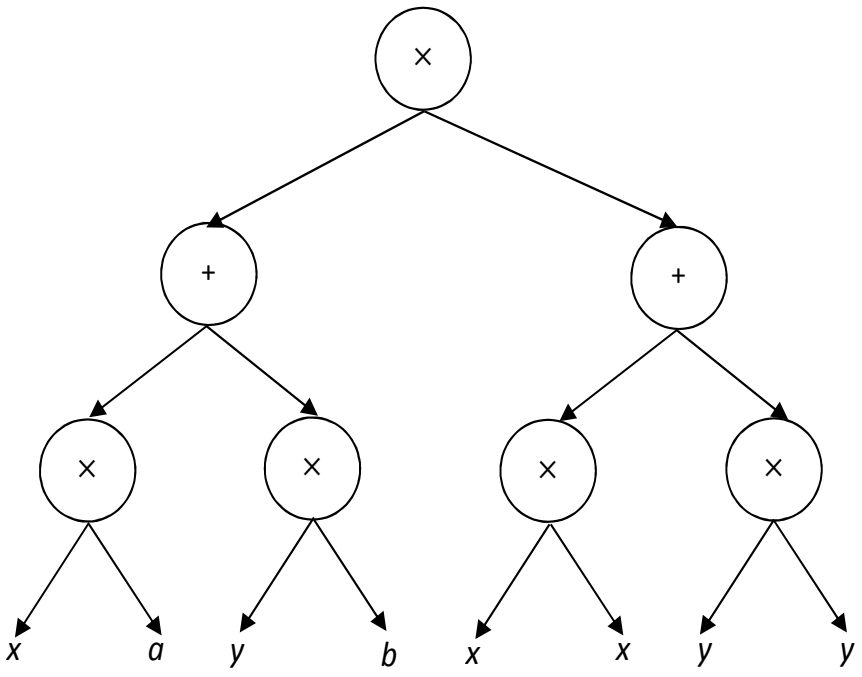



Figure 2.7: Sample GP tree

The general algorithm of GP is closely related to that of the above-discussed type of EA and is the following [118]:

1. At $t = 0$, form an initial population $P(t)$ of programs.
2. Execute each program in $P(t)$ to measure its fitness.
3. Based on fitness of each program in $P(t)$, select from $P(t)$ programs of an intermediate non-empty population $Prn(t)$.
4. Apply variation operators to elements in $Prn(t)$ to produce offspring that form another intermediate non-empty population $Pch(t)$.
5. Repeat steps 2 to 4 on $Pch(t)$ until a terminating condition is reached.

The following describes the components of this general algorithm.

There are other approaches [116, 132] in forming the initial population referred to in step 1. One of these approaches, called the Grow method, is explained in the following. Consider Figure 2.8 and the primitive set,

$$\{+, \times\} \cup \{x, y, a, b\} \tag{2.3}$$

where the left and right subsets are the function and terminal sets respectively. At the first step of the method called Grow method, an eligible (based on the problem being solved) element of a considered primitive set is randomly chosen. If the chosen element is a terminal the method terminates. Otherwise, the chosen function, say $\times$ in Figure 2.8a, has its symbol labelled at the highest node of a tree under construction. Second step, every node added to a tree is attached with links whose number depends on the function indicated at this node. Considering that the $\times$ function is a binary operator then two links extend from the node. Third step, an eligible element is chosen from the primitive set. If the chosen element is a constant, its value is randomly generated and becomes a leaf on arbitrary empty tip of the links. If the chosen element is a non-constant

terminal, it is made as a leaf on arbitrary empty tip of the links. If the chosen element is a function (say + in Figure 2.8b) its symbol is labelled unto a node attached to an empty tip of arbitrarily chosen member of the links. If all links are attached with terminals the method stops. Otherwise, the second and third steps are repeated until there are nodes attached at a given depth or all currently attached links are connected with terminals. If the nodes at the given depth are attached the last step is to attach terminals (as described above) to all of these nodes. The construction process then terminates. Figure 2.8c depicts a fully developed tree/program with depth of two and expressed as,

$$y = x\,(x + 7.3) \tag{2.4}$$

Notice that this equation, which corresponds to the developed program, can be interpreted. Intuitively, all GP-evolved programs can be interpreted. Thus, GP can facilitate the explanation on how the programs solved given problems [147]. The other advantage of utilising GP is that, unlike other types of EA, it would only need to determine one program to solve various instances of a given problem.

There are several methods of measuring program fitness (considered in step 2) [44, 63, 74]. One method would be to average the output values (say $y$ obtained in evaluating Equation 2.4) of the program at many randomly selected independent values ($x$) of this program. Another method, if a tree represents an electronic circuit the power amplification of this circuit can be the fitness measure of this tree.

The selection scheme (pointed in step 3) to obtain the set $Prn(t)$ can be any of the standard selection schemes in EA, e.g. roulette wheel.

Genetic programming differs significantly to other EA approaches in its recombination and mutation operations. For example, in a GP crossover method called sub-tree crossover, two programs, $P$ and $Q$, are randomly picked from a population (e.g. $Prn(t)$). Then, a randomly chosen link in $P$ is cut. The sub-tree obtained in the cutting is connected to the randomly chosen and then cut link in $Q$ to obtain a next generation program

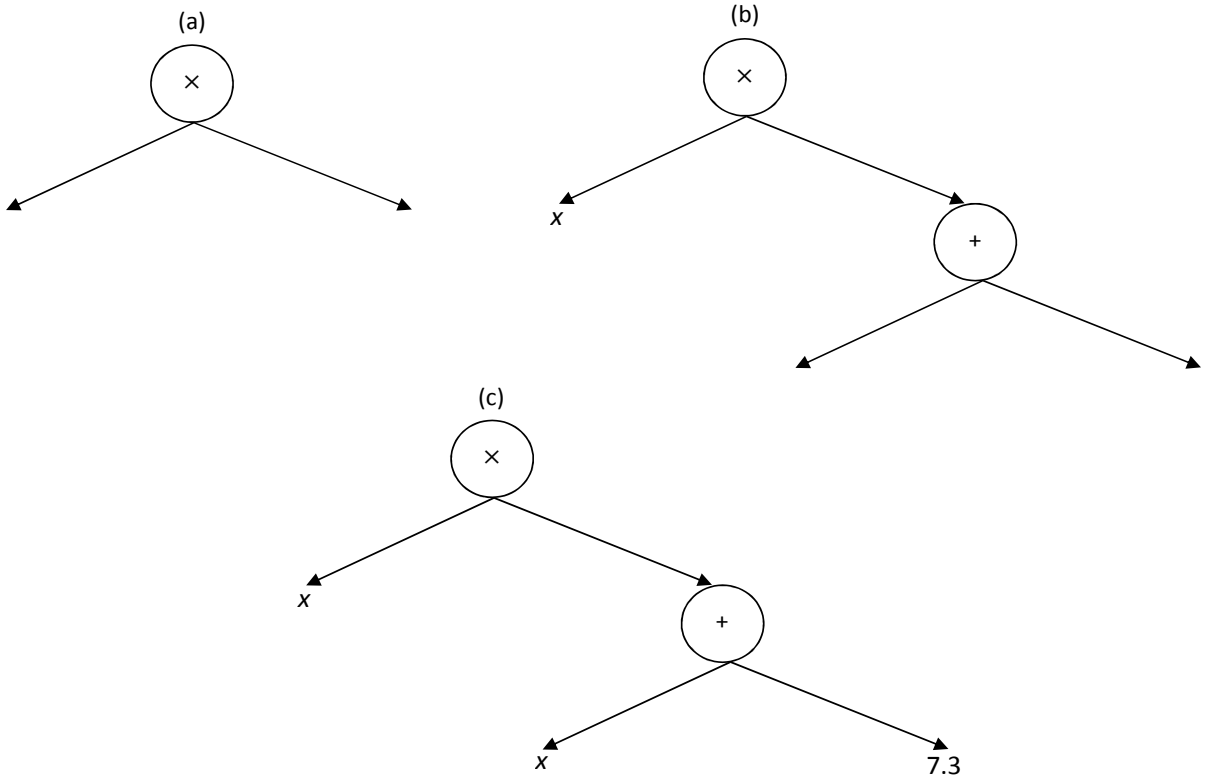



Figure 2.8: Sample program generation: (a) node with multiplication (b) terminals of $x$ and function + (c) Completely generated program

(i.e. element of $Pch(t)$).

The sub-tree mutation is a popular mutation approach in GP. In this approach, a randomly chosen link of a given tree/program is cut and then connected to a randomly generated sub-tree.

These GP components are applied to evolve programs. The evolved programs do not assure the determination of optimal solutions to problems it is applied to solve; rather, GP can quickly determine programs that yield good quality solutions to the problem [43]. In another aspect, GP explores space of programs in its evolutionary process while other types of EA explore space of solutions.

## 2.3 Resource Constrained Project Scheduling

To extend the discussion from the previous chapter on the facets of RCPS, let us discuss different ways of accomplishing a task; a particular representation of the precedence network of tasks (PNT); and a schedule generation method. The mode of task execution is the manner in which it is performed. For example, there could be two ways (two modes) to secure a location during a military operation; a commanding officer may either use bombs or soldiers to accomplish this goal. In RCPS, different task execution modes could entail different combinations of resources utilised. And, task duration could be dependent on the mode in which it is executed [187]. In the thesis, RCPS tasks are executed in single-mode only.

A PNT could be expressed using nodes and arcs. For example, in Figure 2.9, tasks and inter-tasks' precedence relationships are represented by nodes and arcs respectively. The node numbers are IDs of the tasks, and labels "S" and "E" correspond to starting and ending tasks respectively. Note that any PNT that will be considered from here onwards is of the form just described.

A process of generating schedules referred to as *Serial Schedule Generation Scheme* (SSGS) [88] will be explained. But before proceeding, some terms will be defined. Let a genotype be viewed as an ordered set of slots into which IDs of tasks that comprise a schedule will be filled consecutively (e.g. from leftmost slot to the rightmost). Once an ID is filled into the genotype, its corresponding task will be called a scheduled task. A root task is defined as the task from which all other tasks succeed; for instance, the task labeled "S" in Figure 2.9. The root task is considered as a scheduled task even though its ID is not filled into the genotype. Eligible tasks have IDs not yet filled into the genotype and are the immediate successors of scheduled tasks and/or the root task. For example, given the PNT of Figure 2.9, if scheduled tasks have IDs 1, 2 and "S" their immediate successors, based on the figure, have IDs in the sets $\{5, 10, 14\}$, $\{6, 7, 12\}$ and $\{3\}$ respectively. Thus, the set of eligible tasks have IDs in the set $E_t = \{5, 10, 14\} \cup \{6, 7, 12\} \cup \{3\}$.

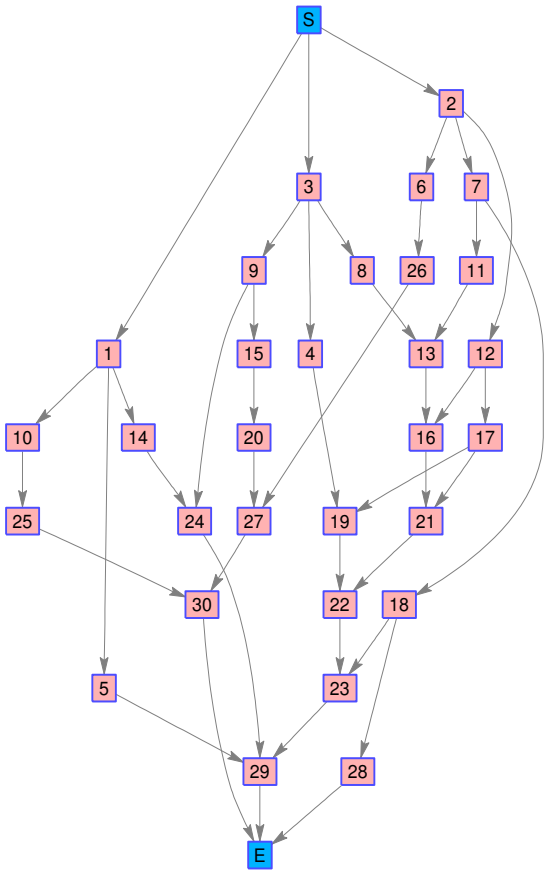



Figure 2.9: Original precedence network of tasks

A version of the SSGS algorithm to create a resource-constrained schedule is depicted in Figure 2.10. It starts by determining the set $E_t$ of IDs of eligible tasks where the only scheduled task is the root task. In the last example, the set of eligible tasks at this stage of the algorithm is $E_t = \{1, 2, 3\}$. A loop is then executed $N$ (the number of tasks to be scheduled) times to fill all slots of the above genotype that starts being devoid of IDs. In the $g^{th}$ cycle of this loop a task ID $j$ is selected randomly from $E_t$ and filled into the genotype at slot indexed $g$. The starting time $st_j$ of the task that corresponds to this ID is set to be the earliest time $t'$ later than the maximum end time (start time $st_i$ plus duration $d_i^r$) of all its predecessors whose IDs comprised the set $Pred(j)$. Further, its starting time $st_j$ is such that there are enough available resources for it to utilise over its entire execution period. The task with ID $j$ is now considered scheduled. And the last step in the loop is to update $E_t$. After $N$ cycles the genotype is completely filled and the starting times of tasks that correspond to IDs in this genotype are determined. Thus, a

schedule is formed.

**Procedure** SSGS
**Begin**
Determine a set $E_t$ of eligible tasks
**For** g=1 to $N$
Randomly select a task ID $j \in E_t$
Put this ID at genotype location $g$
Find $t = max\{0, max\{st_i + d_i^r | i \in Pred(j)\}\}$
Schedule task with ID $j$ at the earliest precedence
and resource-feasible start time $t' > t$
Set $st_j = t'$
Update $E_t$
**End**
**End**

Figure 2.10: Serial Schedule Generation Scheme (SSGS) Algorithm

## 2.4 Dynamic Optimisation Problem (DOP)

As explained in Chapter 1, real-world problems often contain many uncertain and dynamic factors, i.e. military mission planning might have to endure delays or failures of capabilities. The problem of this sort that involves optimisation is referred to as *Dynamic Optimisation Problem* (DOP) [104]. Let us define the following DOP attributes which could change with the environment:

1. *Objective functions*: Consider the objective/fitness function in Equation 2.1 that signifies the cost to implement a schedule. Suppose a task in this schedule is to secure a location. If enemy units unexpectedly arrive at this location during the schedule implementation, it could happen that more items of a certain resource type are needed to secure this location than without these enemies. This new situation can trigger changes in some number-dependent item costs $c_{i,j}$ in the equation. This example illustrates how parametric values of an objective function may change in a dynamic environment. This change could cause a movement of the optima of the objective function to a new location in the search space associated with the optimisation problem that involves the objective function. Also, changes in the environment

could cause changes in number and/or expression of the objective function. The domain of varying objective function has been popular for research in EAs for tracking optima over time, i.e. the moving peak benchmark problem [37].

2. *Variables*: As described in Chapter 1, an example problem for this category is dynamic machine scheduling where unexpected new jobs arrive or old jobs finish and where each job corresponds to a variable in the problem. To date, dynamic job-shop scheduling has been a popular test case for this category [58].

3. *Constraints*: For example, consider change in the configuration of a PNT (embodies precedence constraint). The objective function and variables do not change for this category; however, the constraints do. This category of change does not alter the fitness landscape driven by the objective function, but it will affect the areas of feasibility.

With the presented adverse effects of the changes in the environments where the DOPs are set, it is unlikely that any solution found for these problems would stay valid for a long time. Thus, an adaptive mechanism is required to introduce changes to the current solution to a given DOP to obtain a new high-quality (and feasible) solution. There has been increasing research effort in using EA as the mechanism. This is mainly because of the way EA mimics natural evolution; a number of solutions are allowed to compete and evolve over time. At the time of a bounded change, these solutions (now at fairly different areas of the search space) can easily evolve toward the new optima.

In general, the EA approaches for DOPs are categorised into two broad areas: finding/tracking optima and adaptation. As pointed out in Item 1, tracking of optima may be performed when objective functions change. However, the tracking of optima does not consider the scenario where schedule/solution associated with the objective function is partially executed when the environment changes. Adaptation on the other hand, considers this type of scenario. We discuss these two areas in the following subsections.

### 2.4.1 Finding/Tracking Optima

There are several EA-based methods [42, 142, 200] to search and track optima of varying objective functions. They usually make use of the last population before a problem change as a starting point for the new EA search for the new problem. It should be noted that the change should be bounded; thus its effect on the last associated search space is not radical so that the information in the individuals of the population that corresponds to the last problem is still useful. Three popular approaches are considered for the search and tracking of the optima:

- *Generating diversity after a change*: Population diversity is considered as an important element for EAs to effectively track optima after a problem change. The most natural way to do this is to re-initialise the population after a change. However, with some bounded changes, the new EA search space that corresponds to the new problem is not radically different from the previous one. In such cases, re-initialisation might slow down the EA convergence process; hence, reusing the previous population obtained in solving the previous problems might be a better option. Note that the last population might be driven toward the area surrounding the previous optima/solution for the previous problem. However, with a change, the new optima might be at a different location. Therefore, if we use the last population, it needs some diversity. A popular approach is the use of hypermutation [51] where the mutation rate is set to a high value for a limited number of generations, or to just the first generation and then decreased over time. Note that a very high mutation rate may result in a re-initialisation of the last population, whereas a too low mutation rate does not help to boost diversity of this population.

- *Maintaining diversity during the run*: Diversity is maintained over time to make sure the population is diverse enough to obtain a solution with quality recovered from the effect of change. Typical approaches include the Random Immigrant [82] where new individuals are inserted into the population on a regular basis, the multi-objective-

based method utilising diversity as the second objective [42], and the thermodynamic genetic algorithm where an entropy-based value for niching is used during an optimisation process [142].

- *Multi-population approaches*: Several sub-populations are evolved together to cover several promising areas of the fitness landscape. In this way, it is hoped that the whole system will be able to quickly find new optima when experiencing a change in its environment. Also, with this approach, several local optima can be tracked simultaneously. Some typical examples are self-organising scouts [35], the multinational approach [200], and the shifting balance approach [215].

### 2.4.2 Adaptation

As discussed above, the task of finding and tracking the optimal solution to a problem does not take into account the situation where this solution is partially executed. If some components/tasks of this solution/schedule are finished or ongoing, they cannot be altered or there is a high cost associated with such alteration. In a military operation for example, after a bridge has been destroyed and tanks have been transported over two thousand miles, rescheduling the bridge destruction is impossible and the tank transport is extremely costly. Adaptation, however, takes into account the finished and on-going components. Generally speaking, adaptation is a process of adjusting to the new conditions to keep a system functioning properly. Since the world is dynamic, adaptation is an important process for many existing systems from biological, ecological, to social organisations [168]. After some changes in the weather, species might take several evolutionary cycles to develop new abilities to deal with new weather conditions. A company might need to change some components of their business solution to deal with new market conditions. Although adaptation is a complex process and different from system to system, it needs to be executed in an incremental style in which, in the case of scheduling, current ongoing tasks cannot be cancelled without careful consideration of cost. The following rule has been suggested [127]:

*Schedule revision must judiciously be made to avoid the high cost of altering its ongoing tasks which thus can be preserved.*

Scheduling and planning have emerged as a popular adaptation test problem in which dynamic factors are common, such as the arrival of new jobs (or tasks), cancellation of current jobs, disruption of machines (or resources), and time delays in executing jobs. There have been considerable research efforts on this topic [58, 90, 205, 236]. In general, there are four classes of methods for tackling the adaptation issue.

1. *Robust Proactive*: These methods take into account some assumptions (and anticipations) about uncertainties in environments in order to find the most suitable solution [75]. The assumptions can be on the bound or level of environmental changes, and on the probabilistic distribution of the changes. An example of these methods is the addition of slack time to a schedule to make the quality and feasibility of this schedule more robust to environmental changes. Solutions determined through this method are usually obtained via sensitivity analysis (e.g. using MCS). The methods are also considered as a robustness analysis process [208]. Schedules created through these methods are expected to maintain high quality and be feasible despite environmental changes, i.e. to be robust. In accordance with this expectation they are intended to be implemented entirely regardless of changes in this environment. However, when change in this environment is extreme they could become infeasible [117].

2. *Reactive-Predictive*: In these methods, a pre-optimised solution is used as a baseline for scheduling. Whenever an environmental change happens, this baseline solution is revised or repaired to adapt to new environmental conditions. However, this baseline is determined without any anticipation about present or future environmental uncertainties. The repairing or revising process is usually repetitive and is considered as a local search process [127, 205]. Aside from the cost (e.g. material, fuel and labour) of implementing a schedule, the revision could incur another cost such as penalties in breaching contracts made to material or labour suppliers.

3. *Robust Reactive-Predictive* (RRP): These methods combine Robust Proactive and Reactive -Predictive approaches. Analogous to the Robust Proactive approach, these methods create robust baseline schedules with foresight on future uncertain environmental changes. However, the schedules are revised when the environment changes by a degree beyond a given limit or when the change causes violation of given constraints [11, 30, 58]. The revision may still involve the foresight.

4. *Completely Reactive*: In these methods, there is no pre-optimised solution determined in advance. A new search process is carried out to find a new solution adapting to new environmental conditions [175].

It is usually beneficial that a schedule revised through either method in Items 2 to 4 deviates as little as possible from its original form to avoid the high penalty costs of violating decisions which are dependent on its to-be-executed activities [175]. Further, these methods may be incorporated with machine learning strategies to improve their performance in searching for solutions to problems set in the current environmental state. These strategies utilise information that corresponds to past environmental states. They are categorised as follows [156]:

- *Rote Learning* is a simple memorisation scheme. It records valuable solutions and problems (e.g. $g_2$ in Equation 1.6) set in previous environmental states. The recorded information is then retrieved to assist in solving the problem set in the current environmental state. However, this learning mechanism is not useful when the probability of the environment returning to one of its previous states is small [156].

- *Case-Based Reasoning* records features of solutions and problems. It also forms rules to select its recorded information which is relevant to a current problem. However, if the database of recorded information has a small number of pairs of solution-problem features it is likely to be inadequate to assist in solving a current problem [156].

- *Induction Methods* infer rules from previous information they had accepted and/or produced. However, these rules are not guaranteed to be correct or accurate, i.e.

as in inductive reasoning. For example, after an Artificial Neural Network had been trained [207] using the solutions to previous problems and information on these problems, it can give a prospective solution for a current problem [156]. This prospective solution may become part of an initial population which, after evolving, could yield an optimal solution to the current problem.

- *Classifier Systems* make use of genotypes which are strings of rules. In each genotype, genes are successively used in the iterations of an algorithm to obtain a solution (e.g. schedule) to a given problem. The objective values of this solution become the fitness values of the genotype. EA is applied to evolve the genotypes to obtain a genotype that yields a high-quality solution [156].

Rote Learning and Case-Based Reasoning belong to the class called memory-based approaches explained in Section 1.1. Let us now discuss an important concept behind memory-based approaches. Consider a dynamic environment, say a battlefield. And, suppose sub-problems $g_{prev}$ and $g_{curr}$ (see Equation 1.6) are set, respectively, in previous and current snapshots (defined in Section 1.2) of this environment. If based on a certain definition of distance between sub-problems $g_{prev}$ is close to $g_{curr}$, then the fitness landscape that corresponds to sub-problem $g_{curr}$ could also be close to that of $g_{prev}$. It could then be expected that solutions to these sub-problems are also close, based on another distance definition. By the proximity of these solutions, only a few EA cycles could be required to evolve an initial population that contains solutions or derivatives of solutions to $g_{prev}$, to become solutions to $g_{curr}$ [40, 161]. This expectation underlies several memory-based approaches in EA [40, 131, 161, 224]. Note that the solution-searching ability of memory-based approaches is highly dependent on the diversity of population which they create [36].

## 2.5 Multi-Objective Optimisation (MOO)

Let us now extend the exploration of Multi-Objective Optimisation undertaken in Chapter 1. Let $\mathbf{x} = \{x_1, x_2, \ldots, x_M\} \in \mathbb{C}^M$ be referred to as the decision vector (e.g. genotype) of decision variables $x_i$ (e.g. ID of task in a schedule); $M$ is the number of decision variables; and $f_k(\mathbf{x})$ as the $k^{th}$ objective function which relates a decision vector to the $k^{th}$ objective value (e.g., cost to implement an entire schedule). The objective vector $\mathbf{f}(\mathbf{x})$ for a $K$-objective problem is denoted as,

$$\mathbf{f}(\mathbf{x}) = \langle f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x}), \ldots, f_K(\mathbf{x}) \rangle . \tag{2.5}$$

The concept of dominance of a solution is relevant for comparing the quality of this solution to another solution of the same MOO problem [55]. To explain this concept, consider two sets of indices, $C = \{i_1, i_2, \ldots, i_L\}$ and $D = \{j_1, j_2, \ldots, j_E\}$ where $C \cap D = \emptyset$ and $C \cup D = \{1, 2, \ldots, K\}$. Let the indices be those of the objective functions $f_k$ contained in the expression of $\mathbf{f}(\mathbf{x})$ in Equation 2.5. A solution $\mathbf{x}_1$ dominates another $\mathbf{x}_2$, denoted as $\mathbf{x}_1 \preceq \mathbf{x}_2$, if there exists a non-empty set $C$ where $f_i(\mathbf{x}_1) < f_i(\mathbf{x}_2)$ for all $i \in C$ and $f_j(\mathbf{x}_1) = f_j(\mathbf{x}_2)$ for all $j \in D$, i.e. if there is one or more of the objective functions each yielding an objective value at $\mathbf{x}_1$ less than that at $\mathbf{x}_2$ and if the rest of the objective functions each yield an objective value at $\mathbf{x}_1$ equal to that at $\mathbf{x}_2$. This definition is applicable when objectives are to be minimised. Otherwise, the inequality sign will be reversed and "greater than" will be used instead of "lesser than" in the definition of dominance.

To exemplify dominance, recall from Section 1.2 the three objectives: the minimisation of schedule cost, duration and probability of implementation failure. For any schedule $\mathbf{x}$, let the objective functions $f_1(\mathbf{x})$, $f_2(\mathbf{x})$, and $f_3(\mathbf{x})$ yield schedule cost, duration and the probability respectively. If the cost in implementing schedule $\mathbf{x}_1$ is smaller than that of schedule $\mathbf{x}_2$, i.e. $f_1(\mathbf{x}_1) < f_1(\mathbf{x}_2)$; the duration in accomplishing schedule $\mathbf{x}_1$ is shorter than that of schedule $\mathbf{x}_2$, i.e. $f_2(\mathbf{x}_1) < f_2(\mathbf{x}_2)$; and the probabilities that implementing

$\mathbf{x}_1$ and $\mathbf{x}_2$ will fail are equal, i.e. $f_3(\mathbf{x}_1) = f_3(\mathbf{x}_2)$ then $\mathbf{x}_1 \preceq \mathbf{x}_2$. In this example, the sets of indices $C = \{1, 2\}$ and $D = \{3\}$.

Now, consider the case where $f_3(\mathbf{x}_1) > f_3(\mathbf{x}_2)$. Thus, not all of the three objective functions yield values at $\mathbf{x}_1$ less than or equal to the values they respectively yield at $\mathbf{x}_2$. Therefore, $\mathbf{x}_1$ does not dominate $\mathbf{x}_2$. Following a similar reasoning, the converse is also true, i.e., $\mathbf{x}_2$ does not dominate $\mathbf{x}_1$. In this case, $\mathbf{x}_1$ and $\mathbf{x}_2$ are referred to as *non-dominated solutions*. A set of non-dominated solutions is called a *Non-Dominated Set* (NDS). A set of solutions to any MOO problem contains a NDS [54]. In practice, it is from this NDS that a decision-maker selects one solution guided by his/her experience and intuition, to implement in the field. Let the selected solution be referred to as the *chosen individual* from here onwards. The objective vectors that correspond to elements of NDS form a hypersurface called *Pareto Front* (PF). This hypersurface is exemplified over a bi-objective space in Figure 2.11 and labelled as "PF". In this figure, the circles represent objective vectors $\mathbf{o}_j = \mathbf{f}(\mathbf{x}_j)$ (see Equation 2.5) where $\mathbf{x}_j$ is a decision vector.

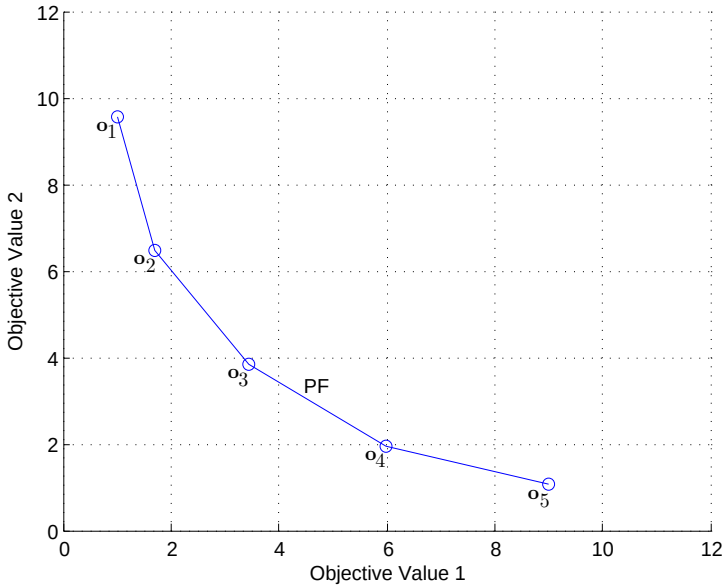



Figure 2.11: Sample Pareto Front

Pareto improvement is the displacement of a decision vector $\mathbf{x}$ in a NDS so as to improve at least one objective value of $\mathbf{f}(\mathbf{x})$ without making other elements of the NDS dominated by the displaced $\mathbf{x}$. If elements of NDS can no longer be Pareto improved then this NDS is called *Pareto Optimal Set* (POS) and its corresponding hypersurface is called

the *Pareto Optimal Front* (POF).

### 2.5.1 Hypervolume

Hypervolume is one measure of the quality of NDS [239]. As shown in the following example, one way to compute hypervolume is as follows. Let $O_v = \{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_5\}$ be a set of objective vectors $\mathbf{o}_j$ that correspond to decision vectors $\mathbf{x}_j$ in an NDS. For each objective vector $\mathbf{o}_j \in O_v$, a hypercube with corners $R_p$ and $\mathbf{o}_j$ is formed where $R_p$ is a chosen reference point that has coordinates greater than those of any point in $O_v$. Figures 2.12(a) to (e) illustrate the hypercubes that correspond to objective vectors $\mathbf{o}_1$ to $\mathbf{o}_5$ respectively. The reference point $R_p$ is indicated by "$\diamond$" in these figures. The volume of the union of all the hypercubes is called, the *hypervolume*, and illustrated in Figure 2.12(f). This approach is appropriate when all objectives in MOO problems are intended to be minimised, such as in the thesis. Other methods to compute hypervolume are also applied in the literature, such as the Hypervolume by Slicing Objectives (HSO) algorithm [213], the LebMeasure algorithm [69], and the MCS-based algorithm [64].

Hypervolume is useful for evaluating the behaviour/performance of a single method for finding solutions under different conditions (e.g. with problem instances), or at different runs or evolutionary cycles of an EA-based algorithm for solving the same MOO problem. It is also useful for comparing the performance of techniques that solve similar MOO problems [239]. In relation to these applications of hypervolume, we define the following:

**Definition 1**: *Let $H_1$ and $H_2$ be hypervolumes of sets $S_1$ and $S_2$ of solutions respectively. Further, let solutions in $S_1$ and $S_2$ be computed by a technique under conditions $C_1$ and $C_2$, respectively. If $H_1 > H_2$ then this technique performs better under condition $C_1$ than under $C_2$.*

**Definition 2**: *Suppose $S_1$ and $S_2$ are solutions computed, respectively, by techniques $T_1$ and $T_2$ under similar conditions, and $H_1$ and $H_2$ are hypervolumes of sets $S_1$ and $S_2$ of solutions respectively. If $H_1 > H_2$ then technique $T_1$ performs better than $T_2$ under this*

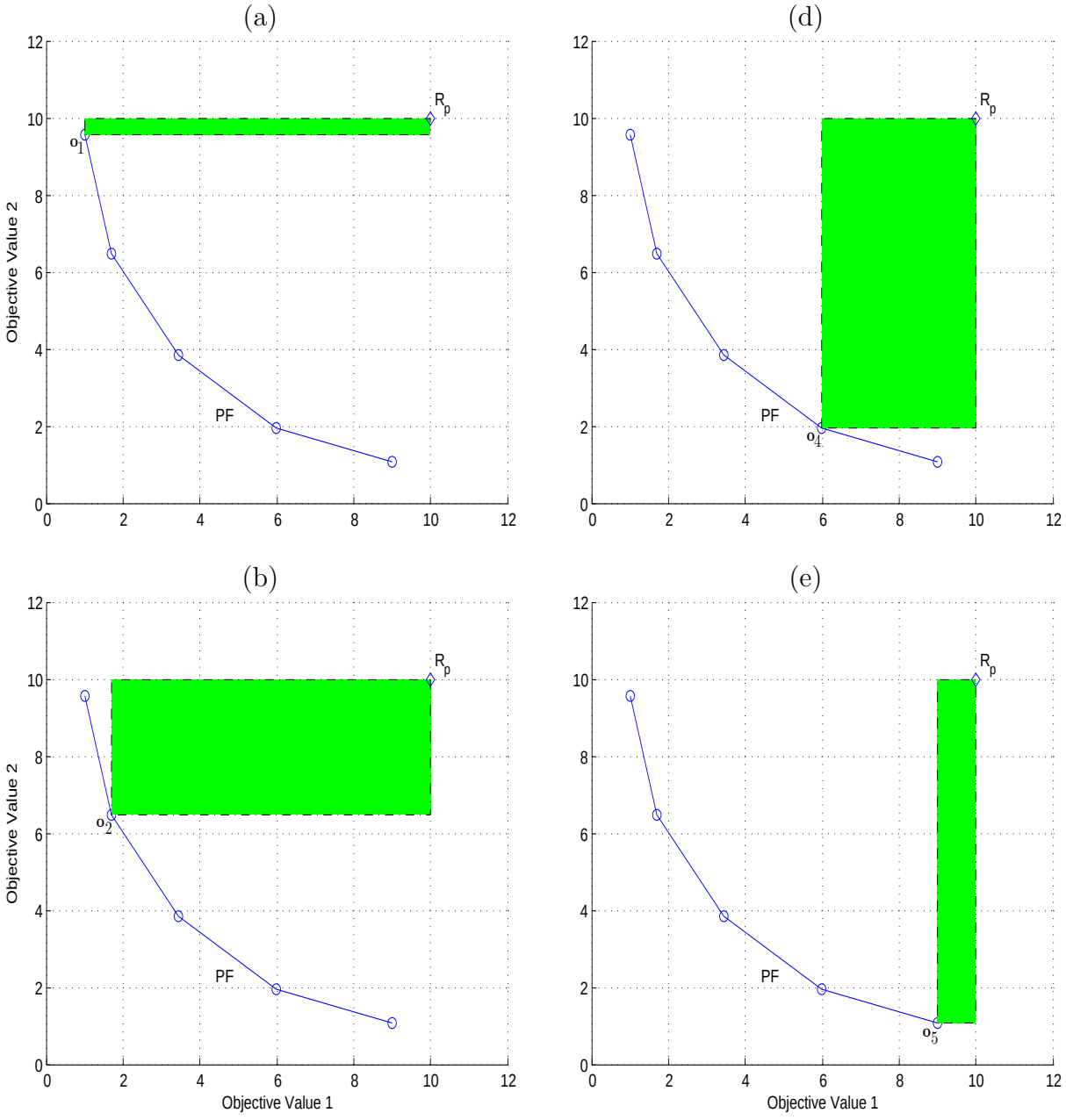



Figure 2.12: Hypercube corresponding to objective vector (a) $\mathbf{x}_1$ (b) $\mathbf{x}_2$ (d) $\mathbf{x}_4$ (e) $\mathbf{x}_5$

*condition.*

### 2.5.2 Set Coverage

Another way to compare the quality of the set $A$ of solutions to a MOO problem, determined by the method $T_A$, to another set $B$ of solutions to the same MOO problem, determined by another method $T_B$, is through the set coverage [55]. Let $D(A, B)$ be the

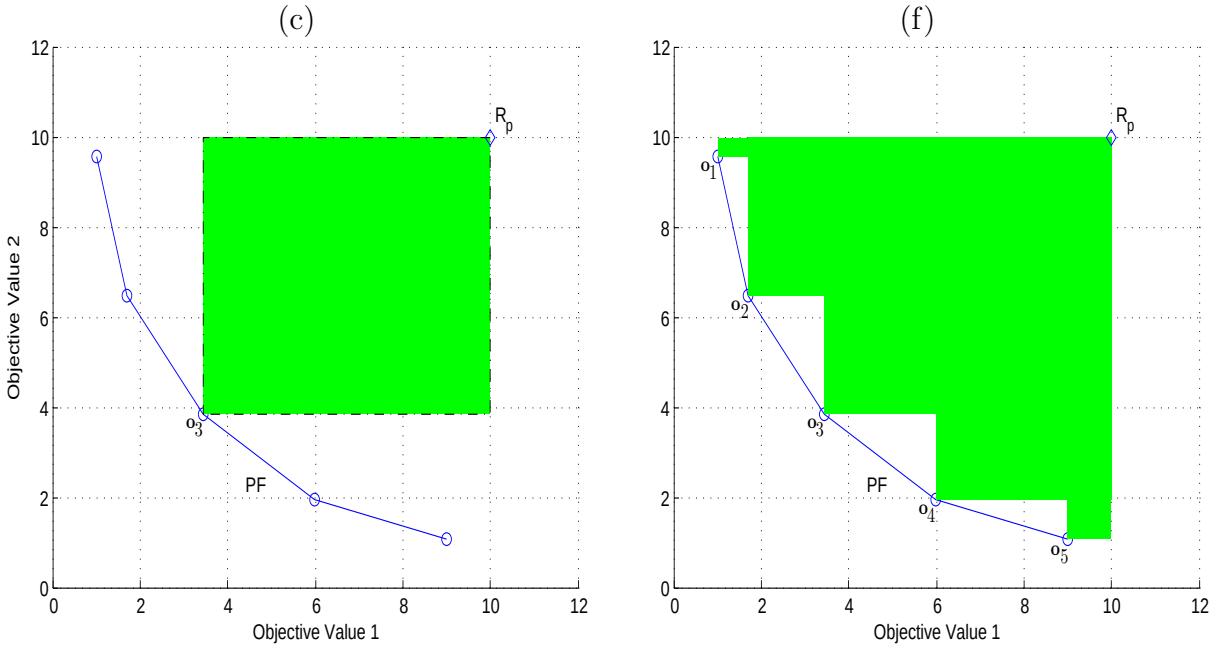



Figure 2.12: (c) Hypercube corresponding to objective vector $\mathbf{x}_3$ (f) Hypervolume

set containing elements of $B$ which are dominated by an element in $A$,

$$D(A,B) = \{b \in B \mid \exists\ a\ \ \in A : a \preceq b\}\,. \tag{2.6}$$

The set coverage is defined as,

$$SC(A,B) = \frac{|D(A,B)|}{|B|}, \tag{2.7}$$

where $|\ |$ is the set cardinality. This definition can also be applied to mono-objective optimisation.

Based on Equation 2.7, the set coverage has a range of $0 \leq SC(A,B) \leq 1$. It will be convenient for later discussions to have a set coverage-related quantity that has a range symmetric around zero. For this purpose we define,

$$dSC(A,B) = SC(A,B) - SC(B,A), \tag{2.8}$$

where $dSC(A,B)$ is referred to as differential set coverage which is anti-symmetric in its

arguments. Assuming that $|A| = |B|$ and based on Equation 2.8, $dSC(A, B)$ has a range of values,

$$-1 \leq dSC(A, B) \leq 1. \tag{2.9}$$

If $dSC(A, B) > 0$, Equation 2.8 implies that $SC(A, B) > SC(B, A)$. By the definition of set coverage in Equation 2.7, the implication indicates that there are more solutions in $B$ dominated by an element from $A$ than the converse. Thus, technique $T_A$ is capable of determining more solutions that dominate those determined by technique $T_B$ than the converse. In view of this, technique $T_A$ is considered to perform better for determining solutions than technique $T_B$. We then take the following definition:

**Definition 3**: *Suppose $A$ and $B$ are sets of solutions determined, respectively, by techniques $T_A$ and $T_B$ under similar conditions. Technique $T_A$ performs better than technique $T_B$ under these conditions if $dSC(A, B) > 0$.*

### 2.5.3 The Multi-Objective Evolutionary Algorithm (MOEA)

As noted in Chapter 1, the RCPS problem is an NP-hard problem and is often approached using modern heuristic methods such as EA [123]. Further, $\mathcal{M}$ is a class of RCPS problems, and each has multiple objectives. A class of EA-based methods applied to solve multi-objective NP-hard problems (such as those in $\mathcal{M}$) is referred to as a *Multi-Objective Evolutionary Algorithm* (MOEA). A popular member of this class is the *Non-Dominated Sorting Genetic Algorithm*-II (NSGA-II) [55]. It is generally efficient for solving MOO problems with small number (two or three) of conflicting objectives [159], such as those being solved in this thesis. In the following, its framework is discussed first followed by its three modules.

#### 2.5.3.1 Non-Dominated Sorting Genetic Algorithm II (NSGA-II)

The NSGA-II algorithm starts with a parent population $P_0$. At evolutionary cycle $i = 0$, these parents are subjected to evolutionary operators and a selection scheme to create

their offspring population $C_i$ of the same size as $P_i$. Next, a module, called *Fast Non-Dominated Sorting* (FNDS) to be described in Section 2.5.3.2, is applied to the union $G_i = P_i \cup C_i$. Then the measure, called *crowding distance*, for each individual in $G_i$ is determined through a process to be described in Section 2.5.3.3. The individuals from $G_i$ are selected through the scheme to be discussed in Section 2.5.3.4 to form the parent population $P_{i+1}$ (of the same size as $P_i$) of the next generation. Now, the offspring population $C_{i+1}$ of the next generation are formed from $P_{i+1}$ by applying the evolutionary operators and selection scheme which selects individuals based on crowding distance or, in some applications, dominance. The steps from the application of FNDS to the creation of $C_{i+1}$ are repeated with $i$ incremented by one at every cycle for a given number of cycles.

#### 2.5.3.2 Fast Non-Dominated Sorting

As pointed out above, *Fast Non-Dominated Sorting* (FNDS) is a module of NSGA-II. It is an algorithm for segregating solutions into groups whose attributes form some of the basis in the selection process of NSGA-II. Given a population $P$, it starts by determining two attributes of each decision vector $\mathbf{s}_j \in P$. The attributes are the number $n_j$ of elements in $P$ that dominate $\mathbf{s}_j$, called *domination count*; and the set $D_j \subseteq P$, called *dominated set*, whose elements are dominated by $\mathbf{s}_j$.

Let $\mathbf{q}_k \in P$ be denoted as $\mathbf{q}_{j,k}$ to indicate that it is related to $\mathbf{s}_j$. If $\mathbf{s}_j \preceq \mathbf{q}_{j,k}$, then $\mathbf{q}_{j,k}$ is stored to $D_j$, which starts from empty. However, if $\mathbf{q}_{j,k} \preceq \mathbf{s}_j$ then the domination count $n_j$ of $\mathbf{s}_j$, that starts from zero, is incremented by unity but $\mathbf{q}_{j,k}$ is not included in $D_j$. These steps are applied for each element of $P$; as a result, each element of $P$ will have its own domination count and dominated set. For example, consider Figure 2.13 where circles, diamonds and squares represent two-dimensional objective vectors labelled by $\mathbf{o}_j = \mathbf{f}\left(\mathbf{s}_j\right)$; $\mathbf{f}$ is defined in Equation 2.5; and $\mathbf{s}_j$ is a decision vector in a population $P$. In particular, consider $\mathbf{o}_2$ indicated in Figure 2.13. Applying the definition of dominance, the domination count of $\mathbf{s}_2$ (corresponds to $\mathbf{o}_2$) is zero and the dominated set of $\mathbf{s}_2$ is $D_2 = \{\mathbf{s}_8, \mathbf{s}_9, \ldots, \mathbf{s}_{15}\}$ whose elements have corresponding objective vectors in the set

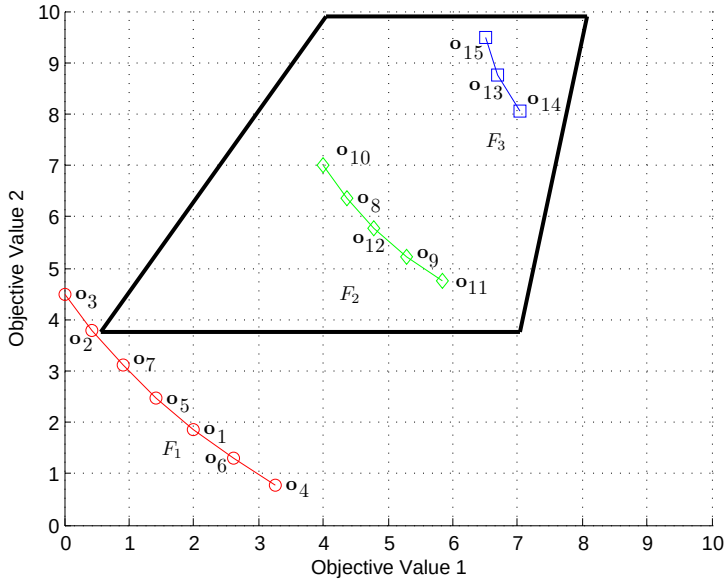



Figure 2.13: Pareto Fronts produced by Fast Non-Dominated Sorting

$\mathbf{O}_2 = \{\mathbf{o}_8, \mathbf{o}_9, \ldots, \mathbf{o}_{15}\}$ enclosed by the thick polygon in Figure 2.13.

After determining the dominated set and domination count for each decision vector in $P$, a set $S_1$, which starts from empty, is formed by transferring to it elements from $P$ whose domination counts are zero. Note that after this transfer the set $P$ is now reduced to a set $P - S_1$ (note the set subtraction). From $S_1$, a set called *First Pareto Front* $F_1$, is determined through,

$$F_k = \left\{ \mathbf{o}_{k,i} \mid \mathbf{o}_{k,i} = \mathbf{f}\left(\mathbf{s}_{k,i}\right),\ \mathbf{s}_{k,i} \in S_k \right\} \tag{2.10}$$

with $k$ set to unity, and where $\mathbf{f}$ is defined in Equation 2.5. To indicate that $\mathbf{s}_i$ corresponds to an objective vector belonging to the $k^{th}$ Pareto Front it is denoted as $\mathbf{s}_{k,i}$. For similar reasons, $\mathbf{o}_{k,i} \equiv \mathbf{o}_i$. As an example of Equation 2.10, the first Pareto Front in Figure 2.13 is, $F_1 = \{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_7\}$. The process of determining the second, third and higher-order Pareto Fronts is discussed next.

A dominated set of $\mathbf{s}_{k,i}$ is denoted as $D_{k,i}$. For example, $\mathbf{s}_{1,2} = \mathbf{s}_2$ has the dominated set $D_{1,2} = D_2$ expressed above. Now, to indicate that a decision vector $\mathbf{q}_m$ is an element of $D_{k,i}$ it is expressed as $\mathbf{q}_{k,i,m}$. The algorithm for determining the first and succeeding

Pareto Fronts is expressed as pseudo-code which utilises the symbols just defined in Figure 2.14. The determination of the second Pareto-Front starts by decrementing by one the domination count of each element $\mathbf{q}_{1,i,m} \in D_{1,i}$ dominated by the decision vector $\mathbf{s}_{1,i} \in S_1$. Note that an element of population $P$ could be a member of more than one dominated set. For example, let $\mathbf{q}$ have a domination count of two and belongs to both $D_{1,2}$ and $D_{1,3}$ which are dominated sets, respectively, of $\mathbf{s}_{1,2}$ and $\mathbf{s}_{1,3}$; both in turn are elements of $S_1$. The domination count of $\mathbf{q}$ becomes one after the domination count of each element in $D_{1,2}$ is decremented and then becomes zero after the domination count of those in $D_{1,3}$ are decremented next. If after all the deductions the resulting domination count of a decision vector $\mathbf{q}_{1,i,m}$ is reduced to zero, then $\mathbf{q}_{1,i,m}$ is transferred from the set $P - S_1$ to $S_2$ which starts empty. This process is applied to all elements of dominated sets $D_{1,i}$ of all $\mathbf{s}_{1,i} \in S_1$ to form $S_2$. Note that after all elements of $P - S_1$ whose domination counts reduced to zero are transferred to $S_2$, the set $P - S_1$ is now reduced to $P - S_1 - S_2$. By Equation 2.10 with $k = 2$, the second Pareto Front $F_2$ is determined from $S_2$. As can be inferred from Figure 2.13, the second Pareto Front $F_2 = \{\mathbf{o}_8, \mathbf{o}_9, \ldots, \mathbf{o}_{12}\}$.

The steps of decrementing and storing above are repeated until each decision vector of $P$ is associated with a Pareto Front. Figure 2.13 showed three Pareto Fronts labelled: $F_1$, $F_2$ and $F_3$ whose elements are differentiated by the type of shape. The order of Pareto Front is called *rank*.

#### 2.5.3.3 Crowding Distance

NSGA-II also provides diversification of individuals in a population as it evolves them. The diversification can facilitate quicker (fewer evolutionary cycles in NSGA-II) determination of optima being a solution to a MOO problem [55]. NSGA-II diversifies a population $P$ by first determining the crowding distance $C_{k,i}$ of each decision vector $\mathbf{s}_{k,i} \equiv \mathbf{s}_i \in P$. The next diversification step is discussed in the next section. To determine $C_{k,i}$, the objective vectors $\mathbf{o}_{k,i}$ (equal to $\mathbf{f}(\mathbf{s}_{k,i})$) are sorted based on their $m^{th}$ components,

**Procedure** Fast Non-Dominated Sorting
**Begin**
Set the first Pareto Front as empty: $F_1 = \emptyset$
Set $S_1 = \emptyset$
**For** each $\mathbf{s}_{1,i} \in P$
Set domination count of $\mathbf{s}_{1,i}$ to zero: $n_{1,i} = 0$
Set dominated set of $\mathbf{s}_{1,i}$ as empty: $D_{1,i} = \emptyset$
**For** each $\mathbf{q}_{1,i,m} \in P$
**If** $\mathbf{s}_{1,i} \preceq \mathbf{q}_{1,i,m}$
Include $\mathbf{q}_{1,i,m}$ to $D_{1,i}$
**Else if** $\mathbf{q}_{1,i,m} \preceq \mathbf{s}_{1,i}$ **then**
Increment domination count of $\mathbf{s}_{1,i}$: $n_{1,i} = n_{1,i} + 1$
**End**
**If** the domination count $n_{1,i} = 0$
Set Pareto Rank of $\mathbf{s}_{1,i}$ to unity
Include $\mathbf{s}_{1,i}$ to $S_1$
Include $\mathbf{f}\left(\mathbf{s}_{1,i}\right)$ to the first Pareto Front $F_1$
**End**
**End**
**End**
Remove $S_1$ from $P$
Set Pareto Rank $k = 1$
Set $S_{k+1} = \emptyset$ and $F_{k+1} = \emptyset$
**While** $P \neq \emptyset$
**For** each element of $\mathbf{s}_{k,i} \in S_k$
**For** each $\mathbf{q}_{k,i,m}$ of dominated set $D_{k,i}$ of $\mathbf{s}_{k,i}$
Decrement domination count of $\mathbf{q}_{k,i,m}$: $n_{k,i,m} = n_{k,i,m} - 1$
**If** $n_{k,i,m} = 0$
Set rank of $\mathbf{q}_{k,i,m}$ to $k + 1$
Include $\mathbf{q}_{k,i,m}$ to $S_{k+1}$
Include $\mathbf{f}\left(\mathbf{q}_{k,i,m}\right)$ to the $(k+1)^{th}$ Pareto Front $F_{k+1}$
**End**
**End**
**End**
Remove $S_{k+1}$ from $P$
Move to the next Pareto Front: $k = k + 1$
**End**
**End**

Figure 2.14: Fast Non-Dominated Sorting (FNDS)

thereby obtaining an ordered set of values,

$$G_{k,m} = \left\langle \mathbf{o}^m_{k,r_m(1)}, \mathbf{o}^m_{k,r_m(2)}, \ldots, \mathbf{o}^m_{k,r_m(L_k)} \right\rangle \tag{2.11}$$

where $\mathbf{o}^m_{k,r_m(l)}$ is the $m^{th}$ component of the objective vector $\mathbf{o}_{k,r_m(l)}$; $l$ is a sequential order of a value $\mathbf{o}^m_{k,r_m(l)}$ in $G_{k,m}$; $L_k$ is the number of elements in the $k^{th}$ Pareto Front $F_k$; and $r_m$ is a sorting function. To exemplify this equation, consider the objective vectors, in the first ($k = 1$) Pareto Front $F_1$ in Figure 2.15, sorted based on their first component (i.e. $m = 1$ that corresponds to the axis labelled "objective value 1" in the figure) from the least to the largest in order to obtain the ordered set of values,

$$G_{1,1} = \left\langle \mathbf{o}^1_3, \mathbf{o}^1_2, \mathbf{o}^1_7, \mathbf{o}^1_5, \mathbf{o}^1_1, \mathbf{o}^1_6, \mathbf{o}^1_4 \right\rangle \tag{2.12}$$

The sorting function $r_1$ for this example maps sequential orders 1 to 7 to subscripts 3, 2, 7, 5, 1, 6 and 4 of elements in $G_{1,1}$ respectively (e.g. $r_1(3) = 7$).

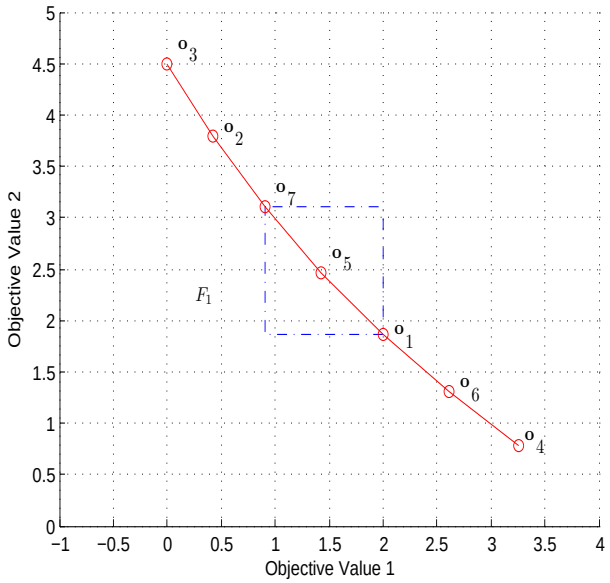



Figure 2.15: First Pareto Front with Cuboid

The crowding distances of the decision vectors (e.g. $\mathbf{s}_3 \equiv \mathbf{s}_{1,r_1(1)}$ and $\mathbf{s}_4 \equiv \mathbf{s}_{1,r_1(7)}$) that respectively correspond to the two objective vectors (e.g. $\mathbf{o}^1_3$ and $\mathbf{o}^1_4$ in Figure 2.15) at the edges of the $F_1$ Pareto Front are set to infinity. Except for these decision vectors, the

crowding distances of decision vectors $\mathbf{s}_{k,i}$ that correspond to objective vectors in the first Pareto Front (i.e. $k = 1$) are,

$$C_{k,i} = \sum_{m=1}^{M} \frac{\left| \mathbf{o}^m_{k,r_m(r_m^{-1}(i)-1)} - \mathbf{o}^m_{k,r_m(r_m^{-1}(i)+1)} \right|}{\left| \mathbf{o}^m_{k,r_m(L_k)} - \mathbf{o}^m_{k,r_m(1)} \right|} \tag{2.13}$$

where $M$ is the number of objectives. This process is repeated for all decision vectors that correspond to objective vectors in Pareto Fronts other than $F_1$.

Let us discuss the implication of Equation 2.13. Consider the objective vector $\mathbf{o}_5$ (that corresponds to $i = 5$ in Equation 2.13) in the first Pareto Front ($k = 1$) illustrated in the two-dimensional ($M = 2$) Figure 2.15. With the number $L_1$ of objective vectors in the first Pareto Front in this figure equal to 7, the crowding distance of $\mathbf{s}_5 \equiv \mathbf{s}_{1,5}$ is,

$$C_{1,5} = \frac{\left| \mathbf{o}^1_{1,r_1(r_1^{-1}(5)-1)} - \mathbf{o}^1_{1,r_1(r_1^{-1}(5)+1)} \right|}{\left| \mathbf{o}^1_{1,r_1(7)} - \mathbf{o}^1_{1,r_1(1)} \right|} + \frac{\left| \mathbf{o}^2_{1,r_2(r_2^{-1}(5)-1)} - \mathbf{o}^2_{1,r_2(r_2^{-1}(5)+1)} \right|}{\left| \mathbf{o}^2_{1,r_2(7)} - \mathbf{o}^2_{1,r_2(1)} \right|} \tag{2.14}$$

Using the definition of $r_m$, the coordinates of the objective vectors in the figure, and the convention, $\mathbf{o}_{k,i} \equiv \mathbf{o}_i$, we obtain,

$$C_{1,5} = \frac{|\mathbf{o}^1_7 - \mathbf{o}^1_1|}{|\mathbf{o}^1_4 - \mathbf{o}^1_3|} + \frac{|\mathbf{o}^2_1 - \mathbf{o}^2_7|}{|\mathbf{o}^2_3 - \mathbf{o}^2_4|} \tag{2.15}$$

The numerator of the first fraction in Equation 2.15 amounts to the length of the horizontal side of the cuboid (square with dash-lined perimeter) in Figure 2.15. And, the numerator of the second fraction amounts to the length of the vertical side of the cuboid. As can be inferred from the figure, the denominators' expressions in Equation 2.15 imply an assurance that all fractions in this equation are normalised, i.e. have values between zero and one. Now, from this equation, if the objective vectors $\mathbf{o}_7$, $\mathbf{o}_5$ and $\mathbf{o}_1$ are crowded relative to the span of the first Pareto Front then the two lengths are small. Consequently, the crowding distance $C_{1,5}$ of $\mathbf{s}_{1,5} \equiv \mathbf{s}_5$ is small. By generalising this conclusion to any decision vector, Equation 2.13 implies that if an objective vector $\mathbf{o}_i$ is in a crowded region of a Pareto Front where it belongs then its corresponding decision vector $\mathbf{s}_i$ has a small

crowding distance, except if it is located at either edge of the Pareto Front. Crowding distance could be monotonically increasing with the distance (e.g. Euclidean distance) between two decision vectors. Thus, $\mathbf{s}_i$ that has small crowding distance could be near to its neighbouring decision vectors and hence could be situated in a crowded region of its associated search space [55].

Now, if all decision vectors in a sub-population have large crowding distances they could be far from each other in their search space. Hence, this sub-population could be diverse. NSGA-II implements diversification by giving a selection preference to decision vectors with large crowding distances to persist to the next generation of its evolutionary cycles. This selection is explained next.

#### 2.5.3.4 Selection

Being a type of EA, NSGA-II has a mechanism to select the next generation population. In this mechanism, the selection of an individual is based on its crowding distance and on the rank of the Pareto Front (a set discussed in Section 2.5.3.2) in which its corresponding objective vector belongs.

The selection process starts by applying FNDS (discussed in Section 2.5.3.2) to a given population $P$ of size $Np$ to yield a set of Pareto Fronts, $\{F_1, F_2, \ldots, F_j, \ldots, F_J\}$ where $J$ is the number of Pareto Fronts. Let $Nsel \leq Np$ be the number of individuals that will persist into the next generation in NSGA-II's evolutionary process. Starting from $S_1$, each $S_j$ (corresponding to $F_j$ in Equation 2.10) is included in ascending rank $j$ in a population $Psel$ – which starts from empty. This inclusion will stop before including $S_q$ which will yield $|Psel| > Nsel$. If at the moment of stopping $|Psel| < Nsel$ then decision vectors in $S_q$ will be sorted in descending order of their crowding distances, and thereby $S_q$ becomes an ordered set $Srt_q$. Next, the set $Sfirst$ of first $Nsel - |Psel|$ elements of $Srt_q$, will be included in $Psel$. Thus, at the end of this step $Psel$ will have exactly $Nsel$ elements. As exemplified in Figure 2.16, if $S_3$ will be included in $Psel$, $|Psel| > Nsel$, such that only its subset $Sfirst$ is included in $Psel$. Now, if at the moment of stopping $|Psel| = Nsel$, there

is no need to include any element of $S_q$.

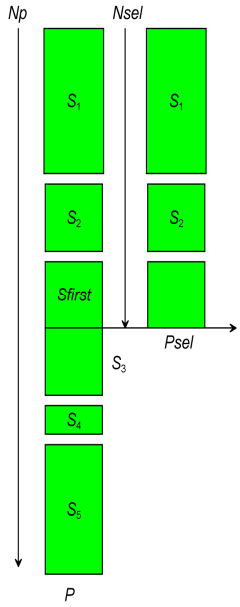



Figure 2.16: NSGA-II selection

Considering that the sorting of elements in $S_q$ is in descending order of crowding distance, elements of *Sfirst* have larger crowding distances than those of $Slast = Srt_q - Sfirst$. From the last paragraph of Section 2.5.3.3, it can be seen that individuals (decision vectors) in *Sfirst* can be more diverse than those of *Slast*. Thus, NSGA-II's selection process just described gives preference to diverse individuals.

As explained in Section 2.5.3.1, NSGA-II applies FNDS to the combined offspring and parent populations at its current evolutionary cycle. This is followed by the application of its selection process to the combined populations to form the next generation parent population. The preference of the selection process to diverse individuals in the combined populations can enable NSGA-II to produce a diverse next-generation population.

## 2.6 Estimation Distribution Algorithm

The *Estimation Distribution Algorithm* (EDA) is a class of evolutionary heuristics which, instead of using evolutionary operators, makes use of sampling and estimation of *Probability Density Functions* (PDFs) to create its next-generation population [119]. It has

the ability to detect and preserve good quality building blocks of chromosomes [183], an ability that could be important in some applications [119].

Before discussing a particular member of EDA, let us describe Figure 2.17. Suppose genes in genotypes, which correspond to individuals in a given population, are task IDs. Each coloured rectangle in this figure signifies the percentage of the genotypes, which have task IDs in the vertical axis at the gene index in the horizontal axis, over all genotypes in the population. The matrix of percentage depicted in the figure is referred to as the probability matrix.

Figure 2.18 presents a particular EDA algorithm: beginning at its first evolutionary cycle $i = 0$, this algorithm creates a probability matrix $P_i$ with equal entries. This probability matrix is then sampled to form a set of genotypes $G_i$, sized $N$. Genotypes in $G_i$ are then selected to form another set $Gsub_i$ of genotypes, sized $\lceil \rho N \rceil$, where $\lceil \bullet \rceil$ is a round-up operator and $\rho$ is a constant where, $0 < \rho \leq 1$. Their selection is based on the fitness of their corresponding phenotypes. A new probability matrix $P_{i+1}$ is then estimated from $Gsub_i$. The cycle of sampling, selection and estimation is repeated until a stopping condition is met.

### 2.6.1 Characteristics of EDA

In Figure 2.17, consideration is given to the visually detected high (of blue hue) probability block of genes indexed 15 to 17 with task IDs 5, 15 and 17 respectively. During the sampling of the probability matrix depicted in the figure, task IDs that correspond to these high probability blocks are more likely to appear at gene indices of the offspring genotypes that correspond to the block, thereby preserving this block [183].

In EDA, if a prospective high quality (e.g., fitness value) genotype has less probability of persisting to the next generation, it is less likely to be in the next generation. Note that, based on elementary statistics, the probability of a genotype in $G_i$ (defined above), for some $i$, to be in the next generation ($i \leftarrow i + 1$) after sampling $P_i$ (defined above) is

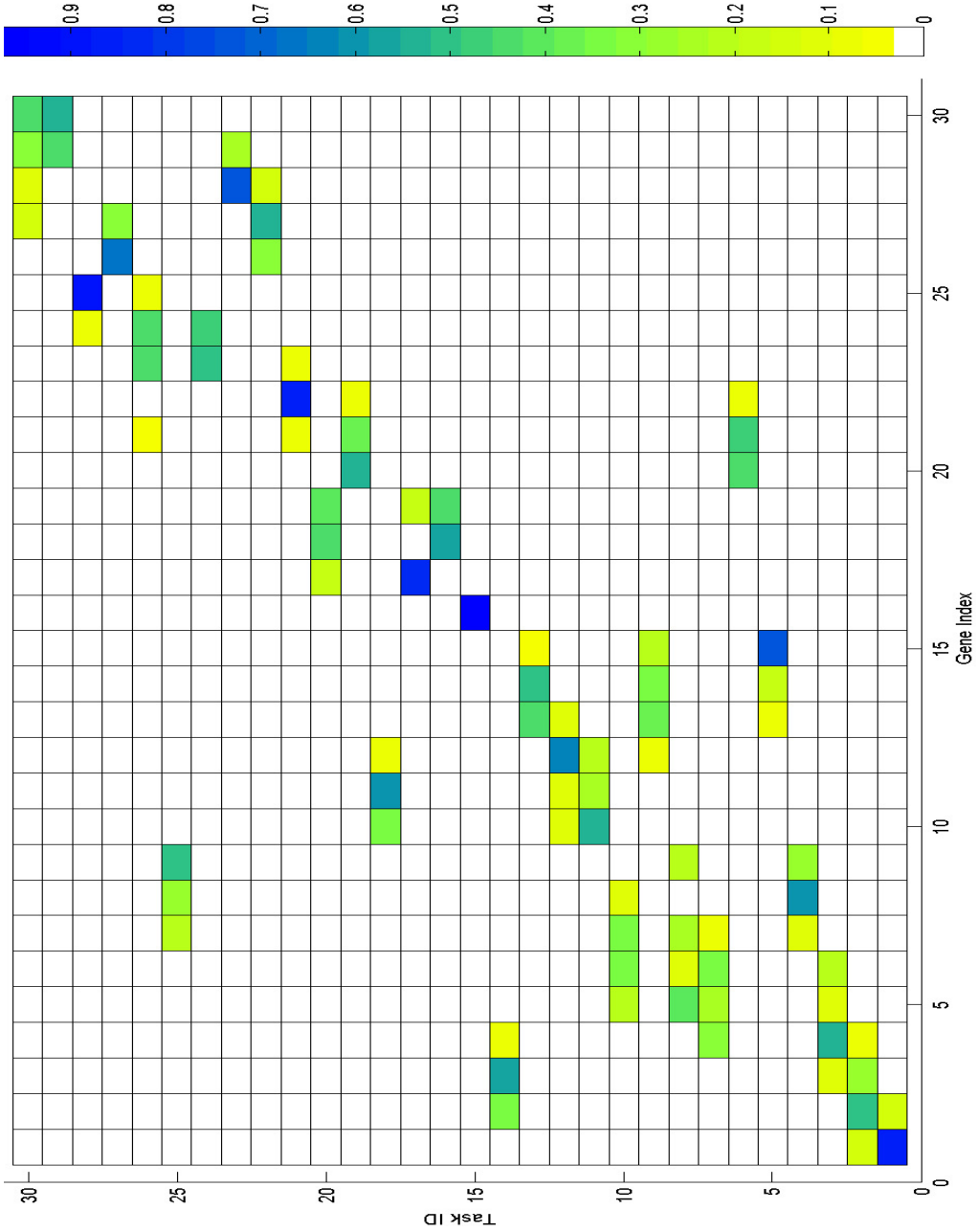



Figure 2.17: Sample task ID distribution

**Procedure** EDA
**Begin**
(1) Set entries of the probability matrix $P_i$, $i = 0$, to equal probability
**While** stopping condition not meet
(2) Create $N$ genotypes by sampling $P_i$ to form a set $G_i$ of genotypes
(3) Select $\rho N$ genotypes from $G_i$ using a fitness measure
(4) $i \leftarrow i + 1$
(5) Estimate $P_i$ from the $\rho N$ of genotypes
**End**
**End**

Figure 2.18: Estimation of Distribution Algorithm

derivable from $P_i$. The EDA drawback was remedied in [178]. Another drawback of EDA is that after several evolutionary cycles (large $i$), the diversity of genotypes in $G_i$ will be lost, a drawback remedied in [130].

### 2.6.2 EDA in Scheduling

In [65, 209], EDA was applied to solve an RCPS problem in a static environment with tasks on various execution modes. The authors utilised a probability matrix,

$$P_i = \left[\lambda^i_{j,k}\right], \tag{2.16}$$

where $j$ is the task index, $k$ is the gene index of genotype used in EDA, and $i$ is the evolutionary cycle count which is equivalent to that in Figure 2.18. Before the start of the first evolutionary cycle ($i = 0$) of the applied EDA, all entries of $P_0$ were set to $1/N$ where, $N$ is the number of tasks in the environment and also the genotype length. This implies that all of the tasks have equal probability of being placed into any gene location in any genotype formed during the sampling of $P_0$ at the first evolutionary cycle.

A copy $C_i$ of $P_i$ was used to generate a genotype as follows: the $k^{th}$ column of $C_i$, i.e. $\mathbf{C}^k_i \equiv \left\{\lambda^i_{j,k} \mid 1 \leq j \leq N\right\}$ is sampled to obtain a task that can be assigned to the gene indexed $k$ of the genotype. However, it could happen that not all tasks with non-zero corresponding probability in $\mathbf{C}^k_i$ are eligible for the assignment due to the task precedence

constraint in the RCPS problem. To remedy this, let $E_k^i$ be a set of indices of eligible tasks with non-zero corresponding probability in $\mathbf{C}_i^k$. The corresponding probabilities of these eligible tasks are provisionally revised to,

$$\widehat{\lambda}_{j,k}^i = \frac{\lambda_{j,k}^i}{\sum\limits_{j \in E_k^i} \lambda_{j,k}^i}, \tag{2.17}$$

where $j \in E_k^i$. The provisional probability vector $\{\widehat{\lambda}_{j,k}^i \mid j \in E_k^i\}$ is then sampled to obtain a task that will be assigned to the gene indexed $k$. After this assignment, all entries of row $j$ of $C_i$ are set to zero to avoid reassigning the obtained task thereafter. Further, each column of $C_i$ is renormalised. The steps just described enable the assignment of a task to one gene only. They are repeated for each gene, from first ($k = 1$) to last ($k = N$) gene consecutively, thereby creating one genotype in the next generation genotype set $G_{i+1}$ (step 2 of Figure 2.18).

The steps described in the last paragraph are repeated, but starting with a fresh copy $C_i$ of $P_i$ for every production of the other genotypes in $G_{i+1}$. The remaining steps in Figure 2.18 are executed to complete one cycle.

## 2.7 Measure of Risk

As shown in Section 1.2, some environmental changes can cause schedules to become infeasible. Consider, for example, the on-going tasks in a given single-mode (defined in Section 2.3) RCPS schedule still being executed even after their expected completion times due to some environmental changes. Suppose further that the items of a certain resource type they utilised are unavailable to their respective immediate succeeding tasks scheduled to commence immediately after their expected completion times. Thus, if no other items of similar resource type are available, the immediate succeeding tasks could no longer be executed immediately after the expected completion times. Hence, the schedule can no longer be fully implemented and needs to be revised. This revision could entail

penalty costs, e.g. a contract infringement penalty with material and labour suppliers. Thus, it is sensible to reduce the risk of failing to fully implement a schedule.

In the scheduling literature, risk is measured in various ways. In some financial applications, risk is measured as a monetary value lost after a given time bound [167]. In a fleet scheduling application, risk is the probability that a fleet would be unable to accommodate the load or passengers [214]. A measure of risk was axiomatically defined in [16]. Risk in a RCPS problem is measured as the probability that a given schedule will violate the resource constraint when the duration of tasks in this schedule is varied, given that task starting times are fixed [41]. The authors in this work defined this probability *Pf* through MCS:

$$Pf = \frac{1}{N_s} \sum_{j=1}^{N_s} x_j \tag{2.18}$$

where $j$ is the simulation index; $N_s$ is the number of simulations; and $x_j$ represents whether a schedule violates resource constraints or not and is determined as follows:

$$x_j = \begin{cases} 1 & \text{if } Pl \text{ violates the constraints} \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

where $Pl$ is the plan.

## 2.8 Response Surface Methodology (RSM)

*Response Surface Methodology* (RSM) is a class of techniques for developing adequate empirical models of phenomena (e.g. McBAR solving problems from $\mathcal{P}^2$) through statistical means [141]. Before proceeding to explore RSM, let us first introduce some definitions. *Experiment* is defined as a procedure undertaken to determine quantity/quantities. For example, the procedure undertaken to determine the performance of an algorithm is an experiment. The independent variables in an experiment are called *factors*, e.g. task duration in an RCPS problem. The value or type of a factor is called *level*, e.g. crossover rate of 0.7 of an EA technique. For equally-spaced levels, the absolute difference between any

pair of adjacent levels is called *resolution*. The absolute difference between the minimum and the maximum levels is called *range*. *Treatment* is a vector (ordered set) of levels, one for each factor. For example,

$$\mathbf{p} = \langle \chi, \mu, \sigma \rangle = \langle 0.8, 0.05, 0.5 \rangle \,, \tag{2.20}$$

where the factors $\chi$, $\mu$ and $\sigma$ are crossover, mutation, and selection rates of an EA technique respectively. *Experiment instance* is an experiment endowed with a treatment, i.e. an experiment whose parameters and/or categories are set to their respective values and/or types found in and comprising a treatment. A sample experiment instance is the application of McBAR to determine the solutions to a problem from $\mathcal{P}^2$ (defined in Section 1.2.5) where McBAR utilises the EA parametric values in Equation 2.20. *Observation* is a particular run/execution of an experiment instance, e.g. a single run of McBAR in the last example. The outcome of an observation is called *data*. For example, data can be a hypervolume of the NDS (elaborated in Section 2.5) of solutions determined by McBAR in the last example. In view of this, data is considered as the *response* of a system to a treatment.

The independent variables in any RSM model correspond to factors and are also called by the same name. Consider a sample model expressed as,

$$\hat{y} = \hat{\alpha} + \sum_{i=1}^{Nf} \hat{\beta}_i x_i + \sum_{i=1}^{Nf-1} \sum_{j>i}^{Nf} \hat{\gamma}_{i,j} x_i x_j + \sum_{i=1}^{Nf} \hat{\omega}_i x_i^2, \tag{2.21}$$

where $Nf$ is the number of factors assumed as being greater than two from here onwards; $\hat{y}$ is the estimated response to a treatment composed of factors $x_i$; and the Greek letters are estimated constant coefficients. Note that the models in RSM are only estimates of the actual characteristics of phenomena [144]. This explains the use of estimates in the equation. Let us now relate these models to some characteristics of phenomena. In an experiment, the *main effect* is present when the system's response due to one of the factors is not influenced by another factor. The factor that causes the main effect is called

the *main factor*. For example, consider a model of McBAR's performance for solving a problem from $\mathcal{P}^2$. If the rate of change of this performance with respect to the change in the number of tasks is not influenced by the level of anything in the environment where the problem is set other than the change, then the change is a main factor. A *linear term* in a polynomial model only contains, aside from a constant coefficient, a single unity-exponent factor, e.g. the term $\hat{\beta}_1 x_1$ in a summation-expanded Equation 2.21. This factor corresponds to a main factor. An *Interaction effect* is present in an experiment when the influence of a factor on experimental responses is dependent on the levels of other factors. An *interaction term* in a polynomial model only contains, aside from a constant coefficient, a product of two unity-exponent factors, e.g. the term $\hat{\gamma}_{1,2} x_1 x_2$ in a summation-expanded Equation 2.21. The interaction term corresponds to the interaction effect. A *quadratic term* in a polynomial model only contains, aside from a constant coefficient, a single exponent-two factor, e.g. $\hat{\omega}_1 x_1^2$ in a summation-expanded Equation 2.21. *Aliasing* exists when it is impossible to determine which factors make a significant system response. It has to be avoided [165].

RSM can be used to model computational characteristics of the algorithms (e.g. EA) which possess stochasticity and yield approximate outcomes. It is relevant for the development of the models for such algorithms due to the following reasons: each of these algorithms could produce different data on different observations of an identical experiment instance; and its stochasticity and approximation could diminish the confidence on the models created from its outputs [165, 166].

### 2.8.1 Design of Experiment (DOE)

In RSM, empirical models are built using data observed from experiments endowed with the designed treatments. The *Design of Experiment* (DOE) is a class of techniques for designing treatments. Note that some applications of DOE are not used for developing empirical models [141].

Let us now explore various techniques in DOE. A *Full Factorial Design* (FFD) is a

type of DOE that utilises all points (others are repeatedly used) in a bounded discretised parametric and/or categorical space as treatments [141]. Suppose, for example, we want to model the performance of McBAR for solving a problem from $\mathcal{P}^2$ with changes only in task duration. Further, the amount of task duration change is either one or two time units only. Thus, for 30 tasks, FFD forms at least $2^{30} = 1,073,741,824$ unique treatments (combinations) of task duration changes. The number of experiments endowed with these treatments could be impractical to conduct. When a design judiciously chooses a proper subset of these treatments, it is called a *Fractional Factorial Design* (FrFD) [141]. The popular FrFDs applied in the EA domain are the *Central Composite Design* (CCD) [144] and the *Box-Behnken Design* (BBD) [33,67]. They are suitable for developing models with interaction terms. In these models, *aliasing* due to the terms of degrees (equal to the sum of all factor exponents) of two or less can safely be ignored but not otherwise [165]. BBD is more commonly applied than CCD to design treatments for RSM that builds models with required quadratic terms. A popular type of DOE called the Taguchi method is generally harnessed to analyse systems that do not have significant interaction effects [169]. However, this method is less effective than RSM as a tool to optimise a system which has a large number of factors [216].

An important use of an RSM-built model is for predicting the outcomes of experiments endowed with given treatments. A set of DOE-designed treatments is described as rotatable if its elements, when utilised in RSM model building, yield a model which has equal prediction variance at any two of its elements that are equidistant to the centre of all of its elements. A type of CCD called a *circumscribed CCD* yields a rotatable set of treatments but is generally not suitable for designing with integral-valued factors. On the other hand, a type of CCD called *center-faced CCD* does not yield a rotatable set of treatments but is suitable for designing with integral-valued factors. BBD yields a nearly rotatable set of treatments and is suitable for designing experiments with integral three-levelled factors [141]. Each factor of the models considered for the thesis has nearly three levels. Hence, we choose BBD to design experiments for creating these models.

### 2.8.2 Model Significance and Adequacy

After designing treatments through DOE, the data gathered from executing experiments endowed with the designed treatments are utilised to determine the coefficients of a model through statistical regression. Next, tests must be performed to measure the significance and verify the adequacy of the coefficient-determined model. Let us first discuss the significance tests. One of these tests is called an *Analysis of Variance* (ANOVA). In the ANOVA of a polynomial model, an F-test is undertaken under the null hypothesis that the model has an all zero-valued coefficient, except the constant term in this model. A p-value obtained from the F-test less than 0.05 is popularly accepted in the literature to indicate that the null hypothesis must be rejected. This rejection implies that at least one term does not have a zero coefficient and hence this term is relevant to the model. Thus, this model is considered as significant. Any RSM-built model found as being significant is regarded to fit the data used to create it. Note that there are other ANOVA results on which model's significance are based, e.g. goodness of fit. For ANOVA results to be valid, the residues (model predictions minus observed responses from a system being modelled at similar treatments) from the model must be normally distributed with a zero mean. This rule is met if, among many plots, the normal plot of residuals is approximately or exactly linear [141].

It is also necessary to verify the adequacy of a model for approximating the system being modelled. The following are some model adequacy criteria:

1. The linear plot of the model residues against observation numbers has no trend.
2. The linear plot of the model residues against the model predicted values is random.
3. The linear plot of the model predicted values against the data used to build the model is nearly or exactly a $45^{o}$ line.
4. Most of the data are located between the $\pm 2$ levels in the *Difference in Fits* (DFFITS) plot.

5. Most of the data are located below the level,

$$\frac{4}{Ndata - Nvars - 1} \tag{2.22}$$

in the *Cooks distance* plot, where *Ndata* is the number of data and *Nvars* is the number of factors in the model.

6. Most of the data are located inside $\pm 3$ levels in the residue plot.

7. Most of the data are located below $\frac{2\text{rank}(X)}{Ndata}$ in the *leverage* plot, where $X$ is the data matrix and "rank" extracts the rank of its matrix argument.

The data located by some magnitudes outside the bounds in Items 4 to 7 are called *outliers*. Note that some additional adequacy criteria are found in the literature.

The fitness (significance) of a model could be improved by applying the *Box-Cox transformation* to the data used to determine the model coefficients and then using the transformed data to determine these coefficients again. This transformation is useful when the data range is fairly large. It is expressed in,

$$BC(\rho) = \frac{1}{\lambda}\left[(\rho + \zeta)^{\lambda} - 1\right], \tag{2.23}$$

where $\rho$ is the data; $\zeta$ is a constant to make the sum $\rho + \zeta$ always positive; and $\lambda$ is another constant that could reduce the data's non-normal distribution. The model fitness could also be improved by judiciously eliminating some outliers from the data then reapplying a regression on the remaining data to determine the new model coefficients [141].

### 2.8.3 Screening of Factors

Let us now explain a methodology applied to enhance model fitness, referred to as screening of factors which is a process of eliminating insignificant experimental factors from being considered either for the design of treatments or as the components of an empirical

model. This process could be performed through some approaches, e.g. by preliminary experimentations without the use of a model or by analysis of a model with known coefficients. The first approach is called a *Model-Free approach* and is applied to possibly reduce the number of factors to be considered in a final treatment design (say, through CCD). Note that the number of treatments produced by a DOE is generally an increasing function of the number of factors. Thus, the Model-Free approach could reduce the number of required treatments to develop an empirical model. One way to identify insignificant factors under this approach is by designing treatments with two-level factors. A T-test is then applied on data observed in executing experiments endowed with the treatments. Factors with p-values determined in this t-test greater than 0.05 are usually considered as insignificant [19, 216]. In the second approach, called a *Model-Based approach*, model terms with p-values determined on t-test greater than 0.05 are considered as insignificant and could be eliminated. Any model term elimination must be pursued judiciously, using the practitioner's knowledge and experiences [141].

### 2.8.4 Model Building

All the topics discussed so far in this section are useful to explain the model building process through RSM. This process involves the search for the highest ordered polynomial model that fits (defined in Section 2.8.2) a given data. A particular RSM model building procedure is as follows:

1. The Model-Free approach is performed to possibly reduce the number of factors to be considered in a DOE.
2. Based on the factors remaining after their possible reduction, treatments are designed through the DOE.
3. Experiments endowed with the designed treatments are executed to obtain data.
4. Fit the data to a first order polynomial model as follows:

(a) Apply regression analysis to determine the model coefficients using the data.

(b) Check the fitness and adequacy of the resulting model as related in Section 2.8.2.

(c) Improve the fitness of the model as explained in Section 2.8.2.

(d) Eliminate the insignificant model terms through the Model-Based approach described in Section 2.8.3.

5. Repeat step 4 with models of increasing polynomial order until either the fitness (based on p-value in F-test explained in Section 2.8.2) of the current un-aliased model decreases or saturates from the last un-aliased model. In either of the events, the last un-aliased model is considered as the best to describe the system being modelled and referred to as the final model [141].

### 2.8.5 Model Prediction

It is important to determine the accuracy of the recently-defined final model for predicting system responses to treatments. One way to determine this prediction accuracy is to randomly generate treatments (not used in step 2 of Section 2.8.4) that have factor levels which lie within the factor ranges established by a practitioner. The experiments endowed with these treatments (i.e. experiment instance) are then executed. If the observed data/responses from these experiment instances are mostly within 95% (the exact number depends on the practitioner) of a *prediction interval* then the final model is considered as accurate. The prediction interval is a range of values whereby the observed responses gathered from executing the experiment instances have 0.95 probability of being within this range.

Figure 2.19 depicts predictions from a model where: the middle curve is the sorted data predicted by the final model at randomly-generated treatments; the horizontal axis indicates the labels of the treatments; the lowest and highest curves bound the 95% prediction interval; and the dots are the observed responses from executing the experiments

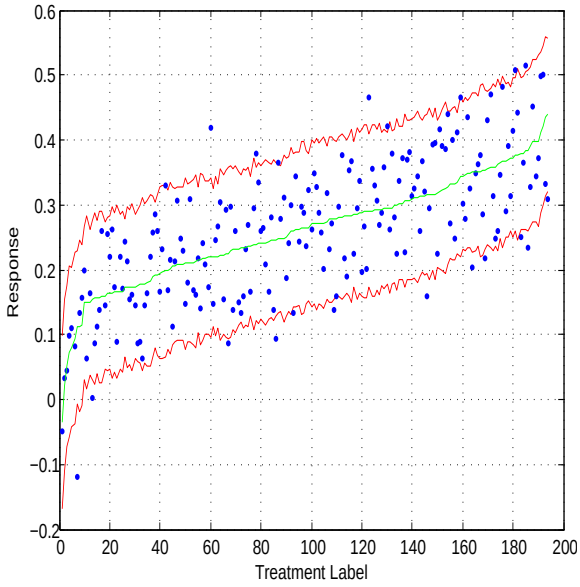



Figure 2.19: Sample confidence interval

endowed with the treatments. The figure shows that most of the observed responses are within the prediction interval. This outcome confirms that the final model is accurate in predicting responses.

### 2.8.6 Optimal Treatment

After building a final model through RSM and proving it to be accurate in prediction as just explained, it can now be used to investigate the system being modelled. One of its usages is for determining treatments at which the system is optimal [141]. For example, the parametric values (e.g. 0.7 crossover and 0.03 mutation rates) of some evolutionary operators that yield optimal McBAR's performance can be determined through RSM. The *Method of Steepest Ascent* (MSA) is a technique that applies RSM and is popularly utilised to determine the response-optimising treatments of some systems. Its modified version which is suitable for system optimisation that involves several important (highly influential on system's response) system factors have the following algorithm [31]:

1. Viewing treatments as points in factor space, design a set of treatments located around and at a centre. This centre is referred to as base point **Bp** and supplied by the RSM practitioner.

2. Execute the experiments endowed with the designed treatments to obtain responses from a system being optimised.

3. Fit a *linear* model to the responses following step 4 in Section 2.8.4.

Let this model be expressed as,

$$\hat{y} = \hat{a} + \sum_{i=1}^{Nw} \hat{b}_i w_i + \sum_{j=1}^{Nx} \hat{c}_j x_j, \tag{2.24}$$

where $\hat{y}$ is the estimate of system response; $\hat{a}$, $\hat{b}_i$ and $\hat{c}_j$ are constants determined through the regression analysis applied in step 4a in Section 2.8.4; $w_i$ and $x_j$ are factors that form the sets $W = \{w_i\}$ and $X = \{x_j\}$ respectively; $W \cap X = \emptyset$; $w_l \in W$ and $x_k \in X$ are the only important factors; $Nw = |W|$; $Nx = |X|$; and $Nf = Nw + Nx$ is the number of factors.

4. Beginning with index $m = 1$, perform the following for a chosen number of cycles:

   (a) Beginning with index $n = 1$, perform the following inner cycle:

      i. Displace the base point to become the treatment,

$$\mathbf{T}_{m,n} = \mathbf{Bp} + m\mu\mathbf{b} + n\lambda\mathbf{c}, \tag{2.25}$$

      where $\mathbf{b} = \sum_{i=1}^{Nw} \hat{b}_i \widehat{\mathbf{w}}_i$; $\widehat{\mathbf{w}}_i$ is a unit vector that corresponds to factor $w_i$ and has coefficient $\hat{b}_i$ in Equation 2.24; the set of $\widehat{\mathbf{w}}_i$s is orthonormal; $\mu = B_l/\hat{b}_l$; $B_l$ is a practitioner-selected constant; $\mathbf{c} = \sum_{i=1}^{Nx} \hat{c}_i \widehat{\mathbf{x}}_i$; $\widehat{\mathbf{x}}_i$ is a unit vector that corresponds to factor $x_i$ which has coefficient $\hat{c}_i$ in Equation 2.24; the set of $\widehat{\mathbf{x}}_i$s is orthonormal; $\lambda = C_k/\hat{c}_k$; and $C_k$ is another practitioner-selected constant.

   Based on Equation 2.25, if $n$ is held fixed, the base point $\mathbf{Bp}$ will be displaced by $\mu\mathbf{b}$ at every increment of $m$. This displacement has a magnitude of $B_l$ along the unit vector $\widehat{\mathbf{w}}_l$ which corresponds to a system-important factor $w_l$.

Further, if $m$ is held fixed, the base point **Bp** will be displaced by $\lambda\mathbf{c}$ at every increment of $n$. This displacement has a magnitude of $C_k$ along the unit vector $\widehat{\mathbf{x}}_k$ which corresponds to another system-important factor $x_k$.

ii. Execute the experiment endowed with treatment $\mathbf{T}_{m,n}$.

iii. Increment $n \rightarrow n+1$.

This inner cycle is to be repeated for a chosen number whereby a peak in system response to treatments $\mathbf{T}_{m,n}$ must be observed. The base point **Bp** should be selected to obtain a likely progressive increase in response to treatments $\mathbf{T}_{m,n}$ at the first few inner cycles; thus, the name MSA.

(b) Increment $m \rightarrow m+1$.

5. The treatment $\mathbf{T}_{m,n}$ that yields the maximum response can be considered as the system optimising treatment being sought or as a new base point for further analysis.

## 2.9 Lagrange Optimisation

The EA is versatile in searching solutions to many optimisation problems and belongs to a class of stochastic algorithms. By its stochasticity, however, it does not consistently yield a similar solution every time it solves the same optimisation problem [166]. Lagrange Optimisation, on the other hand, is one of the popular optimisation methods that overcomes this limitation for being a deterministic method [91]. Let us now discuss its basic features through an example. Consider a problem of determining a vector $\mathbf{x} = \{x_1, x_2\}$ which yields an optimal value of a function $f(\mathbf{x})$ – whose selected contours are depicted as thin curves in Figure 2.20 – under the constraint,

$$c\,(\mathbf{x}) = b \tag{2.26}$$

where $b$ is a constant and $c$ is a certain function. The values of $\mathbf{x}$ that satisfy the constraint are points that comprise the thick curve in the figure. Any thin curve is composed of

points $\mathbf{y}$ whereby $f(\mathbf{y}) = lvl$ with $lvl$ being the number at this thin curve. As any possible solution to the problem must satisfy the constraint, then it must either be the intersection (e.g. point A in the figure) of a contour of $f(\mathbf{x})$ and the thick curve or the point (e.g. point T in the figure), called a touch point, at which a contour of $f(\mathbf{x})$ is tangential to that of the thick curve.

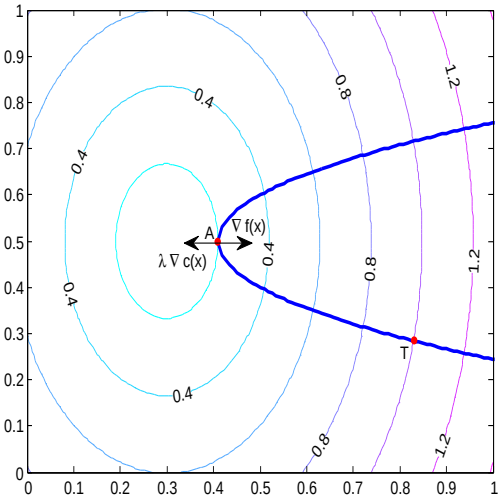



Figure 2.20: Contour graphs

Now, starting from the lowest right point of the thick curve up to its leftmost point $f(\mathbf{x})$ changes to several values, such as 1.2, 0.8, and 0.4, indicated in the figure. It is only at a contour of $f(\mathbf{x})$ which touches the thick curve when $f(\mathbf{x})$ remains constant (zero slope) over an infinitesimal range of points centred on the touch point and being parts of the thick curve. Thus, the touch point corresponds to an optimum (zero slope) of $f(\mathbf{x})$ that satisfies the constraint (found at the thick curve) and therefore the solution to this particular optimisation problem.

This touching happens when the tangent vector (represented as the right-directed arrow in the figure) of a contour of $f(\mathbf{x})$ is in anti-parallel to that (represented as left-directed arrow in the figure) of the constraint. Thus,

$$\nabla f(\mathbf{x}) = -\lambda \nabla c\,(\mathbf{x}) \tag{2.27}$$

where the constant $\lambda$ will make the magnitudes of the two tangential vectors equal and

is called the *Lagragian Multiplier*. This constant must not be equal to zero or complex-valued. Otherwise, the two vectors may not be anti-parallel and yet could satisfy Equation 2.27 [91].

Let the touch point be denoted as $\mathbf{x}^* = \{x_1^*, x_2^*, \ldots, x_M^*\}$ where $M$ is the vector dimension. For any continuous $f(\mathbf{x})$ and $c(\mathbf{x})$, and constraint as in Equation 2.26, Lagragian Optimisation could be performed as follows:

(1) Express $\mathbf{x}^*$ in terms of $\lambda^*$ using Equation 2.27 with $\mathbf{x} \leftarrow \mathbf{x}^*$ to obtain,

$$\mathbf{x}^* = \mathbf{h}(\lambda^*) \tag{2.28}$$

where $\mathbf{h}(\lambda^*)$ is a vector function.

(2) Substitute $\mathbf{x}^*$ from Equation 2.28 to the constraint in Equation 2.26 to obtain,

$$c(\mathbf{h}(\lambda^*)) = b \tag{2.29}$$

(3) Determine $\lambda^*$ from Equation 2.29.

(4) Substitute $\lambda^*$ to Equation 2.28 to obtain $\mathbf{x}^*$.

Note that there could be several values of $\lambda^*$ that can satisfy Equation 2.29. Let $\lambda^{\#}$ be one of these values and which yields $\mathbf{x}^{\#}$ when substituted to Equation 2.28. If $\mathbf{x}^{\#}$ satisfies both Equations 2.26 and 2.27, and $\lambda^{\#}$ is real and non-zero then $\mathbf{x}^{\#}$ is a possible solution to the optimisation problem [91]. If there are several possible solutions, the possible solution $\mathbf{p}$ that yields the maximum/minimum value of $f(\mathbf{p})$ is the solution to the maximisation/minimisation (optimisation) problem.

## 2.10 Measures of Intelligence

The intelligence of a machine, such as a computer, can be measured in several ways. One machine intelligence measuring test is the Turing test where a human and a machine both provide answers to a series of similar questions in a teletype conversation. If the quality of answers given by the machine cannot be distinguished by human judges from that of the human then the machine is considered as intelligent. Although practically impossible but not illogical, the machine can be built with a database containing an infinite number of question-answer pairs. This machine is therefore capable of answering any question and will consistently be judged as intelligent, based on the Turing test, despite lacking other facets of human intelligence such as creativity [121]. A popular measurement of human intelligence is the psychometric test. However, a computer programmed to excel only in psychometric tests will not be able to pass other tests, such as an exam on Riemannian Manifolds [120] or Quantum Gravity [171]. Other machine intelligence measuring tests are Compression, Linguistic Complexity, Multiple Cognitive Abilities (which was used in IBM's Joshua Blue project), and Smith's, to name a few. The Turing and psychometric tests are only suited to measure specific facets of human-quality intelligence. However, the *Universal Intelligence* (UI) test is a general measure of human-quality intelligence [121]. It is based on the principle of Occams Razor and *Reinforcement Learning* (RL).

### 2.10.1 Reinforcement Learning (RL)

Let us briefly introduce Reinforcement Learning. Consider an ancient man in a forest. When feeling hungry, for instance, he either performs a series of bodily actions to go up a hill and then search for food there, or performs a different series of bodily actions to go down a valley and then search for food there. Suppose that on several of his hunger-catalysed experiences, he was able to find food on the hill more frequently than in the valley. The next time he feels hungry after these experiences, he will likely choose the series of actions to go to the hill and search for food there since this series is more likely to

be rewarding (i.e. satisfies his hunger) than the other series of actions. The fundamental principle of Reinforcement Learning is that, actions learned by an agent (a learning entity, such as the ancient man) as having greater likelihood of being rewarded are likely to be executed by this agent. A significant merit of an agent following this principle is the ability to learn autonomously; experience is its only teacher [190].

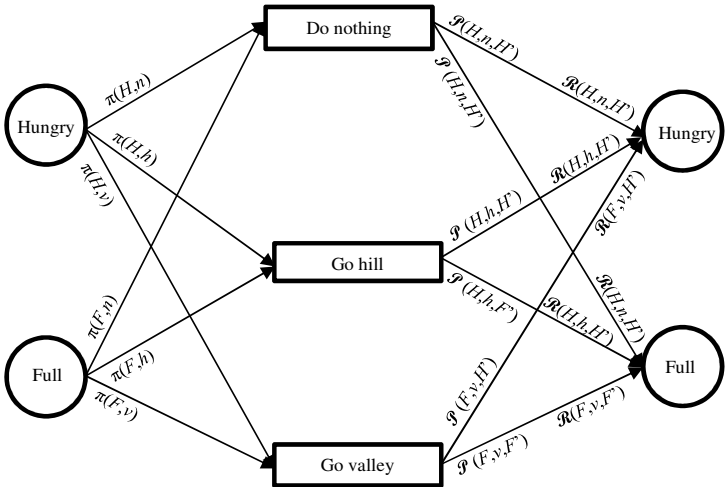



Figure 2.21: Sample Reinforcement Learning system

An RL system is composed of an environment and agents who dynamically interact with this environment and among themselves [121]. For the purpose of introducing RL, only one agent is considered in this subsection. The agent's action can alter the environment, e.g. digging an animal trap. As a consequence of this action, the environment could give this agent an observation (e.g. an animal is trapped) and reward (e.g. food) or punishment (e.g. starvation). Let the reward value quantify reward as a positive value, including zero, while punishment is a negative value.

Consider a sample RL system, as illustrated in Figure 2.21, for the scenario above involving an ancient man. Labels in circles and rectangles in this figure correspond, respectively, to states and series of actions of this ancient man. Let the types of series of actions be in the set $A = \{h, v, n\}$ where $h$, $v$, and $n$ denote, respectively, the series of actions to go to the hill then search for food, go to the valley then search for food, and do nothing. Further, let the types of state he can experience be in the set $S = \{H, F\}$ where $H$ and $F$ denote, respectively, hungry and full. The states on the left of the figure

are the possible current states in $S$ that the ancient man can experience while those on the right are the possible next states also in $S$, distinguished from the current states by an apostrophe.

Now, after the ancient man had gone through several experiences of being in a state $s \in S$ and then taking series of actions $a \in A$ the probability $\mathcal{P}(s, a, s')$ that he ends in next state $s' \in S$ with an average reward value $\mathcal{R}(s, a, s')$ received for executing $a$ can be determined. Both the probability and average reward are indicated in the figure for different (current state)-(series of action)-(next state) triples. For instance, when hungry $H \in S$, the reward (degree of stomach fullness $F \in S$) for going to the valley might be 0.7, i.e. $R(h, v, f) = 0.7$. For a given current state $s$ and series of actions $a$, the sum of probabilities $\mathcal{P}(s, a, s')$ over all next states $s'$ is unity.

The probability of being in a state $s$ and then taking a series of actions $a$ is called a policy, denoted as $\pi(s, a)$. After determining the probabilities $\mathcal{P}(s, a, s')$ and average reward values $\mathcal{R}(s, a, s')$ the policy $\pi^*(s, a)$ – referred to as the optimal policy – that yields optimal reward value can be determined. Once known, the optimal policy can be utilised to determine the series of actions (e.g. go to a hill and then search for food), given a state (e.g. being hungry), that yields the greatest reward (e.g. bountiful food) [190].

### 2.10.2 Universal Intelligence (UI)

The concept behind RL is utilised in defining a measure of intelligence, referred to as *Universal Intelligence* (UI). The UI of an agent/software $\alpha$ over a set of tasks $Tsk$ is expressed as [121],

$$\Psi(\alpha) = \sum_{\tau \in Tsk} 2^{-K(\tau)} R_\tau^\alpha \tag{2.30}$$

where $K(\tau)$ is the Kolmogorov complexity which is the length of the program executed by the agent that can accomplish a given task $\tau$ (e.g. playing chess) where the reward value to the agent for this task is $R_\tau^\alpha$ [121].

Accomplishing a task by an agent can manifest a facet of this agent's intelligence. If

the summation in Equation 2.30 is over all types of tasks then this equation accounts for all facets of intelligence. To determine the agent's intelligence, this equation only requires the obtainable lengths of the programs of the agent to accomplish the tasks; and rewards for executing the tasks. Thus, UI can determine the totality of an agent's intelligence. Based on the equation, Kasparov is still more intelligent than the Deep Blue machine even if Deep Blue won over Kasparov in their chess games. This is because, other than playing chess, Kasparov can accomplish almost infinitely many types of rewarding tasks, which Deep Blue is not programmed to accomplish.

If the degree of simplicity of a program is taken as the program length $K(\tau)$, the simplest program (i.e. smallest $K(\tau)$) that can accomplish a task $\tau$ yields the largest value of $2^{-K(\tau)}$. Thus, Equation 2.30 shows that the simpler the programs utilised by an agent to accomplish most tasks, the more intelligent this agent is, i.e. the simpler the better; an idea based on Occams Razor.

The inclusion of reward value $R_\tau^\alpha$ to Equation 2.30 is inspired by the RL principle [121]. It is a legitimate component of this equation since, accomplishing more rewarding types of tasks, e.g. winning a Nobel Prize in Physics, can be regarded as requiring more intelligence. Kolmogorov complexity, such as in Equation 2.30, is non-computable [126] thus constituting a drawback to the UI measure. A different computable complexity measure might be appropriate [121].

# Literature Review

## Chapter 3

This chapter reviews current work in areas related to multi-objective dynamic RCPS problems with time-varying number of tasks (i.e. problems from $\mathcal{M}$), areas that were investigated in Chapter 2 such as, Genetic Programming, RCPS, schedule optimisation in dynamic environments and multi-objective schedule optimisation. This is followed by the review of current techniques utilised to demonstrate in the thesis the relevance of McBAR, techniques also explained in Chapter 2 such as, EDA, robust proactive and reactive-predictive approaches to scheduling, RSM and Reinforcement Learning. This chapter concludes with the provision of information on a current method in Artificial Intelligence. This method, referred to as Multi-Agent System (MAS) is applied to scheduling but not at all for McBAR.

## 3.1 Genetic Programming (GP)

Let us explore the various applications of GP. In priority scheduling, the priorities of executing certain components of scheduling system are to be determined aside from the schedule itself. The prioritisation scheme is referred to as dispatching rule, scheduling rule, or heuristic. It is usually represented by a priority function which maps scheduling system components to their priorities. It is produced by GP applied in various applications [20, 43, 102] to create schedules for dynamic job-shop environment with time-varying number of tasks. From a GP-evolved heuristic in [147] the priority function was used as basis for executing a job with current highest priority. The GP applied for the job-shop scheduling problem in [102] utilised addition, subtraction, multiplication and protected

(against zero divisor) division of real numbers as functions and utilised job processing time, job due date, and slack time as terminals, to name a few. It was also applied in [146] to determine dispatching rules for single objective job-shop scheduling problems. GP was applied to solve multi-objective problem on edge detection in image in [234], where the objectives are the minimisation of the depth of the GP-evolved tree and the minimisation of the image component misclassification in applying the tree (executing the program). GP evolved a heuristic for an exam timetabling problem instance in [20], where measurement is made at every evolutionary cycle on the quality of the solution found using the current heuristic. This approach is repeated until a high quality solution is found. In a dynamic semiconductor manufacturing scheduling problem (a type of job-shop scheduling problem) in [154], GP was applied to evolve job assignment rules, each composed of rules extracted from fundamental job properties. It is assisted by EA to allocate assignment rules to manufacturing work centers.

Let us now explore a GP-based algorithm, referred to as Diversified Multi-Objective Cooperative Evolution (DMOCC), applied to solve multi-objective dynamic (with time-varying number of tasks and dispatching rule, to name a few) job-shop scheduling problem in [147]. The problem accounts dispatching rule (DR) and due-date assignment rule (DDAR) of the jobs. The algorithm started with two initial populations, $Pdr$ and $Pddar$, created using ramped-half-and-half technique [115] and composed of DRs and DDARs respectively. Each element $pdr \in Pdr$ was paired to a binary tournament (accounts the quality and distribution of selected individuals) selected element $pddar \in Pddar$ to form a complete scheduling policy. This policy was evaluated at the simulation of the system at which the problem was set to obtain the fitness values (e.g. makespan and tardiness of schedule found through this policy) of $pdr$. This process of fitness evaluation was applied to all elements of $Pdr$. The same pairing and evaluation processes were performed on the elements of $Pddar$, except that the pairings are between elements of $Pddar$ and the binary tournament selected elements of $Pdr$. After these evaluations, an archive which started from empty was updated using the populations $Pdr$ and $Pddar$. Next, the pareto rank and crowding distance (defined in Sections 2.5.3.2 and 2.5.3.3 respectively)

of each element of the archive was computed. If the number of evolutionary cycles was reached the algorithm was terminated, otherwise new *Pdr* and *Pddar* was formed from the elements of the archive using mutation and crossover operators.

Let us focus on non-scheduling GP applications. In the domain of signal processing: Esparcia and Sharman [63] applied GP to evolve the topology of and the transfer function of each node in each Recurrent Neural Networks (RNN). However, the neuronal weights were determined using Simulated Annealing. At each stage of the evolution, the developed RNNs are individually applied to equalize nonlinear telecommunication channels with additive white gaussian noise [45], an application where the network fitness is determined. Howard, et al. [97], utilised GP to evolve detectors in algebraic form and applied these detectors to identify information on ships in Synthetic Aperture Radar (SAR) images. Detection of information on vehicles in infrared images was undertaken in [98] where the performances of evolved detectors are compared between two sets of terminals utilised in a GP evolution. The discrete Fourier transform set was found to evolve detectors with the better performance under the considered images in [98]. GP was also applied to profile tissues using the data from the vibrational spectroscopy of these tissues [80].

Genetic Programming was also applied to develop mathematical models in various applications. Cai, et al. [44], utilised GP to evolve regression model of correlations in considered heat transfer processes, where the fitness function was influenced by the observed and model-predicted quantity of transferred heat. Lew, et al. [124], created surrogate model of a highly complex structural dynamics using not the traditional multinomial model through Response Surface Methodology (explored in Section 2.8) but a model evolved through GP. The learning dynamics of each artificially intelligent player was modelled in [49]. The model, called forecast rule, is being evolved through GP while its associated agent/player is interacting to other agents. An interesting result in the investigation in [49] is that the behaviour of the agents (influenced by each associated model) in a game is closely related to that of human players on same game.

An explicit memory-based approach was applied in [147], where some GP-evolved non-

dominated programs applied to solve past sub-problems (defined in Section 1.2.3) were used to search for high-quality programs to solve for a current sub-problem. However, intuitively, it is extremely challenging to define a meaningful representative of a non-dominated set of programs for implicit memory-based approach (defined in Section 3.3.1). Consider, for example, two GP trees with the same structures but with some different nodes. It is extremely challenging to define the average of functions, say + and ×, at two nodes of same locations in two GP trees whereby this average becomes a meaningful part of a representative (also a tree) of a non-dominated set of programs/trees where the two trees belong. In fact, no literature applied GP that incorporated implicit memory-based approach for solving RCPS problems, such as problems found in [72, 149] to which GP was applied. Considering the relevance of implicit memory-based approach (which could utilise representative of non-dominated set of programs) as explained in Section 3.3.1 GP was not utilised in this thesis.

## 3.2 Resource Constrained Project Scheduling

As noted in Chapter 1, many RCPS problems are NP-hard [123]. In general, they can be modelled as mixed integer programming problems. Therefore, conventional linear programming, such as the simplex method, is not really suitable, and many researchers have developed a number of approaches over recent years. Broadly speaking, these approaches can be classified into two categories: exact and approximate methods. For exact methods, the final solution will be the optimal one for the RCPS. Some typical approaches of this category are branch and bound [56], Lagrangian relaxation [59], and dynamic programming [46]. However, exact methods usually face an issue of execution time with large scale problems (the number of RCPS tasks should not exceed 60, according to [10]). Therefore, approximate methods are preferred, including a variety of heuristic/meta-heuristic techniques, such as priority-based, truncated branch and bound, sampling techniques, local search techniques, tabu search, simulated annealing, scatter search, multi-agent systems and EAs [14, 92, 114, 136].

As elaborated in the previous chapters, RCPS inherently possesses a feature of multi-objectivity. Surprisingly, the literature on RCPS is dominated by work considering only single objectives [186, 206]. There exist a number of possible conflicting objectives of RCPS such as the minimisation of schedule duration and implementation cost, resource balancing and maximisation of schedule robustness. Note that for multi-objective optimisation, finding good (optimal) solutions is not the only consideration; addressing the trade-off between objectives among obtained solutions is another consideration.

A common approach in dealing with multi-objective RCPS (also for machine scheduling [145]) is the weighted-sum [1]. For this approach, objectives are summed according to a predefined vector of weights. However, this approach systematically faces several issues, such as objective scaling, and it is also not easy to address the matter of solution trade-off analysis. Alternatively, Pareto-based approaches are used to obtain a set of trade-off solutions in a single run [25, 206]. Therefore, these approaches seem to be more suitable for solving the multi-objective RCPS problem. One of these approaches is NSGA-II described in Section 2.5.3.1 and is utilised in the thesis.

## 3.3 Dynamic Optimisation Problem (DOP)

Some current literature on DOP had already been explored in Section 2.4. Thus this section only presents current work on memory-based approaches (explained in Section 2.4.2) to solve DOPs, and on techniques for solving scheduling problems with a time-varying number of tasks, two areas which are fundamental to McBAR.

### 3.3.1 Explicit Memory-Based Approach

The explicit memory-based approach defines how the raw information produced by EA for solving problems, and/or the raw information about environments where these problems are set, are stored and retrieved [223]. In [223,224], a combination of competitive learning

and explicit memory-based approaches was applied. The fundamental algorithm in these studies is as follows. First, a probability vector $\mathbf{P}$ is initialised with all of its components set to 0.5 and a memory is set as empty. Now a loop is entered that begins with $\mathbf{P}$ being utilised to generate a population $S$ of genotypes. Then, the memory is updated at the random cycle number of this loop. During an update, if the memory is not full, the pair of the best fit sample $\mathbf{B}$ from $S$ and the current probability vector $\mathbf{P}$ is stored in the memory. Otherwise, a sample $\mathbf{B}_m$ in the memory that is mathematically nearest to $\mathbf{B}$ is considered. If the fitness of $\mathbf{B}$ is better than that of $\mathbf{B}_m$, $\mathbf{B}_m$ and its probability vector pair $\mathbf{P}_m$ are replaced in the memory by $\mathbf{B}$ and $\mathbf{P}$ respectively. After processing the memory, if the environment where the problem is set and considered in the work changes, the sample $\mathbf{B}_f$ with the best fit among all samples in the memory is determined. If the fitness of $\mathbf{B}_f$ is better than that of $\mathbf{B}$ (best fit among genotypes of $S$) then $\mathbf{P}$ is replaced by the probability vector $\mathbf{P}_f$ pair of $\mathbf{B}_f$ from the memory. However, if there is no change in the environment, $\mathbf{P}$ is transformed through a competitive learning towards a linear function of $\mathbf{B}$. The probability vector, either the result of the replacement or the transformation, is mutated to obtain a new current probability vector $\mathbf{P}$. The loop is then repeated until a termination condition is reached.

An explicit memory-based EA approach in [185] was based on the human immune system and applied to solve a dynamic knapsack problem. The snapshot (defined in Section 1.2.3) of the dynamic environment, where the problem is set, was considered as an antigen (molecule foreign to the immune system). And, the solutions to the sub-problem (defined in Section 1.2.4) set in the snapshot were considered as antibodies. For each previous snapshot of the environment, the antigen that corresponds to this snapshot and the set of solutions/antibodies that have good fitness among all solutions to the sub-problem set in this snapshot are stored to a memory. Now, the antigen $C$ that corresponds to the snapshot taken right after the current change of the environment is compared to those in the memory. The set of antibodies paired to the memorised antigen nearest to $C$ has its antibodies cloned and then these antibodies replace the less fit antibodies/solutions to the sub-problem set in the last snapshot. The resulting set

of solutions is then evolved to obtain the solutions to the sub-problem set in the current snapshot.

The basic approach in [228] was to search for a solution to a sub-problem set in the current snapshot of a dynamic environment where this solution has a higher likelihood of still having an appreciable quality when implemented in one or more future snapshots of the environment. The implementation of the solution in the future snapshots can be considered as being an explicit memory use. A solution such as this is referred to as a *Robust Solution Over Time* (RSOT) and the problem of determining it is referred to as a *Robust Optimisation Over Time* (ROOT). In [73], some measures on the changes of a dynamic environment were investigated, namely measures that can be used to decide which approach is suitable to solve a ROOT problem.

The system in [161] continuously learned of the changes in an environment by dynamically updating its knowledge base of pairs of environmental properties and solutions to a problem set in the environment. A pair will be retrieved if it contains environmental properties that match, based on a mathematical measure, those of the current environmental state. The solutions contained in the retrieved pair were then utilised to form an initial population which was evolved through EA to search for optimal solutions to a problem set in the current state of the environment. A system described in [131], closely related to that discussed in [161], was applied to some dynamic resource allocation problems in a command and control environment. This allocation problem considered risk and cost in a project implementation, factors which were lumped into one objective function.

### 3.3.2 Implicit Memory-Based Approach

The implicit memory-based approach defines the way in which the results of processing the raw information produced by EA for solving problems are stored and retrieved [223]. This information may or may not be paired in the memory with the raw or processed information in the environment where the problems are set. In the EA approach of [22] for solving a job-shop scheduling, at every fixed number of evolutionary generation and at

every rescheduling moment (when the environment, to where the problem is set, changes), each task in the best fit individual is categorised to obtain a prioritised list of categories that is stored in a memory. A process is applied to store the list to a memory. At every evolutionary generation, pending tasks are categorised. And, using the priority of categories in a prioritised list in the memory a prioritised list of the tasks is obtained, the list that constitutes an individual/schedule. This step is performed for each memory element. The obtained individuals are inserted to the population being evolved.

Multi-stranded chromosomes (polyploidy) were utilised in [85, 125, 202] to store representatives of solutions which correspond to past environmental changes. This utilisation was shown to be useful in searching solutions, through the Genetic Algorithm, to some dynamic problems. Chromosomes utilised in [52] have a multi-level genetic structure which serves as a long-term memory of representatives of solutions and environmental properties. They facilitate quick adaptation of a function optimiser to environmental changes.

In the scheduling domain, performance of an implicit style is better than that of an explicit style because, in a dynamic scenario, the schedule produced by an explicit style will swiftly become irrelevant due to variation in the priority and precedence order of schedule components [22].

### 3.3.3 Time-Varying Task Number

There have been several reactive-predictive (defined in Section 2.4.2) scheduling approaches applied to revise schedules to cope with the effects of the time-varying number of tasks. For example, in the EA approach to the job-shop scheduling problem in [22], genes which correspond to new/old jobs that arrived/finished were inserted/removed to/from genotypes of phenotypes (schedules). The resulting population of gene-inserted genotypes was then evolved further to search for new high-quality schedules. It is worth noting that despite the genotype alteration, significant improvements in the search convergence of the EA and in the quality of the obtained solutions/schedules were found [27, 28, 127, 164].

In [150], tasks in processors of a distributed computing system arrive randomly and were put on queue. These tasks were then processed in batches, each has a varying number of tasks. This processing followed a schedule created through the Genetic Algorithm to obtain a minimal combined execution time of tasks in every batch. In [15], genes that correspond to new tasks were also inserted into genotypes which were then processed by an EA to obtain a new high-quality schedule for multi-resource scheduling with cumulative constraints. In this EA process, the lateness of the new schedule with respect to a given deadline was minimised and at the same time important properties of the original schedule were preserved. EDA (presented in Section 2.6) was applied to solve a non-RCPS problem whose dimension changed in time [78, 79]. However, the objective functions of the problem were simple. More techniques in revising schedules to cope with the variation of the number of tasks in the environments in which these schedules are set can be found in [58].

## 3.4 Multi-Objective Optimisation (MOO)

As explained in Chapter 1, MOO can be useful for creating schedules or plans. Let us now review prominent approaches in this domain. The MOEA techniques (a type of MOO elaborated in Section 2.5) can be classified into two broad categories: those with and those without elitism. In the elitism approach, a set is utilised to store high-quality solutions determined at previous evolutionary generations. Its subset becomes a part of a population of a current generation. In this manner, the best individuals of some generations can be preserved, thereby helping to obtain solutions close to the Pareto optimal front (defined in Section 2.5) for a given problem. Algorithms such as SPEA2 [238], PDE [2] and NSGA2 [55] are examples of this category. In contrast, the non-elitism approach has no concept of elitism when it selects individuals for a next evolutionary generation from the population at a current generation [240]. Examples of this category include VEGA [179] and NSGA [53].

MOEA techniques can also be classified according to criteria other than elitism: criterion selection bases the selection of next generation individuals on the objective values of the individuals [179]; aggregation selection bases the selection on the weighted sum of the objective values of each individuals [86]; and Pareto selection, the most popular, concentrates the selection on finding a set of promising individuals [227]. Examples of Pareto selection schemes are: Pareto ranking [71], niched Pareto GA [96], non-dominated sorting GA (NSGA) [188], strength Pareto EA [237], Pareto archive evolutionary strategy [112], and NSGA-II [55]. NSGA-II is one of the most popular methods in the MOEA literature and was chosen in this thesis as a part of the solution-searching mechanism of techniques from $\mathfrak{T}$ (defined in Section 1.2.6).

MOO is utilised in Chapter 7 to solve some scheduling problems that have random variables. Let us review some of the literature related to its use to solve this type of problems. In Multiprotocol Label Switching [134], the random information transmission blockages degrade the quality of service (QoS) to telecommunication customers. This QoS was traded-off with the operational cost of a telecommunication company through MOO. In [12], the probability of (random) pipe breakage was utilised to estimate the pipe repair cost of a piping system. The repair cost was traded-off with the expected volume of lost water due to the pipe breakage, a trade-off where both of these quantities were minimised through MOO.

## 3.5 Estimation Distribution Algorithm (EDA)

The EDA was presented in Section 2.6 as a heuristic algorithm that significantly differs from EA. Let us now look at current literature pertaining to this heuristic.

### 3.5.1 EDA in Scheduling and Planning

The EDA was applied in [65, 209] to solve a RCPS problem set in a static environment where tasks were executed in various modes (ways). It was also applied in [48, 178, 233] to solve job-shop scheduling problems. Some important guidelines for designing implementations of the EDA were developed in [48] for solving a machine scheduling problem. A type of Particle Swarm Optimisation technique combined with a version of the EDA was applied in [129] to solve a flow shop problem.

The Bayesian Optimisation Algorithm (BOA) is a variety of the EDA and utilises Bayesian networks (statistical inference networks) to estimate its probability matrix (described in Section 2.6). However, it requires solving NP-complete problems to search for the Bayesian networks which fit given criteria best [151]. The search is computationally expensive. BOA was applied to determine task schedules for a multi-processor computing system where its Bayesian network was established by learning the dependencies between the tasks [222].

### 3.5.2 Multi-Objective EDA

The EDA was applied in [87] to improve the performance of a technique which utilises *Reinforcement Learning* (investigated in Section 2.10.1), a performance for solving a multi-objective problem. In the improvement process, its probability matrix was revised every time the Reinforcement Learning system dynamically interacted with an environment. The EDA was applied to classify tissues at molecular level in [66]. The selection (an implementation of step 3 in Figure 2.18) of genotypes in this work is based on the Pareto ranks and crowding distances (explained in Section 2.5.3) of their corresponding phenotypes. This work is one of the bases upon which the thesis relies in its application of the EDA.

### 3.5.3 Dynamic EDA

The EDA was applied to determine the optimum of functions with time-varying dimensions in [78,79]. In this application, the utilised probability matrix was a Mixed Gaussian Model whose number of clusters was determined through Bayesian Information Criteria [182]. The EDA was applied to some dynamic optimisation problems with fixed dimensions [130, 152, 218]. In [152], some parameters of the current state of a dynamic environment were used to retrieve, from a memory, parameters of a probability matrix utilised before hand for solving a problem set in a previous state of the environment. These retrieved parameters were utilised to form the probability matrix of an EDA process that solved the problem set in the current state of the environment. Further, population diversity loss correction (discussed in Section 2.6.1) was applied in the EDA process. A variant of EDA called Univariate Marginal Distribution Algorithm (UMDA) was applied in [130] to combat population diversity loss. It was also applied in [218] with a multimodal probability model and multi-population to tract multiple peaks of the search space associated with a problem.

## 3.6 Robust Proactive Approaches

Some unpredictable events in an environment, where a scheduling problem is set, can cause uncertain changes in the parameters or data of this problem; examples of such changes are events such as a road blockage caused by a landslide, a machine breakdown in a manufacturing system [106], new activities appearing in the process of executing a schedule [208, 241], or asset breakdown [227]. The parameters or data could be the duration of activities during a schedule execution, the number of required or available resources, sensor measurements, or transport and goods information on supply chains [18]. The solution (schedule) to a scheduling problem is defined as being robust if it is still suitable to be implemented, at a specified satisfaction level, despite minor uncertain changes in the parameters and/or data of the problem [104, 106]. This section provides

information on the approach, referred to as robust proactive, used to determine robust solutions and also on some related topics.

### 3.6.1 Support for Schedule Robustness

By virtue of its robustness, the revision of a robust schedule can be unnecessary when slight disturbances occur in the environment in which it is implemented [40]. On the other hand, a non-robust schedule could require frequent revisions which could be detrimental in some applications. For instance, air flight schedules that need to be revised every time minor but frequent flight disruptions occur could be detrimental for airline businesses [241]. The search for robust schedules could entail anticipation or forecast. It is advantageous to anticipate risks arising from executing schedules to avoid possible unfavourable schedule implementation outcomes whether in finance, vacation, or other fields of endeavour [23]. In building a robust supply chain schedule, the ability to forecast could also help envisage the coordination of the activities to attain the goals agreed upon by the constituting entities of the supply chain [23]. Uncertainty in transport and goods information is common in logistics and supply chain applications due to their opaqueness in the entire chain to all entities involved or due to the stochastic customer demands [104, 220].

The techniques for finding robust solutions have also been applied to solve problems with stochastic variables [12, 18, 208, 241]. One such problem was found in Multiprotocol Label Switching (MPLS) where the information transmission blockage is stochastic [134]. In a supply chain of an electronic market, the costs of supplied goods are stochastic [221]. The Markov Decision Process-based Q-learning [9] was applied to solve a project scheduling problem that involved stochastic parameters. The following sub-sections will present some robust proactive techniques.

### 3.6.2 Distribution-Based

One way to obtain a solution to a problem that is robust against a noisy (stochastic) parameter is to take the first or second order moment of the problem's fitness function with respect to the noisy parameter and then optimise the result. Consider a probability density function $\phi(z)$ of a random variable $z$. The first order expectation of a fitness function $f(\mathbf{x}, z)$, also called the effective fitness function [106], with respect to the random variable is,

$$\mathbf{E}_z [f(\mathbf{x}, z)] = \int_{-\infty}^{\infty} f(\mathbf{x}, z)\, \phi(z)\, dz \equiv g(\mathbf{x}) \tag{3.1}$$

and the second order expectation of the fitness function is,

$$\sigma_z^2 (f(\mathbf{x}, z)) = \int_{-\infty}^{\infty} f^2(\mathbf{x}, z)\, \phi(z)\, dz - \mathbf{E}_z [f(\mathbf{x}, z)]^2 \tag{3.2}$$

where $\mathbf{x}$ is a vector of deterministic variables. The effective fitness function $g(\mathbf{x})$ can be different to and can have more local optima than those of its corresponding fitness function $f(\mathbf{x})$ [104]. Probability theory can be applied to the historical data of $z$ to find $\phi(z)$ [220].

The optimisation of $\mathbf{E}_z [f(\mathbf{x}, z)]$ may not yield a sufficiently robust solution $\mathbf{x}$. On the other hand, optimisation of the variance $\sigma_z^2 (f(\mathbf{x}, z))$ alone may not account for solution quality [104, 106]. One remedy for this dilemma is to account for the expectation, the variance, and the original fitness function [106]. For example, in [104], the mean and the variance of a fitness function were optimised through MOO. In [106], a measure of the robustness $R$ of the solution to a problem is defined in terms of the standard deviation of the fitness function and the standard deviation of the random variables of the problem,

$$R = \frac{1}{n} \sum_{i=1}^{n} \frac{\sigma_f}{\sigma_{z_i}} \tag{3.3}$$

where $n$ is the number of random variables; $\sigma_{z_i}$ is the standard deviation of the statistically

independent random variable $z_i$; and

$$\sigma_f^2 = \sum_{i=1}^{n} \left( \frac{\partial f}{\partial z_i} \right)^2 \sigma_{z_i}^2; \tag{3.4}$$

This measure is a decreasing function of robustness.

### 3.6.3 Scenario-Based

Consider the average of a fitness function with several random variables $z_1, z_2, \ldots, z_m$,

$$\begin{aligned} \mathbf{E}_{z_1,z_2,\ldots,z_m} \left[ f\left(\mathbf{x}, z_1, z_2, \ldots, z_m\right) \right] = \; & \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f\left(\mathbf{x}, z_1, z_2, \ldots, z_m\right) \\ & \phi\left(z_1, z_2, \ldots, z_m\right) dz_1 dz_2 \ldots dz_m \end{aligned} \tag{3.5}$$

where $m$ is the number of random variables $z_i$, $1 \leq i \leq m$; $\phi$ is a multi-variate PDF; and $\mathbf{x}$ is a vector of deterministic variables. If $m$ is large the computation of the average could be extremely expensive. One economising approach is to generate values of the random variables, input these values to the fitness function and then average the output of this function. This approach is a type of MCS that has the advantage of circumventing the computational extravagances in solving problems with multi-variate PDFs [18]. For example, in a Vehicle Routing Problem (VRP), various matrices which represent destination and travel time were randomly generated and from this process various fitness values of this problem were obtained. These values were then averaged to yield an effective fitness value (defined above). The measure of schedule robustness was defined in this particular problem as the average delay where delay is the amount of time in missing a given vehicle arrival time window [180]. Notice that the random variables of this problem are statistically dependent: a traffic jam on one road could influence traffic conditions on other roads; and a delay-causing bad weather in one area could influence travel time in other areas. Further, if the fitness function utilised in the MCS is convoluted, the randomly generated input values in the MCS could possibly be utilised for simulating real-life scenarios before their respective scenario outcomes can be obtained, outcomes which are the

output values of the fitness function [18].

### 3.6.4 EA on Robustness

Let us now review some literature that explores the application of EA to the search of robust solutions. A numerical integration applied in either Equations 3.1, 3.2 or 3.5 could be costly as this samples several points in an associated fitness landscape. Note that even to determine the fitness value at a single sample could be costly, such as in some aerodynamic design optimisation problems [105]. Considering that an optimisation which applies a heuristic (e.g. EA) inherently possesses a population of solutions (which could be viewed as samples in the search space), then the integral (e.g. Equations 3.1, 3.2 or 3.5) involved for determining a robust schedule could be estimated from this population, an approach that could reduce the computational cost of the numerical integration [106]. In another approach, a PDF was approximated from an EA-produced population and was shown in some tests to be sufficient for determining a robust solution [106]. Several heuristic and statistical approaches to determine robust schedules can be found in [18, 104, 106].

A non-statistical but purely EA approach was applied in [197] to search for robust solutions. In this work, random perturbations on phenotypes were performed. If the perturbation of a phenotype reduces its fitness then it (i.e., it could be a schedule) is not considered as robust. If at each evolutionary cycle the offspring of non-robust phenotypes are not selected to subsist to the next cycle then a robust phenotype could be obtained at the end of the evolutionary cycles [106].

### 3.6.5 Multi-Objective Optimisation on Robustness

As explained above, the creation of a robust schedule – being a solution to a given scheduling problem – could entail the anticipation of events. If these events will not occur, the robust schedule could have a lower quality than that of a non-robust schedule as a solution

to the same problem. As an example, consider a man who creates a robust travel schedule. In this creation, he anticipates a severe weather condition by bringing equipment which could help him overcome this condition and would not require him to revise the schedule. In another case, he did not anticipate the condition and consequently did not bring the equipment and created a non-robust schedule which needs to be revised when the condition occurs, a revision which could incur travel delay. If in following the robust schedule the condition indeed occurs, he could arrive at his destination sooner than when he follows the non-robust schedule, i.e. the duration of the robust schedule could involve minimal travel time. However, if the anticipated condition does not occur, he could arrive at his destination later than by following the non-robust schedule, perhaps because of the burden in carrying the equipment, i.e. the duration of the robust schedule could not be minimal. Thus, it could be beneficial to trade-off the quality (shortness of travel time) and robustness of the schedule. In [180], a trade-off between the robustness and quality of a solution to a VRP Time-Windowed (VRPTW) problem was performed to determine a detour whereby the robustness and the quality of the solution are simultaneously high. In [18], the variance and expectation of the total cost (supplies, processing, transportation, shortage and capacity expansion) of a schedule were minimised to guard against the effects (i.e. made robust) of the large random changes of the total cost on schedule quality. The standard deviations of the random variables of a problem in [241] were minimised to determine a robust and high quality production schedule. A heuristic called starting time criticality was applied in a multi-objective project scheduling. It utilises the weights of the activities in the project and the variance of each activity duration to insert time buffers in front of critical activities, to cope with unexpected disruptions while implementing a high quality schedule. The insertions preserve an established project deadline [208].

### 3.6.6 Robust Reactive-Predictive Approaches

As explained in Section 2.4.2, the robust reactive-predictive approach in scheduling is a combination of the robust proactive and the reactive-predictive approaches. Let us now

review some work related to this combination. In creating a robust baseline production schedule, Bonfill, et al. [30], anticipate various types of uncertainties, from processing time, equipment availability, to technology change and market parameters. Their results showed that the benefit of considering robustness is heavily influenced by the parametric values of the associated problem; and that schedule robustness becomes less crucial as the need to revise the schedule intensifies during schedule execution. Lambrechts, et al. [117], applied the robust reactive-predictive approach to an RCPS problem. The baseline solution to this problem is created by giving tasks time slack, defined as the amount of time delay in commencing a task without the need to delay other tasks. Further, it is created by giving tasks resource slack whereby tasks are constrained only to utilise a certain number smaller than the maximum available number of resources. The system in [11] assists decision-makers to select, from several pro-actively created schedules, one that suites the current state of a job-shop environment every time this environment changes. The robust reactive-predictive approach was applied in [191] to create a schedule with minimised *Printed Circuit Board* (PCB) production cost, which includes penalties on mailing PCBs through express mail and penalties imposed by customers due to late delivery. The work of Umang and Bierlaire [199] investigated the difference between the cost of executing a berthing schedule of arriving ships that is created and revised through the reactive-predictive method (described in Section 2.4.2) and the cost of executing a schedule for the same environment created through the robust proactive method (explained in Section 2.4.2). The approach of this investigation is utilised in the thesis to demonstrate the significance of a robust reactive-predictive approach to solve the problems from $\mathcal{L}^3$ described in Section 1.2.

### 3.6.7 Anticipating Risk

As explained in Section 1.2, the search for solutions to any problem from $\mathcal{L}^3$ involves anticipation of a type of risk. Let us now review current literature that pertains to risk anticipation in schedule creation. The work of Bui et al. [41] investigated a robust proac-

tive approach to solve a multi-objective RCPS static problem. The approach anticipates the risk of being unable to execute the components of a schedule that immediately succeed just-finished duration-changed components. This risk is one of the few objectives to minimise in creating the schedule. Bruni et al. [39] also examine the risk in an RCPS with renewable resources and uncertain task duration modelled through a cumulative probability distribution function. In [140], the search of a schedule – for a hydro-electric operation – that used stochastic dual dynamic programming [153] considers protecting contracts with electricity consumers. In this protection, several risk factors are anticipated, e.g. price volatility. In the area of finance, Hochreiter [93] used a model of future returns of budget-allocated assets. This model is called scenario tree and determined through EA to be as close to reality as possible given the uncertainties in asset returns. Stochastic programming [173] is applied to this tree to determine the budget allocation which will maximise expected return and minimise investment risk. Tometzki and Engell [194] explored production planning problems that involve uncertain parameters, and applied stochastic models and MOEA to maximise profit and minimise risk of high economic loss during plan execution. In the domain of control theory, Ouarda and Labadie [148] solved a dynamic optimisation problem on a multi-reservoir system by applying Lagrangian formalism which incorporated a stochastic approach in accounting for risk of defying operational restrictions on water resources. Real-time optimisation explored in [225] utilised a transition matrix to model the dynamics of a chemical plant as a *Markov Decision Process* (MDP). Some nodes in the MDP correspond to failed states, e.g. broken machines, with transition probabilities to these states signifying risk. Direct application of memory-based EA approaches (e.g. McBAR), to the best of our knowledge, is absent in the domain of risk anticipation in dynamic environments.

## 3.7 DOE and RSM

As explored in Section 2.8, DOE is useful for configuring parametric values and/or categorical types of experiments to obtain mathematical models of a phenomenon under

investigation or for other purposes. Let us review some literature that made use of DOE. In the domain of scheduling and planning, DOE was applied for RCPS [65, 209], staff scheduling [176], flowshop scheduling [62, 108, 176], sequence planning [195], vehicle routing problem that applied Ant Colony Optimisation [165, 166, 196], and trading [217]. In conjunction with MOO, DOE was applied to solve problems in [219, 229]. All of these applications are set in static environments. Under dynamic environments, DOE was applied for solving problems in economic lot scheduling [162, 163] which involves the determination of production sequences, each implemented at similar production facility. Also, it was applied in [230] for solving semiconductor wafer production problems whose dimension varies in time. Note that the last problems are closely related to the problems from $\mathcal{M}$ that are investigated in this thesis. However, due to the several significant differences between the single-objective semiconductor wafer production problems and the multi-objective RCPS problems from $\mathcal{M}$, in this thesis the approaches in [230] were not adopted.

### 3.7.1 Performance Comparison

One thesis' goal involves the comparison of performance of techniques. Let us now explore the literature on the application of DOE to compare the performance of some EA techniques that solved similar problems. This type of application was undertaken in [184] through statistical hypothesis testing on performance. To measure performance, it utilised in [99] statistical quantities, such as mean and standard deviation, and computational time spent for determining solutions, and in [226] RSM-built models of computational time and memory storage. Without using statistical techniques, it was applied in [13] to the performance of techniques solving MOO problems, and in [50] with memory-based techniques. Without applying DOE, an Artificial Neural Network was used in [26] to model the performance of an EA-based technique. After training the network, computational cost was significantly saved by not executing the technique but instead by using the network to predict the technique's performance.

### 3.7.2 Search for Optimising Parameters

DOE and/or RSM were/was utilised to determine the performance optimising parametric values and/or categorical types of some EA algorithms. In [47], the parametric values of a Genetic Programming (GP) technique applied to solve a MOO problem were determined whereby this technique produced high-quality solutions. The authors applied *Fractional Factorial Design* (FrFD) to design treatments that contain the parametric values. The responses to these treatments were used to form a polynomial empirical model that has linear and interaction terms. First, the search for the parametric values separately applied the methods of steepest descent and ascent (investigated in Section 2.8.6). Between these two methods, the method of steepest descent was shown to be promising and consequently used to determine a treatment that yielded an approximate best performance of the GP technique. Next, BBD was applied with three-levelled factors and the treatment to accurately determine the performance-optimising parameters of the GP technique. A *Central Composite Design* (CCD) variant called Minimum Run Resolution V design was used in [165] to build the performance models of an *Ant Colony System* (ACS) applied to solve vehicle routing problems. The search for the performance-optimising parameters of the ACS involved a MOO which used the performance models and did not involve the method of steepest descent/ascent. CCD was also applied to optimise the physical and production parameters of Integrated Circuit (IC) encapsulation in [111]. In [163], *Full Factorial Design* (FFD) was applied to determine the efficiency enhancing parametric values of a Tabu search algorithm used to solve an economic lot scheduling problem. FFD was also applied to find the efficiency optimising parametric values of a *Genetic Algorithm* (GA) used to solve a flow-shop scheduling problem in [62]. The factors considered in [62] were population size, selection rate, crossover method (six levels/different methods), crossover rate, mutation method (five levels/different methods), and mutation rate of the GA. The uniformity and the value of the sensitivity distribution of an electrical sensor were maximised using RSM and GA in [34]. The optimal parametric setting of an electrical discharge machining process was determined in [198] through Artificial Neural Network

(ANN), RSM and GA. The optimum cutting conditions were determined through RSM and GA in [170] to obtain minimum surface roughness. The treatments designed through the Taguchi method were utilised in the search for performance-optimising parametric values of the following techniques: Shuffled frog-leaping algorithm applied to solve an RCPS problem [65, 209]; Mixed-integer algorithm used to solve a non-linear stochastic staff scheduling problem [176]; and Self-guided Ants algorithm applied to solve a sequence planning problem [195]. DOE was applied to tune the weights of the multi-objective fuzzy rule criteria for a semiconductor wafer production scheduling problem that has a time-varying dimension [230].

### 3.7.3 Other Usages of RSM Models

The RSM models developed in [165] were utilised to predict the performance of an ACS for solving a vehicle routing problem. Similar approach was undertaken in [143] to predict the quality-improving size of a weld zone. CCD was utilised in building an RSM model for predicting surface roughness at different cutting conditions considered in [103].

A simple but effective RSM model was developed in [210] as a surrogate to a computationally expensive model and was utilised for MOEA. RSM was used to create a model which is the reduced form of the complex model composed of differential and algebraic equations for chemical distillation investigated in [177]. A versatile approach was developed in [77] to reduce the experimental cost in building a RSM model with multi-dimensional output, instead of the customary scalar output. This model is the surrogate of a computationally expensive model of a dynamic temperature field defined by partial differential equations. As far as our research goes, there seems a lack of dedicated research to apply RSM to model the performance of memory-based EA techniques for solving problems set in dynamic environments.

## 3.8 Reinforcement Learning in Scheduling

Reinforcement Learning (RL), being one of the fundamental concepts of universal intelligence (utilised in this thesis), was applied to solve scheduling, planning and allocation problems. Let us now present literature related to this application. RL was applied in electrical power allocation [211]. RL states (refer to Section 2.10.1 for terminology) in this allocation were derived from the continuously varying electrical power input to an electrical system; actions signify the allocation percentages of the power; and reward is related to the difference between allocation result and targeted allocation value. A dynamic optimisation problem (DOP) was solved in this allocation without the use of EA. Another DOP was solved in [235] through EA with phenotypes which represent schedules of elevators; and through RL which learns the merits of scheduled tasks (as actions) encoded in the genes together with the merits. RL was applied to solve multi-objective DOP in [87, 139].

EA and RL were combined to solve non-DOPs, such as, the travelling salesman problem in [128]. In this combination, locations correspond to states, location selections correspond to actions, and distance travelled is one of the domains of a reward function. In [24], RL was applied to learn the best move in a Tabu Search [163] for solving a lot scheduling problem. And in [68], it was applied to learn the best attributes of EA operators, such as mutation rate and crossover point.

## 3.9 Multi-Agent System (MAS)

A software agent (referred to as an agent for brevity) is a computer program authorised in a system to select and execute actions independently or on behalf of other entities in the system [23]. MAS is a system that has one or more agents and environments. One way to simulate a system is to model each system-constituting entity – the environments in this system may be exempted – as an agent, and then allow these agents to interact among themselves and the environment. In the manufacturing system in [203],

for example, machines, personnel, and raw materials are simulated as agents that mimic their properties. These agents were put into action from which their manufacturing gain optimising parametric values are determined. MAS is suitable for systems that could be disassembled into components. This facilitates ease of adding new components/agents or agent capabilities to a system. Thus, MAS could be scalable [23]. Let us now discuss more properties of MAS.

### 3.9.1 Hierarchical Systems

The traditional scheme of managing a system is hierarchical where commands start from the topmost level of the hierarchy and reached the bottom passing through intermediate levels. However, studies showed that hierarchical systems have disadvantages not limited to the following: 1) inadequate flexibility and slow response when new critical conditions arise; and 2) the information flow across the hierarchies is impeded by some constraints and rules of hierarchical components [135]. Examples of the inflexibility can be found in hierarchical structures of some manufacturing plants where schedules developed and imposed by higher levels of these structures could quickly become useless after executing the first few components of these schedules [100, 203].

### 3.9.2 Heterarchical Systems

The heterarchical system is the opposite of the hierarchical system. Each of its agents has ample freedom to decide on the immediately necessary actions to be taken and is responsible only to itself [189]. However, in several applications, heterarchical systems have no central controller and no global coordinating information which precipitates to systemic disadvantages such as: 1) poor performance; 2) behaviour predictions are challenging; and 3) instability of performance that can result in chaos [192], instability that is due to lack of coordination of agents [227].

### 3.9.3 Holonic Systems

The hybrid of hierarchical and heterarchical systems is called a holarchical or holonic system which in some cases can overcome the disadvantages of the two hybridising systems. The agents of holonic structures are called holons [113] which can function as whole or parts of a holonic system [227], i.e. holonic system is a type of MAS.

Being members of the partly heterarchical holonic system, holons are semi-autonomous. For example, in the case of a transport breakdown, transport personnel (represented by a holon) can utilise knowledge around the breakdown vicinity instead of executing suggestions from a far flung decision maker that would only delay problem resolution [23]. Being members of the partly hierarchical holonic system, holons may receive instructions from or be controlled by other holons of the system [227]. In a holonic system, an optimal plan for the locality of the holon could be formed with this holon at any level. For example, in a manufacturing application, holons were present at all levels. At the lowest level, where real-time operations dominated, the holons were coupled to the operations of the manufacturing control devices [38]. And on some higher levels, they are involved in planning [70].

Holonic systems can offer several advantages [76, 81, 89]. For example, holons that start from being disorganised could evolve autonomically and dynamically into a highly complex system triggered by some driving stimuli from their milieu. This immense plasticity can aid the development of systemic behaviours unknown to the holons [29,122], as can be observed in the schools of fish, human society, or some system containing interacting entities [138, 203]. Thus, a designer of holons need not plan the eventual remarkable behaviours of these holons, a facet of a holonic system that could result in a simplified design of this system [21]. Holonic systems are adaptive to the changes in the environments in which their holons interact. Holonic systems that interact with market environments were quick to adapt and flexible to market changes, abilities which are the key components of business competitiveness [100, 101, 122].

The evolution of a holonic system into an intricate system could be influenced by the learning mechanism of each of its holons, a mechanism that helps these holons to discover opportunities for systemic performance improvements. In a specific application, the learning of holons had boosted the performance of a holonic system in [172] by 500%. The learned knowledge of holons in [203] was use to forecast market demands.

Coordination of holons could be achieved in various ways. When holons leave traces as they explore or interact with their environment, indirect coordination could emerge that could help them to self-organise. This is akin to ants leaving pheromones as they move around that can help other ants to search for food, i.e. an emergence of a systemic intelligence that requires no control imposed on the ants [122]. The direct coordination among holons could be facilitated by inter-holon communication or system-global broadcast of information [23]. And, a control among holons can foster coordination among themselves [227].

The robust proactive approach defined in Section 2.4.2 entails anticipation. Being a part of their anticipation, holons of the holonic system in [203] explored various systemic states to guard the system against unexpected operational disruptions. The holons of some holonic systems performed statistical analysis to make these systems proactive [101]. A holonic manufacturing system in [14] with proactive agents was shown to be resilient to the failures of system components, to the changes in volume and variety of demands, and to the changes in job priority.

There are several holonic architectures in the literature, such as ADACOR [122], PROSA [204], MACSIMA [172], and Organisational Control Architecture (OCA) [227]. OCA in [227] is modelled from a command and control organisation, such as in the planning of a military mission. Note that this type of environment is similar to that utilised in this thesis.

### 3.9.4 MAS Applications

Traditionally, humans mediate between some components of most supply chains. With the fast-paced business transactions nowadays, agents in a supply chain could bear mediator roles. In this supply chain, each agent could work on behalf of each supply chain entity and cooperate with other agents through online inter-agent interactions to come up with decisions in a humanly-like as possible a manner [23, 94]. For example, suppose a customer purchases a can of milk from a supermarket. This transaction information could be transmitted online. Then, an agent representing the supermarket and another agent representing a milk producer may interact to decide on the wholesale price and the number of milk cans to be purchased by the supermarket from the producer to replenish the milk cans purchased by the supermarket customers.

An approach that utilised MAS was applied to solve a multi-objective dynamic paper production scheduling problem in [109]. Each agent in this application was dedicated to work on each stage of the production. The agents interact to determine the best feasible production schedule that satisfies several objectives. In a remarkable application, a MAS was utilised for the autonomous navigation of the Viking mission robot, thereby circumventing the need for an Earth-based control of the robot over the enormous Earth to Mars distance [23].

MASs (e.g. holonic system) could be extremely versatile for solving DOPs as exemplified in some applications discussed above. They may also be useful for solving the problems from $\mathcal{M}$ in this thesis. Their versatility could be enhanced by utilising learning or intelligent mechanisms for their individual holons [172, 203]. The chosen implicit memory-based EA approach (McBAR) in the thesis to solve problems from $\mathcal{M}$ could be useful in the future as one of the learning mechanisms. However, the scope of the thesis excludes the use of MAS.

# Test Environments

Chapter 4

The selection of some RCPS-related problems from $\mathcal{M}$ as the test problems in the thesis is based on the popularity of RCPS in the study of adaptation in dynamic environments [58]. Each of these problems is set in a military operation environment (MOE) such that it is referred to as the Military Mission Scheduling (MMS) problem. Note that, based on Equations 1.3, 1.5 and 1.6, there is a specific MOE for each of the problems. It is considered, however, that all sub-problems of the same problem are set in the same MOE. Section 4.1 explores the planning stage of MMS that is referred to as Military Mission Planning (MMP) and discusses the qualitative features of the MMS problems from $\mathcal{M}$. Sections 4.2 and 4.3 present concrete features of sub-problems (defined in Section 1.2.3) of problems from $\mathcal{L}^2 \cup \mathcal{L}^3$ and $\mathcal{O}^2$ (subsets of $\mathcal{M}$ and defined in Section 1.2.5) respectively; problems taken into account by the limited and general methods, respectively, to investigate some characteristics of techniques from $\mathcal{T}$ (defined in Equation 1.8). Section 4.4 elaborates on the procedure to computer simulate the problems whose mathematical definitions are found in Appendix A.1.

## 4.1 Military Mission Planning (MMP)

Military mission planning is a decision making process and an important element in military command and control aimed at providing a solution that implements the commander's intent, establishes activities, time or conditions for the operation, allocates resources, assigns tasks and coordinates subordinates. This is a complicated process that involves two aspects: (1) Science which deals with measurable factors such as capabilities, tech-

niques and procedures, and which is closely related to the analytic decision making; and (2) Art where the intuition of the commanders about the relationships between friendly forces, enemies and environment as well as the effects of the operation on the soldiers, is the focus and which can be considered as a kind of intuitive decision making. Mission planning is usually done as a matter of urgency or within a short time frame of a planning horizon [201].

It is quite common in military domains that each level in mission planning corresponds to a level of conflict: *strategic*, *operational*, and *tactical*, although the borders between these three is not always clear. The *strategic level* of a conflict involves determining national or alliance security objectives and developing and using national resources to accomplish those objectives. It establishes strategic military objectives, sequences the objectives, defines limits and assesses risks for the use of military and other instruments of power. It develops strategic plans to achieve the objectives, plans which are then provided to armed forces and other capabilities. Meanwhile, the *operational level* is designated for campaigns and major operations in order to accomplish strategic objectives within theatres or areas of operations. The linking of tactics and strategies is achieved by establishing operational objectives needed to accomplish the strategic objectives, sequencing events to achieve the operational objectives, initiating actions and applying resources to bring about and sustain those events. Lastly, the *tactical level* involves battles and engagements; they are planned and executed to accomplish military objectives assigned to tactical units. The focus of this level is on the ordered arrangement and manoeuvre of combat elements in relation to each other and to the enemy in order to achieve combat objectives established by the operational level commander. In other words, the context of tactical operations is defined at the strategic and operational levels [32, 201].

In this section, we focus on the planning process at the operational level. Planners at operational level need to follow the Operational Art (OA) of using military forces. According to OA, the issues at this level include (1) identifying the military conditions or end-state that constitute the strategic objectives; (2) deciding on the operational ob-

jectives that must be achieved to reach the desired end-state; (3) ordering a sequence of actions that lead to fulfilment of the operational objectives; and (4) applying the military resources (capabilities) allocated to sustain the desired sequence of actions. From this point onwards unless otherwise stated, we use the term mission planning to indicate planning at the operational level.

There is no doubt that the planning process is based on a particular military doctrine. However, the main steps are similar among militarised forces. We will take the JMAP framework from Australian Defence Forces (ADF) [232] as an example:

1. Initialisation: Including obtaining mission information (basically called Intelligence Preparation of the Battle-space - IPB)
2. Mission Analysis: Determining the objectives, available capabilities and other constraints for the mission. Also forming the commander's planning guidance for the next step.
3. Course Of Action (COA) Development: Developing the course of actions (ways to achieve the objectives with regards to the constraints)
4. COA Analysis: Comparing and analysing COAs to obtain an optimal plan.
5. Decision & Execution: Deciding on the plan and executing it

Note that this is a repetitive process. Step 1 will be used to update information for all other steps. Once updated, the steps will be restarted for further analysis. If the time is available and the urgency is low, we will have a DEliberate Planning (DEP) process. This is used to produce plans for contingencies and for later execution. When a plan is needed for immediate action or within a very short time with a high urgency, there will be a crisis action planning (or immediate planning) process (CAP). These two types of planning are closely interrelated. DEP produces the plans, while CAP uses these plans and adapts them to current situations. In other words, CAP provides situation awareness.

For OA, it is also essential to define several key concepts. The concept of an end-state can be considered as the final behaviour of the system when the system stops operating. For JMAP, the end-state is defined as the set of conditions which will achieve the strategic objective. The national end-state is the set of desired conditions, incorporating the elements of national power, that will achieve the national objectives. The military end-state is the set of desired conditions beyond which the use of military forces is no longer required to achieve the national objectives. The military end-state for a mission planned at the operational level is defined by the command at the strategic level. This needs to be done in Step 2.

Next is the concept of *Centre Of Gravity* (COG). For JMAP, COG is considered as the key characteristic, capability, or locality from which a military force derives its freedom of action, strength or will to fight. Another concept is *critical vulnerability* (CV). That is a characteristic or key element of a force that if it is destroyed, captured, or neutralised will significantly reduce the fighting capability of the force and its COG. Also the concept of *decisive point* (DP) needs to be considered. JMAP defines a DP as a major event that is a precondition to the successful disruption or negation of COG. A DP can be defined either in a time or geographical space. A mission plan might have many DPs. The line of DPs forms a path of attack or defeat to achieve the end-state. We also call it the *line of operations* - (LOP). Determining LOP is the most important component of operational level planning. The sequence of operations needs to be followed in order to achieve the end-state. Each operation or (action or task) is defined to take care of one DP.

Determining COG, CV, and especially DPs is a challenge. It relies very much on the experience and knowledge of the staff and commanders. Further, given that these concepts are defined, finding LOPs is also a big issue. The scope of the problem means that a large number of possible LOPs exist. The limitation of capabilities, synchronisation of operations (the precedence relationship between operations) and time, make it very difficult to arrange sequences of tasks to achieve all DPs. From a computational point of view, it is valuable to quantify these concepts. Given the limitation of capabilities,

precedence, and time, there is a need to schedule the tasks to obtain the optimal sequence. The MMP problems can be transformed to MMS problems from $\mathcal{M}$ that has the following qualitative features.

### 4.1.1 Inputs

In the context of MMP a task is defined as a tactical operation (or action) that a military force must do for achieving a DP. From the previous analysis, we can see that modelling tasks is a very important component. Quantitatively, a task usually has the following:

1. *A set of pre-conditions*: Defining operational conditions for a task that need to be achieved before commencing it. It might be derived from the defined precedence relationship between tasks.
2. *A set of effects*: It is usually defined by the DP.
3. *Duration of execution*: A task cannot be executed without a time limit in order to synchronise with other tasks.
4. *A set of required capabilities*: This might be equipment, weapon, vehicles or troops.
5. *An identification number.*
6. *A status*: This can either be on-going, finished, or not yet executed. Cancelled un-executed tasks are not considered in this thesis.
7. *Starting time.*

The input information to the sub-problems of problems from $\mathcal{M}$ is Items 1 to 5 of the tasks in MOEs where the sub-problems are set. Feature values of the tasks in the MOEs are found in Table 4.1, the MOEs to where the sub-problems of problems from $\mathcal{L}^2$ and $\mathcal{L}^3$ (subsets of $\mathcal{M}$) are set. The first and second columns of this table are task IDs (Item 5) and durations (Item 3) respectively. Tasks with IDs of 2, 5, 16, and 17 in the table are exemplified in Section 1.2.1. They use the PNT illustrated in Figure 2.9.

Table 4.1: Properties of tasks in MOEs to where the sub-problems of problems from $\mathcal{L}^2$ and $\mathcal{L}^3$ are set

| Task ID | Duration | $R_1 \leq 16$ | $R_2 \leq 17$ | $R_3 \leq 5$ | $R_4 \leq 16$ |
|---|---|---|---|---|---|
| 1 | 18 | 4 | 0 | 0 | 0 |
| 2 | 13 | 10 | 0 | 0 | 0 |
| 3 | 16 | 0 | 0 | 0 | 3 |
| 4 | 23 | 3 | 0 | 0 | 0 |
| 5 | 18 | 0 | 0 | 0 | 8 |
| 6 | 15 | 4 | 0 | 0 | 0 |
| 7 | 19 | 0 | 1 | 0 | 0 |
| 8 | 12 | 6 | 0 | 0 | 0 |
| 9 | 17 | 0 | 0 | 0 | 1 |
| 10 | 19 | 0 | 5 | 0 | 0 |
| 11 | 22 | 0 | 7 | 0 | 0 |
| 12 | 16 | 4 | 0 | 0 | 0 |
| 13 | 23 | 0 | 8 | 0 | 0 |
| 14 | 19 | 3 | 0 | 0 | 0 |
| 15 | 10 | 0 | 0 | 0 | 5 |
| 16 | 16 | 0 | 0 | 0 | 8 |
| 17 | 15 | 0 | 0 | 0 | 7 |
| 18 | 23 | 0 | 1 | 0 | 0 |
| 19 | 17 | 0 | 10 | 0 | 0 |
| 20 | 22 | 0 | 0 | 0 | 6 |
| 21 | 27 | 2 | 0 | 0 | 0 |
| 22 | 22 | 3 | 0 | 0 | 0 |
| 23 | 13 | 0 | 9 | 0 | 0 |
| 24 | 13 | 4 | 0 | 0 | 0 |
| 25 | 17 | 0 | 0 | 4 | 0 |
| 26 | 18 | 0 | 0 | 0 | 7 |
| 27 | 23 | 0 | 8 | 0 | 0 |
| 28 | 17 | 0 | 7 | 0 | 0 |
| 29 | 20 | 0 | 7 | 0 | 0 |
| 30 | 20 | 0 | 0 | 2 | 0 |
| 31 | 12 | 6 | 0 | 0 | 0 |
| 32 | 17 | 0 | 0 | 0 | 1 |
| 33 | 19 | 0 | 5 | 0 | 0 |
| 34 | 16 | 0 | 0 | 0 | 8 |
| 35 | 15 | 0 | 0 | 0 | 7 |
| 36 | 23 | 0 | 0 | 0 | 0 |
| 37 | 18 | 0 | 0 | 0 | 7 |
| 38 | 23 | 0 | 8 | 0 | 0 |
| 39 | 17 | 0 | 7 | 0 | 0 |
| 40 | 20 | 0 | 7 | 0 | 0 |

As explained in Chapter 1, each type of resource in RCPS has a limited number of items, a number that is input information to the sub-problems of problems from $\mathcal{M}$. The third to the last columns of Table 4.1, respectively, are the resources $R_1$ to $R_4$ utilised by the tasks with IDs in the first columns. For example, task 12 is to bomb enemies for 16 time units and to utilise at most four light mortar batteries ($R_1$).

After one task is finished, each resource item it utilised is either transferred to another task location or returned to a depot called central base. Another input to the sub-problems of problems from $\mathcal{M}$ is the cost of moving each resource item from one task location to the next. Task location is identified by the ID of the task. For example, location labelled 7 in a battlefield is the venue for the execution of task with ID 7, e.g. to care for refugees by one infantry company. Central base has a location label of zero.

The transfer status of resource item can either be moved or unmoved, and its availability-to-task status can either be available, broken/dead, or occupied (utilised by a task). Additional inputs to the sub-problems of problems from $\mathcal{M}$ are the transfer status and the availability status of resource items, and the execution locations of tasks.

Let us prove that the total number of tasks can only be constant or increase. As explained in Chapter 1, the total $\sigma$ number of tasks is the sum of the number $F$ of finished tasks, number $G$ of on-going tasks, and number $X$ of to-be-executed tasks. Now, suppose that at the moment of an environmental change there are $X'' \geq 0$ to-be-executed new tasks[1], $G'' \geq 0$ just started new tasks, $G'$ of the previously on-going tasks that are finished, and $X'$ of the previously to-be-executed tasks that are started (considered as on-going). Considering that, as given in this thesis, none of the on-going and to-be-executed tasks are cancelled the number of finished, number of on-going and number of to-be-executed tasks at the moment of change are $F+G'$, $(G-G')+X'+G''$, and $X-X'+X''$ respectively. Thus, the total number of tasks at this moment is $\sigma' = (F+G')+(G-G')+X'+G''+(X-X'+X'')$ $= F+G+E+G''+X'' = \sigma+G''+X''$. Therefore, if there are no new tasks (i.e. $G''=0$ and $X''=0$) then $\sigma'=\sigma$, i.e. the total number of tasks remains. However, if there are new tasks (i.e. $G''+X''>0$) then $\sigma'>\sigma$, i.e. the total number of tasks increases.

### 4.1.2 Objective Functions

Minimisation of schedule cost and duration are the two objectives in sub-problems of problems from $\mathcal{L}^2$ and $\mathcal{O}^2$ (subsets of $\mathcal{M}$). As explained in Section 1.2 and elaborated in Section 2.7, another scheduling objective considered in the thesis is the probability of a schedule becoming infeasible due to resource constraint violations caused by changes in task durations. In addition to the schedule cost and duration, sub-problems of problems from $\mathcal{L}^3 \subset \mathcal{M}$ also have the probability of becoming infeasible as another objective to minimise. All of these objectives are mathematically defined in Appendix A.1.

[1]These are tasks which do not exist before any change in a considered environment

### 4.1.3 Constraints

The constraints imposed by a PNT and on resources are explained in Section 1.2.1. In Table 4.1, $R_i \leq n_i$ expresses the constraint on the total number of resource of type $i$ utilised by all on-going tasks; this is not to exceed $n_i$.

### 4.1.4 Output

Let us now investigate the computational products in solving the sub-problems of problems from $\mathcal{L}^2 \cup \mathcal{L}^3 \cup \mathcal{O}^2$. Some of these products are genotypes, each expressed as an ordered set,

$$I = \langle I_1, I_2, \ldots, I_N \rangle \,, \tag{4.1}$$

where $I_i$s are the ID of tasks in a MOE and $N$ is the genotype length. From here onwards, any genotype has a form similar to that in Equation 4.1. The ordering of IDs in the genotype is dependent on a given PNT. The ordering rule is:

*Any task with ID $I_k$ at index $k$ in the genotype have successor tasks with IDs $I_j$ which must be at index $j$ in the genotype, where $j > k$.*

For example, the section $\langle 1, 14, 2, 7, 3, 9 \rangle$ of a genotype satisfies the ordering rule based on the PNT depicted in Figure 2.9. Given a PNT, any genotype whose IDs satisfy the ordering rule is characterised as task-precedence feasible. Genotypes are generated either through SSGS (explained in Section 2.3) or by some other means, such as described in Section 5.1.

The starting time of each task in the MOE is another computational product. It is determined either through SSGS or by some other means, such as described in Section 5.1.4. Any solution to a sub-problem of problems from $\mathcal{L}^2 \cup \mathcal{L}^3 \cup \mathcal{O}^2$ is an ordered set of determined starting times and referred to as a schedule,

$$S = \langle T_1, T_2, \ldots, T_k, \ldots, T_N \rangle \,, \tag{4.2}$$

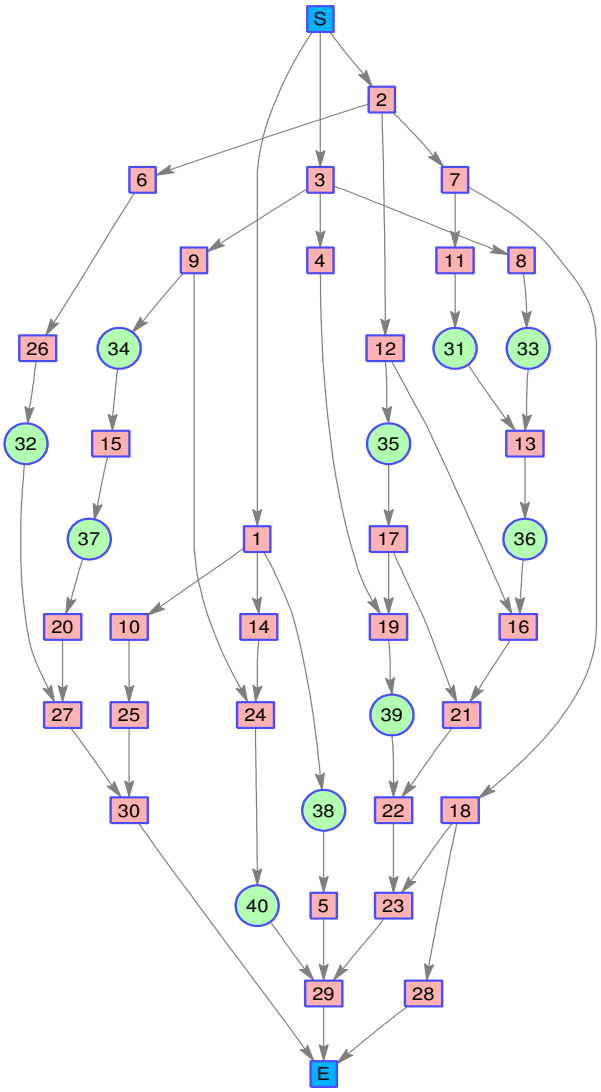



Figure 4.1: Precedence network of tasks for TNIS $T_4$

where the task with starting time $T_k$ at index $k$ in the schedule has ID $I_k$ at index $k$ in the genotype in Equation 4.1. By the ordering rule of IDs in genotypes, any task with starting time $T_k$ in Equation 4.2 must have successors with starting time $T_j$ where $j > k$.

Sample schedules are depicted in Figures 4.2 and 4.3 whose features are similar to those of Figure 2.3 (described in Section 2.1.1). All tasks in the schedules obey the PNT found in Figure 4.1. For example, in Figure 4.2 task 2 ends at 13 time units at which task 7 starts. Hence, task 7 succeeds task 2, thereby abiding the PNT. More details of Figures 4.2 and 4.3 will be given in the next section.

### 4.1.5 Dynamic Factors of $\mathcal{M}$

Dynamics and uncertainties are unavoidable factors for military missions. This is the nature of wars where enemies as well as environmental aspects are highly unpredictable. This is the reason for introducing the concept of crisis action planning in Section 4.1. One important requirement from the US Army is that the planning process needs to be continuous and adaptive to any changes. The presence of these factors, such as delay in mission execution, failure of capabilities or uncertain intuition of commanders on the relationship between operations of the mission, makes the task of mission planning more complex [110, 201, 232] with a large number of what-if scenarios that usually goes beyond the handling ability of human planners. Hence, there is a need for finding a robust and responsive mechanism for the support and planning staff. These mechanisms could be the techniques enumerated in Section 2.4.2 that are intended to solve dynamic problems from $\mathcal{L}^2 \cup \mathcal{L}^3 \cup \mathcal{O}^2$.

Some factors in dynamic environments are already described in Chapter 1 and Section 2.4. Here, we formally identify dynamic factors in MOEs where the problems from $\mathcal{L}^2 \cup \mathcal{L}^3 \cup \mathcal{O}^2$ are set:

1. Execution time of tasks that varies according to Equation A.9
2. Availability of resources
3. Number of tasks

It should be noted that any changes to a military mission plan can cause an enormous cost in terms of logistics and safety.

Let us exemplify the dynamics of a problem $p^3$ from $\mathcal{L}^3$. The schedule depicted in Figure 4.2 is one of the solutions to the sub-problem $p_5^3$ of problem $p^3$ set in the snapshot of the MOE taken immediately after the fifth SOSA, the state alteration at 13 time units. The schedule depicted in Figure 4.3 is one of the solutions to the sub-problem $p_6^3$ of problem $p^3$ set in the snapshot taken immediately after the sixth SOSA of the MOE, the state

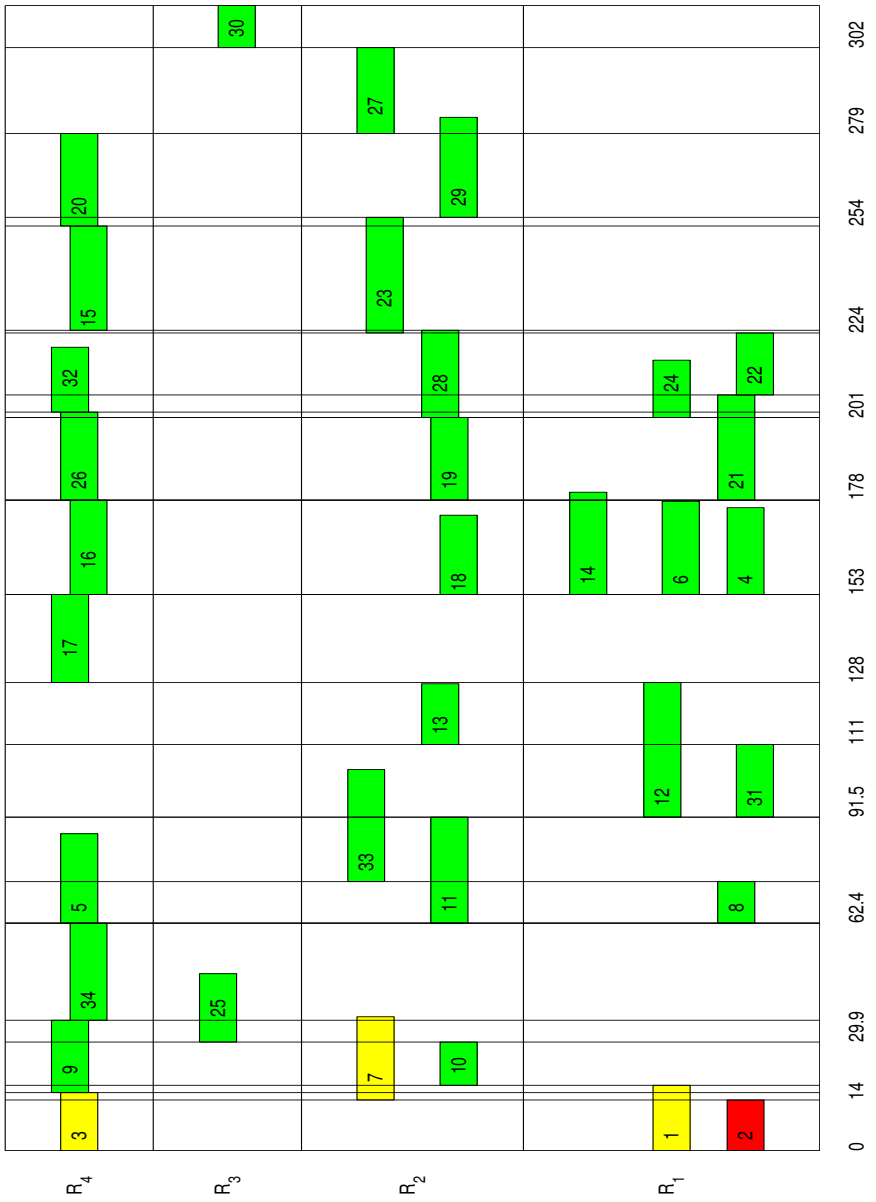



Figure 4.2: Solution to the sub-problem set in the fifth snapshot

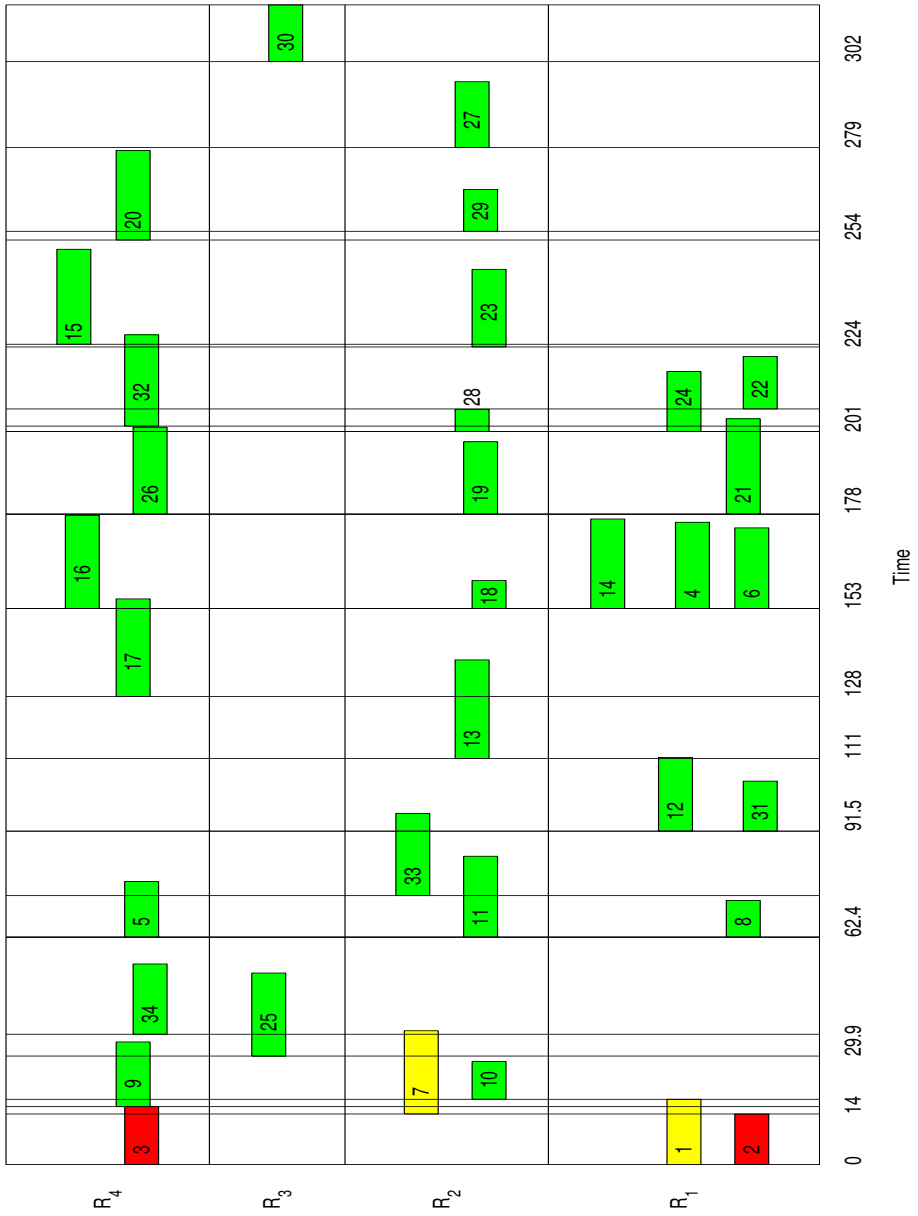



Figure 4.3: Solution to the sub-problem set in the sixth snapshot

alteration at 16 time units. Problem $p^3$ has attributes from Table 4.1 and PNT in Figure 4.1. The duration of each task in the schedule depicted in Figure 4.3 is an alteration, following Equation A.9, of the duration of similarly-IDed task in the schedule depicted in Figure 4.2. The starting times of the similarly-IDed tasks are identical but their durations may not be due to the dynamics of the MOE. This is true for all similarly-IDed tasks depicted in the Figures 4.2 and 4.3. This illustrates the dynamics of task duration.

In Figures 4.2 and 4.3, the red rectangles correspond to a finished task, e.g. task 2; the yellow rectangles to on-going tasks, e.g. tasks 1 and 7; and the other coloured rectangles to tasks yet to be executed. Based on Table 4.1 and on the PNT of Figure 4.1, when the MOE – to where $p^3$ is set – changes at 13 time units (which corresponds to Figure 4.2), tasks 1, 3 and 7 are still on-going, task 2 is finished, and the rest are yet to be executed. Further, at the next SOSA of the MOE at 16 time units (which corresponds to Figure 4.3), tasks 1 and 7 are still on-going, tasks 2 and 3 are finished, and the rest are yet to be executed. This illustrates the dynamics of task status.

## 4.2 Problems from $\mathcal{L}^2$ and $\mathcal{L}^3$

Section 4.1 provides an intuitive description of the general features of the problems from $\mathcal{L}^2 \cup \mathcal{L}^3$, while Appendix A.1 presents mathematical formulations of these features. This section provides information on some parametric values and categorical types of the features of the problems from $\mathcal{L}^2 \cup \mathcal{L}^3$.

Each task in the MOEs, where the problems from $\mathcal{L}^2 \cup \mathcal{L}^3$ are set, utilises only one of the following four types of resources:

- Light Mortar Batteries ($R_1$) $\leq 16$
- Infantry Companies ($R_2$) $\leq 17$
- C130s ($R_3$) $\leq 5$
- Apache helicopters ($R_4$) $\leq 16$

The maximum number of items of any listed resource that can be utilised by all ongoing tasks in the MOEs is indicated near the resource label $R_i$, $1 \leq i \leq 4$. Note that the availability status (defined in Section 4.1.1) of any of the resources can change from available to occupied (e.g. mortars firing) or vice-versa (e.g. mortars returned from combat to depot) in the course of executing a schedule, i.e. without the effects of a dynamic MOE. As proposed in this thesis, the only effect of the MOE on the availability status is to change this status from being available or occupied to broken (e.g. killed infantry soldier).

As pointed out in Section 4.1.1, properties of tasks in the MOEs where the problems from $\mathcal{L}^2 \cup \mathcal{L}^3$ are set are found in Table 4.1. Any one of the MOEs has 30 original tasks (which exist before this MOE changes) that abide by the PNT depicted in Figure 2.9. A total of ten new tasks are added to the original tasks during the entire dynamics of this MOE. New $dN$ tasks, which occur in a particular SOSA of the MOE, have IDs $I$, $l < I \leq l + dN$, where $l$ is the maximum ID of tasks which existed prior to this SOSA. For example, if at the last SOSA the maximum task ID is 30 and there are four new tasks at the current SOSA, then these new tasks are labelled with IDs of 31 to 34.

### 4.2.1 Labels

For convenience in the following discussion, we ascribe labels to some categorical types of problems from $\mathcal{L}^2 \cup \mathcal{L}^3$. The considered types of simultaneous changes in the MOEs where the problems are set are listed in Table 4.2. The first column contains labels for types of changes, and the second contains the parameters that change simultaneously. For example, type 6 indicates simultaneous changes in any one of the MOEs on task duration, total number of tasks, and resource availability. Change type 0 is referred to as *light* change type and the other types as *severe* change type.

The types of changes are chained to form a sequence of changes. Each *type of sequence of changes* (TSC) is presented as a column, labelled $S_i$, in Table 4.3. The first and last columns of this table contain, respectively, the SOSAs and times at which a specific type

Table 4.2: Types of environmental changes

| Type | Parameter |
|---|---|
| 0 | task duration |
| 1 | resource availability |
| 2 | total number of tasks |
| 3 | task duration<br>total number of tasks |
| 4 | task duration<br>resource availability |
| 5 | total number of tasks<br>resource availability |
| 6 | task duration<br>total number of tasks<br>resource availability |

of changes occur. For example, sequence $S_3$ of changes in one of the MOEs begins and is followed by change in task duration only (type 0 in Table 4.2), then is succeeded by simultaneous changes in task duration, total number of tasks, and resource availability (type 6 in Table 4.2) at the third SOSA of the MOE. The number of moments at which any one of the MOEs changes state is 12.

Table 4.3: Types of sequence of changes (TSC)

| SOSA | $S_1$ | $S_2$ | $S_3$ | Time |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 4 |
| 2 | 0 | 0 | 0 | 6 |
| 3 | 0 | 2 | 6 | 8 |
| 4 | 2 | 1 | 0 | 12 |
| 5 | 0 | 0 | 4 | 13 |
| 6 | 6 | 3 | 0 | 16 |
| 7 | 1 | 0 | 0 | 19 |
| 8 | 4 | 4 | 5 | 23 |
| 9 | 0 | 6 | 1 | 26 |
| 10 | 5 | 5 | 2 | 30 |
| 11 | 3 | 0 | 0 | 33 |
| 12 | 0 | 0 | 3 | 37 |

Some components of each TSC are types of changes that involve an increase in the total number of tasks. For example, $S_3$ has types of changes 6, 5, 2 and 3 that occur, respectively, at third, eighth, tenth, and $12^{th}$ SOSAs of one of the MOEs; and all of these types involve an increase in the total number of tasks (based on Table 4.2). *Task Number Increase Sequence* (TNIS), labelled as $T_k$ in Table 4.4, is a sequence of numbers of new tasks which appear at the SOSAs of the MOE with a given TSC. The order of the numbers in TNIS found at the first column of Table 4.4 is the same as the order of the SOSAs. Using the example above, if $S_3$ has TNIS of $T_5$, there will be five, three, one and one new tasks which appear at the third, eighth, tenth, and $12^{th}$ SOSAs of the MOE, respectively (based on Tables 4.3 and 4.4). Note that even with similar TSC but with different TNIS,

the number of new tasks in a given SOSA of the MOE could be different.

Table 4.4: Types of task number increase sequence (TNIS)

| Order | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ |
|---|---|---|---|---|---|
| 1 | 3 | 4 | 5 | 6 | 7 |
| 2 | 2 | 4 | 3 | 2 | 1 |
| 3 | 2 | 1 | 1 | 1 | 1 |
| 4 | 3 | 1 | 1 | 1 | 1 |

Problems from $\mathcal{L}^2 \cup \mathcal{L}^3$ with TNIS of $T_3$, $T_5$, $T_6$, $T_7$ or $T_4$ abide with the PNTs illustrated in Figures 4.4 to 4.7 or 4.1 respectively. The PNTs are formed by placing new nodes (which correspond to new tasks) on the original PNT, illustrated in Figure 2.9, and differ in their forms by the locations at which these new nodes are placed.

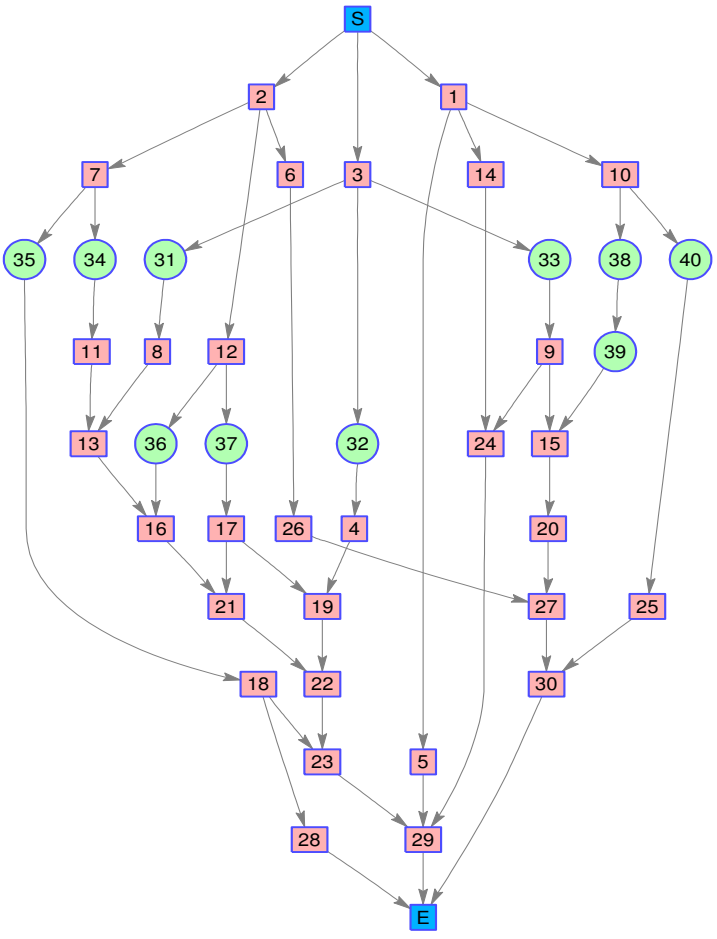



Figure 4.4: Precedence network of tasks for TNIS $T_3$

Some components of the sequences of changes in Table 4.3 involve change in task duration that is modelled by Equation A.9 whose parameter $\delta$ has to be specified. To recapitulate, a problem from $\mathcal{L}^2 \cup \mathcal{L}^3$ is specified by a TSC (e.g. $S_3$), a TNIS (e.g. $T_5$)

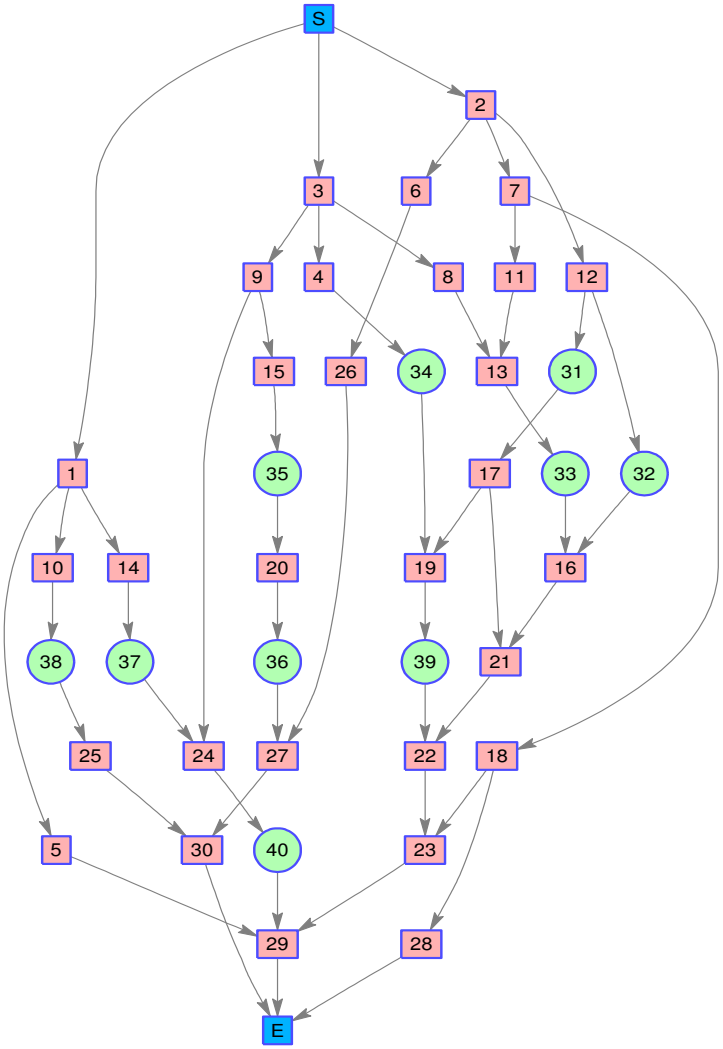



Figure 4.5: Precedence network of tasks for TNIS $T_5$

in the TSC, and a value of $\delta$ in Equation A.9. Note that the actual amount of change in task duration is not specified in the problem but instead the value of $\delta$ is provided. Thus, to distinguish, we define a problem simulation as being a problem with its task duration change fully specified. More will be said regarding problem simulation in Section 4.4.

The problems from $\mathcal{L}^2$ have labels $l$ listed in Table 4.5 and denoted as $l^2$. For example, a problem from $\mathcal{L}^2$ with label 5 is denoted as $5^2$. With the mid column (of TNIS types) in Table 4.5 as the reference column, the problems from $\mathcal{L}^2$ with labels at the left and right correspond to $\delta = 3.0$ and $\delta = 6.0$, respectively; problems with labels under $S_i$ have TSC of $S_i$; and problems with labels at the same row as $T_k$ have TNIS of $T_k$. For example, problem 25 has TSC of $S_1$, TNIS of $T_7$, and task duration changes modelled by Equation

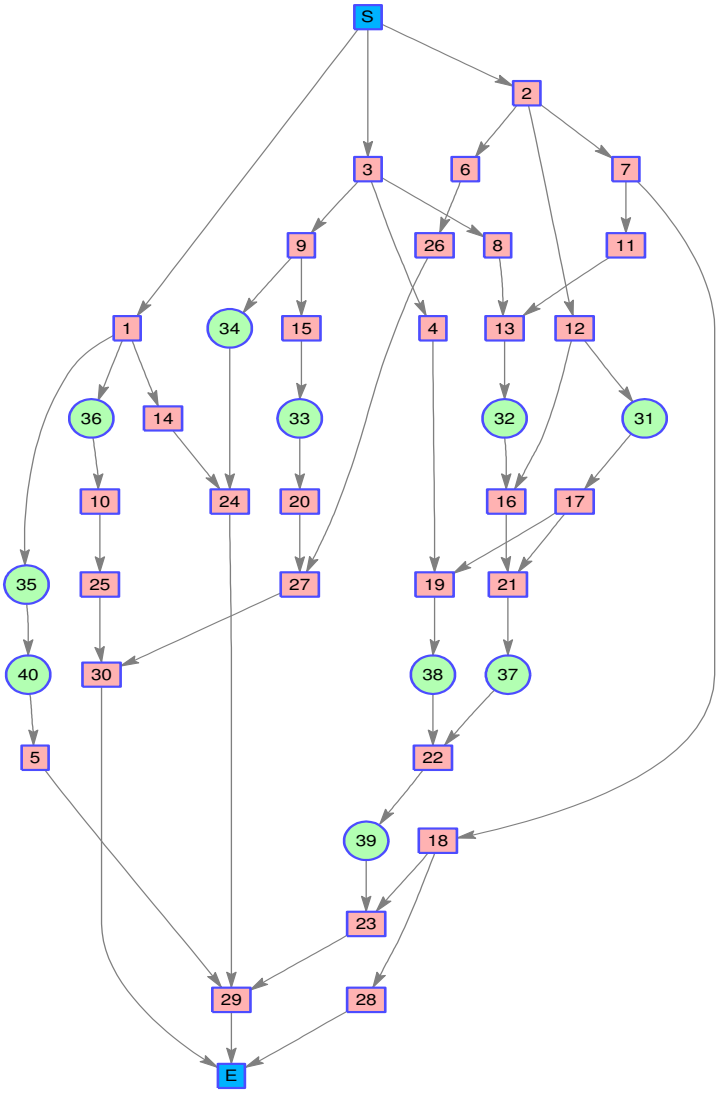



Figure 4.6: Precedence network of tasks for TNIS $T_6$

A.9 with $\delta$ of 3.0.

All the problems from $\mathcal{L}^3$ have labels $l$ of one to 30 and denoted as $l^3$. Thus, a problem from $\mathcal{L}^3$ can have an identical label to a problem in $\mathcal{L}^2$. These problems are differentiated by a superscript, e.g. $5^2 \in \mathcal{L}^2$ and $5^3 \in \mathcal{L}^2$. A problem from $\mathcal{L}^3$ with a particular label is defined by the objective to minimise the probability of its solution (schedule) to become infeasible due to resource constraint violations (explained in Section 2.7) and by all the features of the problem from $\mathcal{L}^2$ with a similar label found in Table 4.5. Note that a sub-problem of a problem from $\mathcal{L}^2 \cup \mathcal{L}^3$ may be denoted purely by numbers. For example, the sub-problem $12^2_7$ denotes a sub-problem of the problem labelled 12 from $\mathcal{L}^2$, the sub-problem set in the seventh snapshot of the MOE where the problem is set.

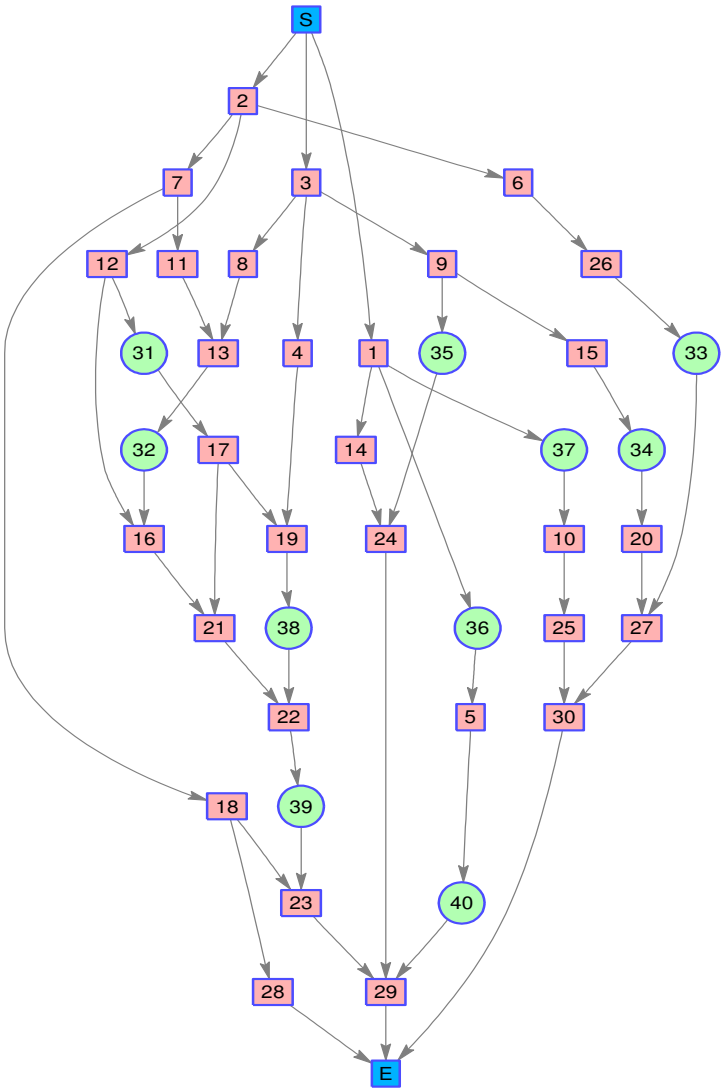



Figure 4.7: Precedence network of tasks for TNIS $T_7$

### 4.2.2 Tables as Functions

Let us define some functions that yield values found in Tables 4.2 and 4.4. Given a problem from $\mathcal{L}^2$ labelled $l$, a TSC $S_i$ of this problem can be obtained using Table 4.5. Given a SOSA $s$ of the MOE where problem $l^2$ is set, a change type label $c$ can be determined using $S_i$ in Table 4.3. Thus, Tables 4.5 and 4.3 serve as a function $Ct$,

$$c = Ct\,(l, s)\,. \tag{4.3}$$

Using $c$ in Table 4.2, the MOE attributes that change simultaneously can be determined. For example, the problem labelled 3 has a TSC of $S_3$, based on Table 4.5. From the $S_3$

Table 4.5: Problem labels

| δ = 3.0 | | | | | δ = 6.0 | | |
|---|---|---|---|---|---|---|---|
| $S_1$ | $S_2$ | $S_3$ | | | $S_1$ | $S_2$ | $S_3$ |
| 1 | 2 | 3 | | $T_3$ | 4 | 5 | 6 |
| 7 | 8 | 9 | | $T_4$ | 10 | 11 | 12 |
| 13 | 14 | 15 | | $T_5$ | 16 | 17 | 18 |
| 19 | 20 | 21 | | $T_6$ | 22 | 23 | 24 |
| 25 | 26 | 27 | | $T_7$ | 28 | 29 | 30 |

column of Table 4.3, the change type at the eighth SOSA of a MOE is 5, i.e. $Ct\,(3, 8) = 5$. Based on Table 4.2, type 5 of changes signifies simultaneous changes in resource availability and the total number of tasks in the MOE. This result implies that the simultaneous changes occur at the eighth SOSA of the MOE where problem labelled 3 is set.

Table 4.5 serves as a function $Pt$ that maps a problem labelled $l$ to type $p$ of TNIS,

$$p = Pt\,(l)\,. \tag{4.4}$$

For example, $T_3 = Pt\,(3)$, i.e. problem 3 has TNIS of $T_3$ based on Table 4.5.

Note that, based on the $S_3$ column of Table 4.3, the increase in the total number of tasks for the third occasion is at the tenth SOSA of the MOE where the problem with TSC of $S_3$ is set. Further, the intersection of the third row (which is the third occasion) and $T_3$ column in Table 4.4 is two, the number of new tasks that appear at the tenth SOSA of the MOE (the first and the second occasions are at the third and eighth SOSA respectively). Thus, the set of Tables 4.2, 4.4 and 4.5 serves as a function $Nt$ to determine the number of new tasks,

$$w = Nt\,(l, s)\,. \tag{4.5}$$

This function is applicable only when $Ct\,(l, s)$ yields a type of change that is either 2, 3, 5 or 6 which, based on 4.2, all involve an increase in the total number of tasks. Otherwise, there is no new task at a given $s^{th}$ SOSA of the MOE where the problem $l$ is set.

Let the functions $Nt$, $Pt$ and $Ct$ be distributive to vector elements, i.e. if $\mathbf{l}$ =

$\langle l_1, l_2, \ldots, l_m \rangle$,

$$\begin{aligned} \mathbf{c} &= Ct(\mathbf{l}, s) \\ &= \langle Ct(l_1, s), Ct(l_2, s), \ldots, Ct(l_m, s) \rangle, \end{aligned} \tag{4.6}$$

$$\begin{aligned} \mathbf{p} &= Nt(\mathbf{l}, s) \\ &= \langle Nt(l_1, s), Nt(l_2, s), \ldots, Nt(l_m, s) \rangle \end{aligned} \tag{4.7}$$

and

$$\begin{aligned} \mathbf{w} &= Pt(\mathbf{l}) \\ &= \langle Pt(l_1), Pt(l_2), \ldots, Pt(l_m) \rangle. \end{aligned} \tag{4.8}$$

where $l_i$ is a problem label; and $m$ is the dimension of $\mathbf{l}$. For example, for an ordered set of problem labels $\mathbf{l} = \langle 1, 7, 13, 19, 25 \rangle$, $Ct(\mathbf{l}, 4) = \langle Ct(1, 4), Ct(7, 4), Ct(13, 4), Ct(19, 4), Ct(25, 4) \rangle = \langle 2, 2, 2, 2, 2 \rangle$ which implies that all problems with labels in $\mathbf{l}$ has change type 2 (change in the total number of tasks) at the fourth SOSA of the MOEs where these problems are set.

## 4.3 Problems from $\mathcal{O}^2$

Some parametric values and/or the categorical types of the features of the problems from $\mathcal{O}^2$ will now be explored. These problems are taken into account by the general method (explored in Section 1.4.1) used to investigate the characteristics of the techniques from $\mathcal{T}$. Note that the MOEs considered in this section are where the problems are set. Any one of the MOEs has six moments of state alterations which occur at 4, 8, 12, 16, 21, and 26 time units after the start of the baseline schedule (unrevised schedule) implementation in this MOE.

### 4.3.1 Tasks

The properties of tasks in the MOEs are listed in Table 4.6, MOEs where the problems from $\mathcal{O}^2$ are set. In this table, the first and fifth columns are task IDs, the second and

sixth columns are task durations, the third and seventh columns are the numbers of items of resource type $R_1$ used by the tasks, and the fourth and eight columns are the numbers of items of resource type $R_2$ used by the tasks.

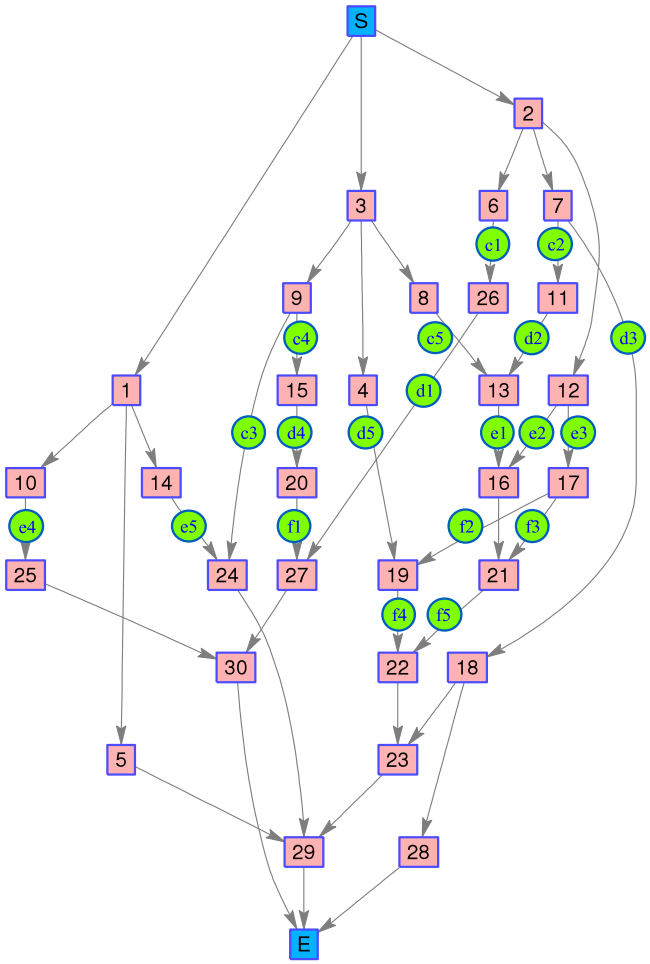



Figure 4.8: Precedence network of tasks of problems from $\mathcal{O}^2$

Each of the MOEs has 30 original tasks before it changes. The maximum number of tasks is 50. They obey the precedence relationship depicted in Figure 4.8. New tasks may appear only from the third to the sixth SOSA, the range of SOSAs referred to as *significant range of orders* (SRO). The maximum number of new tasks which appear at any SOSA in the SRO in any of the MOEs does not exceed five.

In Figure 4.8, the circular nodes labelled "c1" to "cn" represent $n$ new tasks that appear at the third SOSA. An analogous representation scheme is applied for new tasks which appear at the fourth to the sixth SOSA whereby letters "d" to "f" respectively are

Table 4.6: Properties of tasks in the MOE where the problems from $\mathcal{O}^2$ are set

| ID | Duration | $R_1 \leq 28$ | $R_2 \leq 28$ | ID | Duration | $R_1 \leq 28$ | $R_2 \leq 28$ |
|---|---|---|---|---|---|---|---|
| 1 | 18 | 4 | 0 | 26 | 18 | 7 | 0 |
| 2 | 14 | 10 | 0 | 27 | 23 | 0 | 8 |
| 3 | 16 | 0 | 3 | 28 | 17 | 0 | 7 |
| 4 | 23 | 3 | 0 | 29 | 20 | 0 | 7 |
| 5 | 18 | 0 | 8 | 30 | 20 | 2 | 0 |
| 6 | 15 | 4 | 0 | 31 | 12 | 6 | 0 |
| 7 | 19 | 0 | 1 | 32 | 17 | 0 | 1 |
| 8 | 12 | 6 | 0 | 33 | 19 | 0 | 5 |
| 9 | 17 | 1 | 0 | 34 | 16 | 8 | 0 |
| 10 | 19 | 0 | 5 | 35 | 15 | 0 | 7 |
| 11 | 22 | 0 | 7 | 36 | 23 | 0 | 1 |
| 12 | 16 | 4 | 0 | 37 | 18 | 7 | 0 |
| 13 | 23 | 0 | 8 | 38 | 23 | 0 | 8 |
| 14 | 19 | 3 | 0 | 39 | 17 | 0 | 7 |
| 15 | 10 | 0 | 5 | 40 | 20 | 0 | 7 |
| 16 | 16 | 8 | 0 | 41 | 27 | 2 | 0 |
| 17 | 15 | 0 | 7 | 42 | 22 | 3 | 0 |
| 18 | 23 | 0 | 1 | 43 | 13 | 0 | 9 |
| 19 | 17 | 0 | 10 | 44 | 13 | 4 | 0 |
| 20 | 22 | 6 | 0 | 45 | 17 | 0 | 4 |
| 21 | 27 | 2 | 0 | 46 | 18 | 7 | 0 |
| 22 | 22 | 3 | 0 | 47 | 12 | 3 | 0 |
| 23 | 13 | 0 | 9 | 48 | 17 | 6 | 0 |
| 24 | 13 | 4 | 0 | 49 | 19 | 0 | 5 |
| 25 | 17 | 0 | 4 | 50 | 16 | 0 | 8 |

used instead of “c”. The number of new tasks ranges from zero to five. Thus, as there are five circles in Figure 4.8 that contain similar letters then not all of them may be used to represent new tasks, in which case non-representing circles/nodes are removed from the figure leaving links.

Although the alpha-numeric node labels represent tasks, they are not IDs of the new tasks. The ID of a new task about to be represented in the figure is equal to the current total number of tasks (finished or otherwise) plus one. For example, suppose there are 3 new tasks that appear at the sixth SOSA and there is a total of 34 tasks that existed prior to this SOSA. Based on the previously explained scheme, these new tasks will be represented by the circles labelled “f1”, “f2” and “f3” and will have IDs of 35 = 34 +1, 36 = 35 +1, and 37 = 36 +1 respectively. Note that the letter “f” in the labels corresponds to the sixth SOSA. Considering that the circles/nodes “f4” and “f5” do not represent any task then they will be removed from Figure 4.8 and this will leave links (curved arrows) between tasks 19 and 22, and between tasks 21 and 22 respectively.

### 4.3.2 Resources

The types and the constraints on the number of resources in the MOEs are as follows:

- Light Mortar Batteries ($R_1 \leq 28$)
- Infantry Companies ($R_2 \leq 28$)

The number of newly broken resources of either $R_1$ or $R_2$ type does not exceed four. The changes in resource availability from available or occupied to broken may occur only at any SOSA in the SRO.

Note that the definition given in this sub-section of any problem from $\mathcal{O}^2$ is *not* complete. At any SOSA in the SRO, the number of new tasks and the numbers of newly broken resources of types $R_1$ and $R_2$ are undefined. The additional information to complete this definition is presented in Section 8.3.1.

### 4.3.3 Slight and Extreme Changes

Suppose that at each SOSA in the SRO five new tasks appear and, simultaneously, four items of each of $R_1$ and $R_2$ resource types are newly broken. This type of environmental dynamics is referred to as *extreme change*; the problem $x^2 \in \mathcal{O}^2$ set in this MOE is referred to as the *extreme problem*; and the sub-problem $x_i^2$ set in the $i^{th}$ snapshot of this MOE, where $i > 2$, is referred to as the *extreme sub-problem*. A MOE is considered to undergo a *slight change* when, at each SOSA in the SRO, the number of new tasks, the number of newly broken resources of $R_1$ type and the number of newly broken resources of $R_2$ type are not simultaneously five, four and four, respectively (but as indicated above, range from zero to five, zero to four, and zero to four respectively). Tasks in the MOE, where a problem from $\mathcal{O}^2$ is set, also have a change in duration that follows Equation A.9 with $\delta = 3.0$; and occur from the first to the sixth SOSA of the MOE.

## 4.4 Simulations

The MOE, in which a problem from $\mathcal{L}^2 \cup \mathcal{L}^3 \cup \mathcal{X}^2$ is set, is computer simulated for several times, where $\mathcal{X}^2 \subset \mathcal{O}^2$ and is comprised of DOE-designed problems and $\mathcal{X}^2$ is defined in Section 8.3.5. In each simulation, to reflect a real-life scenario:

1. Resources in the MOE may be moved from one task location to another (e.g. tanks moved from base camp to a valley).

2. Status of tasks and resources (enumerated in Section 4.1.1) in the MOE may be altered (e.g. soldiers killed).

3. Type of dynamics (e.g. TSC labelled $S_2$) of the MOE is implemented.

4. The original (before any change in the MOE) duration of each unfinished task in the MOE is added with a random sample of the model in Equation A.9 with $\delta$ dependent on the type of problem. Consequently, an unfinished task in the $i^{th}$ snapshot (taken immediately after the $i^{th}$ SOSA) of one MOE simulation could differ in the duration of the same task in the $i^{th}$ snapshot of other MOE simulations. For example, if task 43 in the fifth snapshot of the second MOE simulation has a duration of 16 time units its duration can be 25 time units in the fifth (same ordered) snapshot of the ninth MOE simulation. The difference could be true between the same-IDed unfinished tasks in the same-ordered snapshots of different MOE simulations, where the snapshot is of any order from the first to the sixth.

5. Sub-problems of the problem are sequentially solved (e.g. from $p_0^2$ to $p_6^2$) independently by techniques from $\mathcal{T}$. Further, the MOE is simulated at its $i^{th}$ SOSA before the techniques solve a sub-problem of the problem set in the $i^{th}$ snapshot of the MOE, where $1 \leq i \leq 6$. One $i^{th}$ snapshot is taken for each MOE simulation. By the various simulations, a group of different sub-problems are set in the $i^{th}$ snapshots due to the application of Item 4. Despite the difference, techniques from $\mathcal{T}$ independently solve the sub-problems from the same group. This approach reduces

the number of variables to consider for comparing the techniques.

6. The random seed used by the evolutionary processes of techniques from $\mathcal{T}$ to solve a sub-problem of the problem differs across snapshots and simulations of the MOE (where the problem is set).

This simulation is performed for each problem from $\mathcal{L}^2 \cup \mathcal{L}^3 \cup \mathcal{X}^2$. Note that the task duration change in the problem is specified, by the sampling, in Item 4 that makes the problem a problem simulation referred to in Section 4.2.1.

# Techniques from $\mathcal{T}$

# Chapter 5

This chapter describes the principal technique in this thesis, i.e. McBAR, and the other techniques utilised to manifest the characteristics of McBAR. It starts by explaining in Section 5.1 the algorithm of a source technique upon which McBAR is derived. This explanation is followed in Section 5.2 by the demonstration of the way in which the source technique becomes inapplicable to solve the sub-problems (related in Equation 1.5) of problems from $\mathcal{M}$. The adopted remedy for this inapplicability is explained in Section 5.3. Section 5.4 demonstrates the performance degrading effect of this remedy on the source technique. Section 5.5 explains and rationalises a possible strategy to nullify the degradation. Section 5.6 describes the algorithm of McBAR that incorporates the source technique, the inapplicability remedy and the degradation nullification. Section 5.7 explores a slightly different version of McBAR. Then, the adaptation of EDA (described in Section 2.6) to solve the problems from $\mathcal{M}$ is presented in Section 5.8. Section 5.9 enumerates all techniques being compared in the thesis.

## 5.1 CBAR

McBAR is an extension of the memory-based EA technique referred to as *Centroid-Based Adaptation with Random Immigrant* (CBAR). CBAR was applied in [40] to solve problems from $\mathcal{C}^2$ defined in Section 1.2.5, i.e. bi-objective dynamic RCPS problems with a fixed total number of tasks. In this section, the algorithm of CBAR is explained. Further, a MOE where a problem $d^2$ from $\mathcal{C}^2$ is set is simply referred to as the MOE.

Being an implicit memory-based technique (explained in Section 3.3.2), CBAR applies representatives of sets of non-dominated solutions to past sub-problems $d_n^2$ of problem $d^2$ from $\mathfrak{C}^2$ to compute the solutions to a current sub-problem $d_c^2$ of $d^2$, where $t_o(c) \leq n < c$ and $t_o(c)$ is dependent on $c$. Each representative is the centroid of genotypes of non-dominated solutions (phenotypes) to sub-problem $d_n^2$. This centroid $C(t)$ is defined as a genotype whose $k^{th}$ gene is [40],

$$c_k(t) = \left\lfloor \frac{1}{N_d} \sum_{x^j(t) \in Gnds(t)} x_k^j(t) \right\rfloor , \tag{5.1}$$

where $k = 1, \ldots, N$; $N$ is the total number of tasks in the $t^{th}$ snapshot (a static environment) where the sub-problem $d_n^2$ is set; $x^j(t)$ is a genotype of a phenotype/schedule being a solution to the sub-problem; $x_k^j(t)$ is the $k^{th}$ gene/ID in the $x^j(t)$ genotype (defined in Equation 4.1); $Gnds(t)$ is a population of genotypes of all phenotypes/solutions in the non-dominated set $Pnds(t)$ of solutions to the sub-problem; and $\lfloor \bullet \rfloor$ is the operator to round its real-valued argument to integer. Note in Equation 5.1 that the equally-weighted average (mean) of IDs $x_k^j(t)$ forms a centroid gene.

### 5.1.1 Centroid Repair

Centroid $C$ formed through Equation 5.1 may not necessarily be task-precedence-feasible (defined in Section 4.1.4). If this is so, it will be repaired. Before explaining the repair process, a definition is presented. Given a genotype $R$, its complementary genotype $R^c$ is $C$ less the elements of $R$. For example, let the genotype $R = \langle 2, 1 \rangle$ and the centroid,

$$C = \langle 14, 1, 2, 7, 9, 3, 11, \ldots \rangle , \tag{5.2}$$

where numbers to the right of 3 are fixed but not shown for brevity. Thus, $R^c = \langle 14, 7, 9, 3, 11, \ldots \rangle$. The repair of centroid $C$ is undertaken by successively appending (described below) IDs to a genotype $R$ which starts from empty. The ID $\rho$ from $C$ will be

appended to $R$ if it satisfies the following appending rule: the appending of ID $\rho$ should result in a new $R$ whose corresponding complementary genotype $R^c$ has IDs of tasks that are not predecessors of the task with ID $\rho$, and $\rho$ should not be found in the former $R$. However, if $\rho$ does not satisfy the appending rule, a different ID $\rho'$ is randomly picked from $R^c$ and then appended to the former $R$, if it satisfies the appending rule. Note that the complementary genotype used in checking the satisfiability of $\rho'$ to the appending rule is different from $R^c$ where $\rho'$ is picked.

At the start of the repair of centroid $C$, the first element of $C$ is attempted to be appended to an empty genotype $R$ following the appending process described previously. Note that based on the explanation above, the first element or a different element of $C$ may be appended to $R$. Next, the second element of $C$ is attempted to be appended to $R$ (which has one element at this stage). This repair process is continued until the last element of $C$ is attempted to be appended to $R$. After this stage, $R$ is the repaired version of $C$. The following three paragraphs will provide examples of this repair process.

Consider the centroid $C$ in Equation 5.2 whose repair will be based on the PNT illustrated in Figure 2.9. The repair process starts by attempting to append the first gene/ID 14 of $C$ to an empty genotype $R$. So let the new $R = \langle 14 \rangle$ whose complementary genotype $R^c = \langle 1, 2, 7, 9, 3, 11, \ldots \rangle$. However, ID 1 in $R^c$ is the ID of the task that is the predecessor of task with ID 14, based on the PNT. Thus, the appending rule is violated. A random pick of ID, say 1, from $R^c$ is then undertaken. If 1 is appended to the former $R$ (which is empty) the new $R$ becomes $\langle 1 \rangle$ whose complementary genotype $R^c = \langle 14, 2, 7, 9, 3, 11, \ldots \rangle$. Now, no task with ID in the new $R^c$ is a predecessor (based on the PNT) to the task with ID 1. Thus, the appending of 1 to $R$ is allowed, thereby obtaining $R = \langle 1 \rangle$. Notice that based on Section 4.1.4, the obtained $R$ is task-precedence feasible with respect to the PNT.

The next step in the repair process is to attempt to append the second gene/ID 1 of the centroid $C$ to $R$. Note that 1 is already present in $R = \langle 1 \rangle$ whose complementary genotype $R^c = \langle 14, 2, 7, 9, 3, 11, \ldots \rangle$. Thus, based on the above appending rule, a randomly chosen

ID, say 14, is randomly picked from $R^c$. It is then appended to $R$ to obtain $R = \langle 1, 14\rangle$ whose complementary genotype $R^c = \langle 2, 7, 9, 3, 11, \ldots\rangle$. Now, no task with ID in the last $R^c$ is a predecessor to the task with ID 14. Thus, the appending of ID 14 is allowed, thereby obtaining $R = \langle 1, 14\rangle$. Again, notice that this obtained $R$ is task-precedence feasible based on the PNT illustrated in Figure 2.9.

Consider this time the third gene/ID 2 of $C$. When appended to $R = \langle 1, 14\rangle$ it yields $R = \langle 1, 14, 2\rangle$ whose complementary genotype is $R^c = \langle 7, 9, 3, 11, \ldots\rangle$. Now, no task with ID in $R^c$ is a predecessor to the task with ID 2. Thus, the appending of 2 is permitted. Again, notice that the obtained $R = \langle 1, 14, 2\rangle$ is task-precedence feasible based on the PNT illustrated in Figure 2.9. The repair process is continued until the last gene of $C$ is attempted to be appended to $R$. This sample repair process supports the claim that the resulting $R$, at each appending cycle, is task-precedence feasible based on the PNT. It will be proven that after $R$ is completely appended it will be task-precedence feasible.

Consider the expressions of the completely appended $R(t) = \langle r_{1,t}, r_{2,t}, \ldots, r_{N,t}\rangle$ and the centroid $C(t) = \langle c_{1,t}, c_{2,t}, \ldots, c_{N,t}\rangle$, where $N$ is the total number of tasks; and $t$ is a SOSA of the MOE. The element $r_{j,t}$ of $R(t)$ could be regarded as the result of mapping the element $c_{j,t}$ of $C(t)$. Let this mapping be denoted as $\mathfrak{R}$ and referred to as *Random Repairer*. Formally stating the repair,

$$r_j(t) = \begin{cases} c_j(t) & c_j(t) \text{ satisfies append rule} \\ \\ \mathfrak{R}\left(c_j(t)\right) & \text{otherwise,} \end{cases} \tag{5.3}$$

where $c_i(t)$ is defined in Equation 5.1.

### 5.1.2 Initial Population

One sub-algorithm of CBAR is the creation of an initial population which CBAR evolves to obtain solutions to a sub-problem $d_t^2$ of problem $d^2$ from $\mathcal{C}^2$, $t \geq 0$, a problem set in the

$t^{th}$ snapshot of the MOE. For $t = 0$, this initial population is a set of $N$ SSGS-generated (explained in Section 2.3) genotypes. For $t > 0$, it is expressed as,

$$G_0(t) \equiv U(t) \cup Rnd(t) \cup Ichs(t-1), \tag{5.4}$$

where,

1. $U(t)$ is a set of centroids,
$$U(t) = \bigcup_{k=t_o(t)}^{t-1} R_k. \tag{5.5}$$

2. $R_k$ is the repaired centroid of the population $Gnds(k)$ of genotypes of all the non-dominated phenotypes/solutions to the sub-problem $d_k^2$ of problem $d^2$;

3. The subscript of the repaired centroid $R_k$ starts at a value,
$$t_o(t) = max\{t - N_c, 0\}; \tag{5.6}$$

4. $N_c$ is a given maximum number of centroids in any initial population formed by CBAR. Note that, based on the definition of $t_o(t)$, the maximum number of centroids in $U(t)$ is restricted to $N_c$.

5. $Rnd(t)$ is a set of SSGS-generated genotypes where, $|Rnd(t)| = P - |U(t)| - 1$. Section 2.3 explains that SSGS randomly selects IDs of eligible tasks, in a given RCPS environment, to form genotypes. Hence, these genotypes have a stochastic facet. Thus, the SSGS-generated set $Rnd(t)$ constitutes the random component of the initial population $G_0(t)$.

6. $P$ is a fixed size of the initial population;

7. $Ichs(t-1)$ is the chosen genotype from $Gnds(t-1)$ that will be investigated in Section 5.1.6.

### 5.1.3 Genetic Operators

Being an EA-based technique, CBAR possesses an evolutionary process. The crossover and mutation operators in this process are designed to suit the computing system (defined in Section 5.6.1) for this thesis. The designed crossover operator has the following algorithm. Consider two parent genotypes $G_1$ and $G_2$ for crossover from whom two offspring genotypes $O_1$ and $O_2$ are generated. Each of the parent genotypes is broken into three parts (as exemplified in Section 2.1.4.1) at two randomly selected crossing points labelled $L$ and $R$. The crossing point $L$ is between gene locations $l$ and $l+1$ and the crossing point $R$ is between gene locations $r$ and $r+1$, where $1 < l < r < N$ and $N$ is the individual lengths of the parent and offspring genotypes. These crossing points are similar for both parents. First, offspring $O_1$ inherits the first part of parent $G_2$. Second, it inherits the genes of parent $G_1$ and consecutively places these genes at its gene locations $l+1$ to $r$. Suppose the $k^{th}$ gene location of $O_1$ is to be filled with a gene, where $l < k \leq r$. The genes of parent $G_1$, located from one to $r$, are consecutively searched for a gene different to all genes in $O_1$ located before location $k$. Once found, the search is terminated and the different gene is placed at location $k$ of $O_1$. Third, the second step is repeated with genes of $O_1$, located from $r+1$ to $N$, inheriting from $G_2$ where the search process is applied to all genes of parent $G_2$. The three parts of $O_2$ are inherited by consecutively using parents $G_1$, $G_2$ and $G_1$ in the first to the third steps respectively. The presented inheriting process has similarities to the PMX crossover [137]. The mutation of a genotype swaps two of its consecutive genes, at a randomly selected gene location with predefined probability, provided the resulting genotype is task-precedence feasible.

An infeasible or low quality schedule (ordered set) will become feasible or high quality, respectively, by rearranging its sub-sequences. Intuitively, the more sub-sequences are rearranged the better schedule improvement can be obtained. Rearranging three sub-sequences was found in preliminary tests to be best for improvement with small computational effort. Using the definitions of genotypes and schedules in Section 4.1.4, the two-point crossover of two genotypes (of phenotypes/schedules) just presented can be re-

garded as effectively rearranging three sub-sequences of each of the two schedules. Thus, this type of crossover is utilised in the thesis. The schedule improvements will also be attained by changing task order in the schedule. Based on the definitions, the mutation method presented above can change the task order through the swapping of genes. Thus, it is also utilised in the thesis.

### 5.1.4 Schedule Formation

Let us discuss a method, referred to as *Schedule Formation*, to determine a schedule from a genotype. Given the genotype $G = \langle g_1, g_2, \ldots, g_N \rangle$ of length $N$ (described in Section 4.1.4), each of its genes/IDs is used consecutively. At the stage of using the gene/ID $g_k$ in $G$, the starting time $t$ of task with this ID is set to the earliest time later than or equal to the latest end time (starting time plus duration of a task) among its predecessors, i.e.,

$$t = max\{0, max\{st_i + d_i | i \in Pred(g_k)\}\}, \tag{5.7}$$

where $st_i$ and $d_i$ are the starting time and duration of task with ID $i$, respectively; and $Pred(g_k)$ is a set of IDs of tasks preceding the task with ID $g_k$. Further, the starting time is such that there are enough resources for the task with ID $g_k$ to utilise until the completion of this task. After the consecutive usage of genes, the starting times of all tasks with IDs in the genotype are determined. As defined in Section 4.1.4, a schedule is an ordered set of starting times of comprising tasks. Thus, after the usage, the schedule/phenotype of the genotype is obtained. The cost to implement this schedule is determined through Equation A.8 and the schedule makespan is the end time of the finish of the last task. Hereafter, a genotype that corresponds to a schedule signifies that the genotype is used in the Schedule Formation method to obtain the schedule.

### 5.1.5 The Algorithm of CBAR

The static sub-problems $d_i^2$ (set in the $i^{th}$ snapshot of the MOE) of a dynamic problem $d^2$ from $\mathcal{C}^2$ are solved by CBAR sequentially, i.e. from $d_0^2$ to $d_L^2$ where $L$ is the number of state alterations of the MOE. To determine the solutions to the sub-problem $d_0^2$ (set in the original state of the MOE) CBAR simply executes SSGS to generate an initial population $G_0(0)$ of genotypes. It then evolves $G_0(0)$ to obtain the population $G(0)$ of evolved genotypes. Then by applying the Schedule Formation method (described in Section 5.1.4) to the genotypes in $G(0)$, the population $P(0)$ of baseline schedules is then determined. These schedules are the solutions to the $d_0^2$ sub-problem. This and the remaining algorithms of CBAR are expressed as pseudo-code in Figure 5.1.

The evolutionary process executed by CBAR is performed through NSGA-II. In each cycle of this process, NSGA-II creates an offspring genotype population from a parent genotype population. The genetic operators utilised by NSGA-II to create the offspring are those described in Section 5.1.3. The offspring are then feed to the Schedule Formation method to obtain the cost and duration of the schedules that correspond to these genotypes. The Pareto rank and crowding distance (elaborated in Section 2.5.3) of each of these schedules are determined based on the cost and duration of each [55]. Following the explanation in Section 2.5.3, NSGA-II uses the Pareto ranks and crowding distances of the schedules in its selection process to obtain its next generation genotype and schedule populations.

To determine the solutions to the sub-problem $d_t^2$, $t > 0$, CBAR starts by determining the centroid $C(t-1)$ of genotypes which correspond to the non-dominated solutions to the sub-problem $d_{t-1}^2$ (set in the last snapshot before the $t^{th}$ snapshot of the MOE) through Equation 5.1; then it repairs this centroid using $\mathcal{R}$ (as explained in Section 5.1.1), followed by forming an initial population $G_0(t)$ through Equation 5.4; and then it evolves this initial population using NSGA-II as described above. The schedules that correspond to the evolved genotypes are the solutions to the sub-problem $d_t^2$ and form a population $P(t)$.

**Procedure** CBAR
**Begin**
Apply SSGS to generate an initial population $G_0(0)$
Evolve $G_0(0)$ using NSGA-II to obtain the set $G(0)$ of genotypes
Apply Schedule Formation method to $G(0)$ to obtain the set $P(0)$
of solutions to $d_0^2$
**For** $t = 1$ to number of SOSAs
Determine centroid $C(t-1)$ of the set of genotypes of
non-dominated solutions in $G(t-1)$
Repair centroid $C(t-1)$ using $\mathcal{R}$
Form an initial population $G_0(t)$ based on Equation 5.4
Evolve $G_0(t)$ using NSGA-II to obtain the set $G(t)$ of genotypes
Apply Schedule Formation method to $G(t)$ to obtain the set $P(t)$
of solutions to $d_t^2$
**End**
**End**

Figure 5.1: Algorithm of Centroid-Based Adaptation with Random Immigrants (CBAR)

SSGS and NSGA-II are components of CBAR. Thus, from here onwards, CBAR is regarded as that which generates the initial population $G_0(0)$ and the SSGS-generated genotype population $Rnd(t-1)$ in Equation 5.4, at any $t > 0$, by having the SSGS algorithm as the generator of the genotypes in these populations. Further, it is regarded as that which evolves the initial populations described in this section by having NSGA-II as its evolutionary engine.

### 5.1.6 Chosen Schedule

After CBAR has computed the population $P(t)$ of solutions to the $d_t^2$ sub-problem, a schedule $c$ is chosen from $P(t)$. This chosen schedule is utilised in the simulations of the MOE discussed in Section 4.4. At the $(t+1)^{th}$ SOSA of a simulation of the MOE, CBAR evolves an initial population $G_0(t+1)$ to obtain the solutions to the sub-problem $d_{t+1}^2$ of problem $d^2$ (set in the MOE). This initial population contains, based on Equation 5.4, the chosen genotype $Ichs(t)$ that corresponds to the chosen schedule $c \in P(t)$.

Consider the tasks in a schedule $S(t)$ which corresponds to a genotype in $G_0(t+1)$ different to the chosen genotype $Ichs(t)$, tasks whose IDs are similar to those of the tasks

of the chosen schedule $Schs(t)$ (corresponds to $Ichs(t)$) that are on-going or finished at the instant of the $(t+1)^{th}$ SOSA of the MOE. The starting time of each of the tasks in $S(t)$ is set equal to the starting time of its counterpart (task of same ID) in the chosen schedule $Schs(t)$, i.e. copying them from the chosen schedule. This process is performed on all schedules that correspond to genotypes in $G_0(t+1)$ different to $Ichs(t)$; and also on schedules which correspond to offspring genotypes in every cycle of the evolutionary process of CBAR. At the end of this evolutionary process, the schedules which correspond to the evolved genotypes have copies of the on-going or finished tasks of the chosen schedule. Note that these schedules, as solutions to the $d^2_{t+1}$ sub-problem (set in the current state of the MOE), are revisions of the chosen schedule $Schs(t)$ which is one of the solutions to the $d^2_t$ sub-problem (set in the last state of the MOE). Thus, the preservation (copying) of the on-going and finished tasks in the revised schedules demonstrates CBAR will abide with the schedule revision rule described in Section 2.4.2.

## 5.2 Inapplicability of CBAR

Let us now discuss how CBAR becomes inapplicable for solving the dynamic problems from $\mathcal{P}^2$, each involving change in the total number of tasks. Suppose a static sub-problem $p^2_0$ of problem $p^2$ from $\mathcal{P}^2$ is set in the original state of a MOE which has $N$ total number of tasks. For this sub-problem, a sample PNT is depicted in Figure 4.1 where original tasks (the only tasks found in the original state of the MOE) and additional tasks are represented by numbered rectangles and circles respectively.

Now, suppose that the total number of tasks is increased for the first time to $N_\tau = N + dN$ at the $\tau^{th}$ SOSA of a MOE to where the problem $p^2$ is set; $\tau > 0$. When $dN$ or more tasks are finished at the $\tau^{th}$ SOSA of the MOE, IDs of new $dN$ tasks can be placed instead of the IDs of $dN$ finished tasks in each genotype in the initial population $G_0(\tau)$ used to solve through CBAR the sub-problem $p^2_\tau$ of problem $p^2$. Thereby, the need to change genotype length may be avoided. However, in the case where there is no task

finished at the $\tau^{th}$ SOSA of the MOE, the genotype must have a sufficient number of genes to accommodate IDs of $N$ original and $dN$ new tasks. Thus, this condition requires the genotype to be implemented with $N_\tau$ number of genes. Therefore, the non-dominated set of solutions (schedules) to the $p^2_\tau$ sub-problem must correspond to genotypes – that form the set $G(\tau)$ – each of length $N_\tau$.

Before continuing our discussion, let us take the notation,

$$G\,[t] = \bigcup_{k=t_o(t)}^{t-1} G(k), \tag{5.8}$$

to denote the combined genotype populations that correspond to the non-dominated sets of solutions to static sub-problems $p^2_{t_o}$ to $p^2_{t-1}$ of a problem $p^2$ from $\mathcal{P}^2$, where $t_o(t)$ is defined in Equation 5.6. Based on the definition of $t_o(t)$ and on Equation 5.8, the number of genotype populations combined to form $G\,[t]$ is limited at most to $N_c$, the maximum number of centroids used in Equation 5.6.

Suppose there is no finished task prior to the $\tau^{th}$ SOSA of a MOE where the dynamic $p^2$ problem is set, where $0 < \tau < N_c$. Thus, each genotype in the combined populations $G\,[\tau]$ has $N$ number of genes. Based on Equation 5.4, repaired centroids and the chosen genotype in the initial population $G_0(\tau)$ are derived from $G\,[\tau]$. Thus, they each have a length of $N$. Considering that CBAR does not increase the length of each of the genotypes as it evolves them, it cannot determine $G(\tau)$ (whose genotypes must have the length $N_\tau > N$) by evolving $G_0(\tau)$ (whose genotypes have length $N$). Thus, CBAR is unsuitable to determine the solutions to $p^2_\tau$ that correspond to genotypes in $G(\tau)$.

## 5.3 Partial Remedy

Let us now consider how CBAR is revised to overcome its unsuitability to solve the dynamic problems from $\mathcal{P}^2$. Further, let any MOE considered in the remaining portion of this chapter be the one where the problems are set and referred to simply as the MOE.

Now, suppose new tasks (not found in the original state of the MOE) appear for the first time at the $\tau^{th}$ SOSA of the MOE. Following [15], the partial remedy for the unsuitability is to insert new genes, which correspond to the new tasks, to each genotype $G$ in $G[\tau]$ defined in Equation 5.8.

The new gene that corresponds to a new task is inserted to the right of, and as near as possible to, the gene currently in $G$ that corresponds to the immediate predecessor of the new task. If there is more than one immediate predecessor to the new task, the insertion is to the right of and as near as possible to the gene of a randomly chosen immediate predecessor. This gene insertion abides by the gene ordering rule (explained in Section 4.1.4). It is performed for each of the new tasks. Sample gene insertion is illustrated in Figure 5.2 where genes are represented by boxed numbers. Genes/IDs 31 to 33 correspond to the new tasks and the rest of IDs to those of the original tasks. The gene arrangement in this figure abides by the gene ordering rule which uses the PNT illustrated Figure 4.1 with circles that represent tasks 34 to 40 replaced with links.

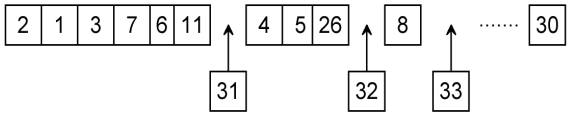



Figure 5.2: Sample gene insertion

The gene-insertion process performed for each population $G(k)$ (defined in Equation 5.8) of $G[\tau]$ is symbolised in Figure 5.3. In this figure, the horizontal line is the SOSA of the MOE. And, the $k^{th}$ SOSA in this line corresponds to the sub-problem $p_k^2$ of the problem $p^2$ from $\mathcal{P}^2$. Each SOSA that is downward-pointed by an arrow corresponds to the sub-problem whose non-dominated solutions (phenotypes) correspond to genotypes being inserted with the new genes; these genotypes are from the population $G(k)$ in $G[\tau]$. Thus, the gene-inserted population $G(k)$ corresponds to the downward arrow pointing to the $k^{th}$ SOSA, for $t_o(\tau) \leq k < \tau$. This explains the presence of the downward arrows pointing toward the $(t_o(\tau))^{th}$ to the $(\tau - 1)^{th}$ SOSAs in the figure.

After the insertion of new genes into each genotype of each population $G(k) \in G[\tau]$, the initial population in Equation 5.4 is formed using the gene-inserted $G(k)$s and then

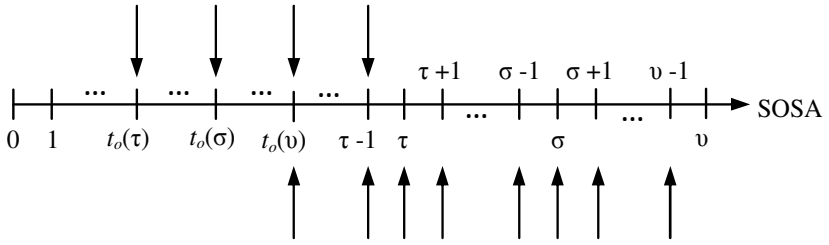



Figure 5.3: The SOSAs of the MOE at which genes are inserted

evolved by CBAR. The evolved genotypes are inputted to the Schedule Formation scheme (explored in Section 5.1.4) to obtain the solutions to sub-problem $p^2_\tau$ of problem $p^2$. The inclusion to CBAR of the just presented gene-insertion process, which is performed every time new tasks appear in the MOE, yields a technique referred to as *Gene-Inserting CBAR* (GIBAR).

## 5.4 Gene-Insertion Side-Effect

Let us now investigate the consequence of the above-explained gene-insertion process on the performance of GIBAR. First, Section 5.4.1 presents a problem whereby the gene-insertion is demonstrated to be the cause of the performance degradation of GIBAR upon the appearance of new tasks. Then the effect of the gene-insertion on the genotypes determined by GIBAR for solving the problem and on the centroid of these genotypes is analytically and empirically investigated in Sections 5.4.2, 5.4.3 and 5.4.5. And using the knowledge gained from these investigations, the performance degradation is intuitively explained in Section 5.4.6. The figure format utilised in Section 5.4.5 is described in Section 5.4.4.

### 5.4.1 Sample case

Consider the application of GIBAR to solve the problem labelled 7 from $\mathcal{L}^2$ (described in Section 4.2.1) that has TSC of $S_1$ based on Table 4.5. The effects of the environmental changes expressed by $S_1$ on the performance of GIBAR for solving the problem

are illustrated in Figure 5.4. The vertical axis of this figure is the hypervolume of the non-dominated set of solutions to sub-problem $7_i^2$ of problem 7, where $i$ is the horizontal axis and the SOSA of the MOE where the problem is set. The hypervolume in this figure increases from the zeroth up to the third SOSA of the MOE and generally decreases thereafter. This decrease started exactly when the total number of tasks in the MOE increases for the first time, an increase that occurs at the fourth SOSA based on Tables 4.2 and 4.3 and on $S_1$. The general decrease in the hypervolume implies the general degradation of the performance of GIBAR, based on Definition 1 in Section 2.5.1. Considering that GIBAR performs gene-insertion when the total number of tasks in the MOE increases and that the performance degradation starts at this increase then the gene-insertion could be the cause of the performance degradation.

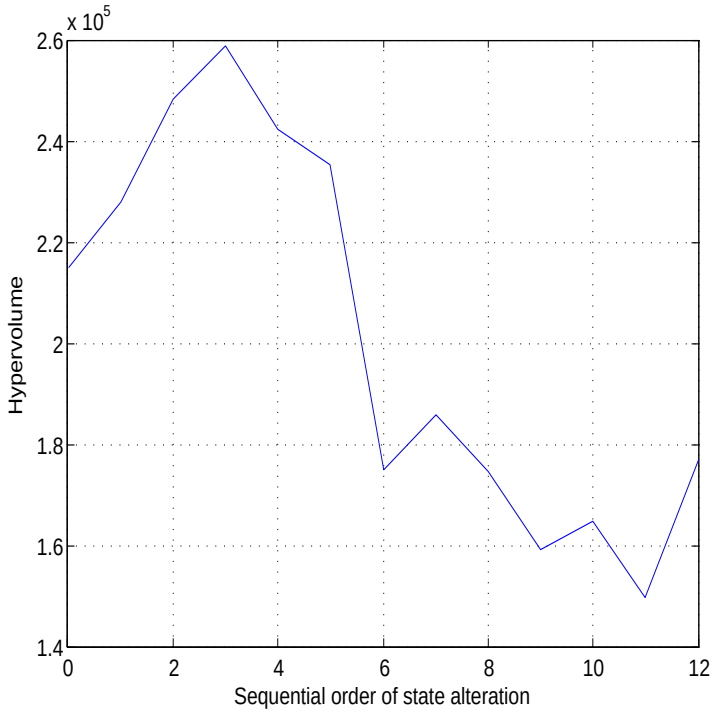



Figure 5.4: Hypervolume of NDS determined by GIBAR

Note that based on Equation 4.5, the number of new tasks that appears at the fourth SOSA is $Nt(7, 4) = 4$, where the function $Nt$ is defined in Equation 4.5. Further, based on the ID assignment rule in Section 4.2, the IDs of the new tasks are 31 to 34.

### 5.4.2 Large ID Discontinuity

Let us now explore the effect of the above-described gene-insertion. It could happen in practice that new tasks (absent in the original state of a MOE) in a MOE cannot be anticipated. Assuming positive integral task IDs, it is then sensible to label original tasks (present in the original state of a MOE) in the MOE with IDs from one to their number $N$ (e.g. 30). In this set-up, the new tasks that correspond to new genes must have IDs (e.g. 31 to 33 in Figure 5.2) greater than $N$. As exemplified in Figure 5.2, the labelling scheme can bring about a large discontinuity of task ID values along a gene-inserted genotype.

### 5.4.3 Analytical Investigation

Let us analytically investigate the centroid of gene-inserted genotypes. Consider a discrete space of genotypes, each being regarded as a point. For example, the genotype $\langle 5, 2, 31, 7, \ldots \rangle$ expresses that its first coordinate has a value (ID) of five. To simplify our discussion, let us consider three-gened genotypes expressed as $\langle \delta_1, \delta_2, \delta_3 \rangle$, where $\delta$s are task IDs. Suppose that a set $G_0$ of genotypes is formed with genes/IDs each allowed only to take a value from the set $L = \{1, 2, 3\}$. All possible genotypes (e.g. $\langle 2, 3, 1 \rangle$) created under this restriction comprise the cluster $C_0$ in Figure 5.5 whose every coordinate axis signifies the task ID at a gene index indicated by the number beside the word "Gene" near the axis. Now, suppose another set $S_1$ of genotypes is generated, genotypes whose second gene is allowed to take a value only from the set $H = \{31, 32, 33\}$ while the rest of the genes are each only allowed to take a value from $L$. Note that the second gene has IDs related to those of the new genes considered in Section 5.4.1. The set $S_1$ constitute the cluster $C_1$ of genotypes in the figure. Along the genes in each genotype (e.g. in $\langle 1, 31, 2 \rangle$) in the set $S_1$ is a large discontinuity in IDs. The other sets $S_2$ and $S_3$ of genotypes that constitute clusters $C_2$ and $C_3$ in the figure are configured through a scheme in an analogous manner to that of $C_1$. It will be shown that the large discontinuity of IDs in the genotypes from $S_1$ to $S_3$ is responsible for the existence of the clusters $C_1$ to $C_3$. Further,

the union of clusters $C_0$ to $C_3$ has a centroid, indicated by the solid rectangle in the figure, that is far from each of them. Note that if the clusters are absent, only cluster $C_0$ is present and its centroid would be near to itself. Thus, the presence of the clusters $C_1$ to $C_3$ causes the existence of the far centroid. Considering that the presence of clusters $C_1$ to $C_3$ is due to the large discontinuity, then the presence of the far centroid is due to the large discontinuity. If the genes from $H$, which caused the large discontinuity, are regarded as inserted genes then the existence of the far centroid is due to the inserted genes.

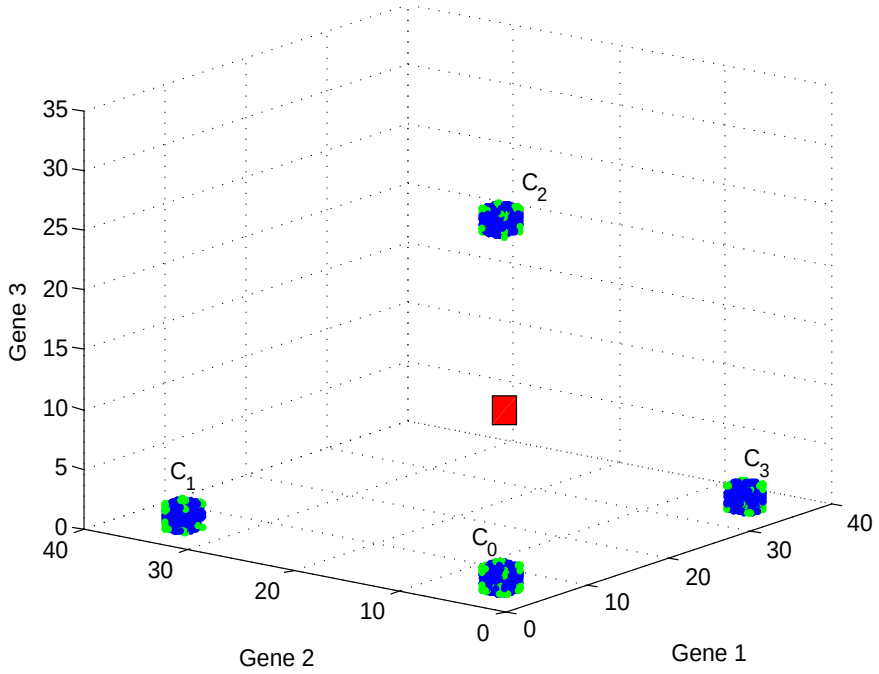



Figure 5.5: Clusters of genotypes, some with abrupt changes in IDs

### 5.4.4 Cluster Representation

To prepare for the discussion in the next sub-section, let us translate the presentation of cluster $C_1$ in Figure 5.5 to that in Figure 5.6(a) whose format is similar to that of Figure 2.17. Each of the gene indexes $k$ at the horizontal axis of Figure 5.6(a) corresponds to a dimension in the genotype space represented in Figure 5.5, the dimension labelled as "Gene $k$". Following the construction of Figure 2.17, the genotypes with ID $i$ at gene index $k$ that comprised cluster $C_1$ are counted. Then, the percentage of this count of

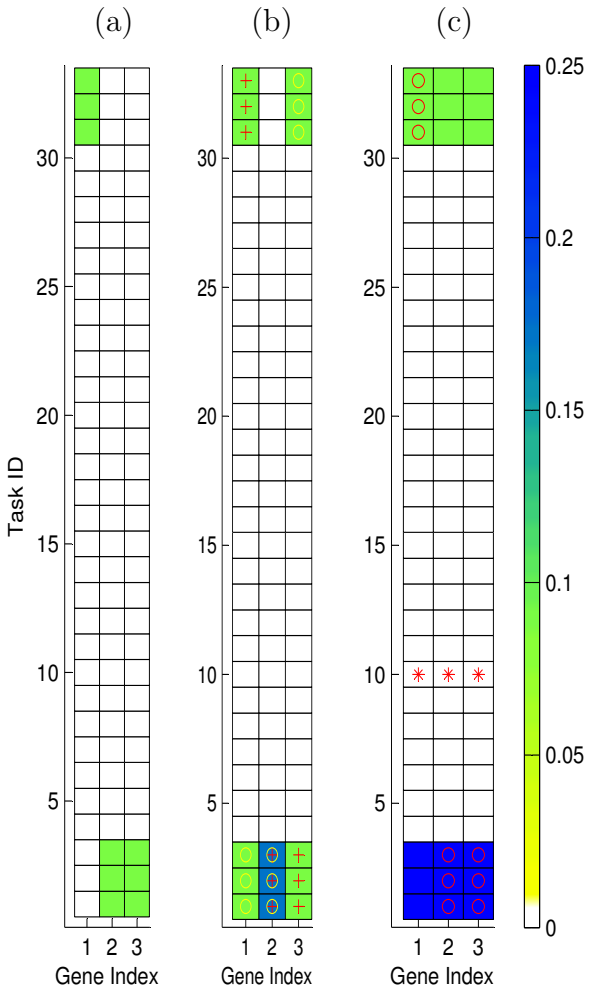



Figure 5.6: Cluster representation in distribution format of cluster/s (a) $C_1$ (b) $C_1$ and $C_3$ (c) $C_0$ to $C_3$

the total number of genotypes utilised to form Figure 5.5 is represented by a colour of the square located at $(k,i)$ in Figure 5.6(a). The colour scale beside Figure 5.6(c) reveals that the nearer the colour to blue the more popular an ID $i$ is; the ID at gene index $k$ of the genotypes utilised to form Figure 5.5. Consider only one cluster in Figure 5.5 that is represented in a figure X with the same format as that of Figure 5.6(a). The popularity of this cluster is defined as the average percentage count (nearness to blue of the colour) of all squares in figure X.

The clusters $C_1$ and $C_3$ illustrated in Figure 5.5 are translated in Figure 5.6(b) to coloured squares that are discriminated by "+" and "o" marks respectively. In a figure format such as Figures 5.6(a) to (c), the set of squares with the same mark is defined as

a cluster. Notice that some squares have double marks, which denote that the percentage count at each of these squares is the sum of percentage counts (in this case for Figure 5.6(b), half for cluster $C_1$ and half for cluster $C_3$) in one-to-one correspondence with the clusters. Further, all clusters have equal popularities.

Let the ID coordinate of a symbol "*" in Figure 5.6(c) be the mean of IDs rounded to the nearest integer, the IDs at gene index equal to the index coordinate of this symbol. And, suppose that the ID coordinate of a "*" is copied to a location in a genotype at index equal to the index coordinate of the "*". If this process is performed on all "*"s, it will be shown that the genotype formed is equivalent to the centroid (red rectangle) of all the clusters in Figure 5.5. In a given figure with the same format found in Figures 5.6(a) to (c), the set (genotype) of rounded means ordered according to their corresponding gene indexes is defined as a centroid of the distribution illustrated in the given figure. We then say that the set of "*"s correspond to a centroid in Figure 5.6(c). Notice that this centroid is far from the cluster ($C_1$) marked by "o"s in Figure 5.6(c).

### 5.4.5 Empirical Investigation

In Section 5.4.3, we concluded that the insertion of new genes caused the existence of the far centroid. This analytically-derived conclusion is based on three-dimensional genotype space. Let us now investigate, through empirically obtained genotypes, the distance of the centroid of the clusters derived from these genotypes to these clusters set in a high-dimensional genotype space. Considering that each gene index in Figure 5.7 corresponds to a dimension (as in Figures 5.6(a) to (c)), Figure 5.7 illustrates a high-dimensional distribution. This distribution $D$ is that of IDs/genes in genotypes that correspond to solutions (schedules) determined by GIBAR for solving the sub-problem $1_1^2$ of problem $1^2$ from $\mathcal{L}^2$ (notations are based from Section 4.2.1). Using Bayesian Information Criteria [182], the distribution $D$ can be shown to be composed of several clusters (akin to those depicted in Figure 5.6(b)) with possibly different levels of popularity. Let one of these clusters be labelled as $H$ and be represented (through the scheme explained in Section

5.4.4) by the set of "o"-marked squares (akin to that in Figure 5.6(b)) in Figure 5.7, each having the highest percentage count among squares of the same gene index coordinate, i.e. cluster $H$ is highly popular. Note that in Figure 5.7, the centroid (corresponds to the set of "+"s akin to "*"s in Figure 5.6(c)) of the distribution $D$ is close to cluster $H$. Further, the distribution $D$ corresponds to the sub-problem $1_1^2$ which, by using Tables 4.2, 4.3 and 4.5, can be shown to be set in a MOE that had not undergone an increase in the total number of tasks at the first SOSA (denoted by the subscript in $1_1^2$).

Consider now Figure 5.8 which has a similar format to that of Figure 5.7. It illustrates the task ID distribution of genotypes that correspond to the solutions (schedules) obtained through GIBAR for solving the sub-problem $1_{11}^2$ of problem $1^2$ from $\mathcal{L}^2$. Based on the sub-problem notation and Tables 4.2, 4.3 and 4.5, this sub-problem is set in a MOE that had undergone increases in the total number of tasks at the fourth, sixth, $10^{th}$ and $11^{th}$ SOSAs. By these increases, several new genes are inserted by GIBAR into the genotypes of a baseline solution to sub-problem $1_0^2$ set in the original (zeroth) state of the MOE. Among the squares at each gene index in Figure 5.8, the square (denoted by "o") with the highest percentage count is generally far from the rounded mean (denoted by "+"). Thus, following the definition in Section 5.4.4, the centroid – corresponding to the set of "+"s – is far from the highly popular cluster marked by the set of "o"s in the distribution. Considering that the centroids illustrated in Figures 5.7 and 5.8 are, respectively, near to and far from their respective highly popular clusters and that the figures correspond to the non-insertion and insertion of new genes, respectively, then the far centroid in Figure 5.8 will be due to the insertion of the new genes. Therefore, based on either the empirically-based analysis in this section or on the analytically-based analysis of Section 5.4.3, the insertion of new genes by GIBAR to genotypes can cause the centroid of the distribution of IDs from these genotypes to become far from the highly popular clusters in this distribution.

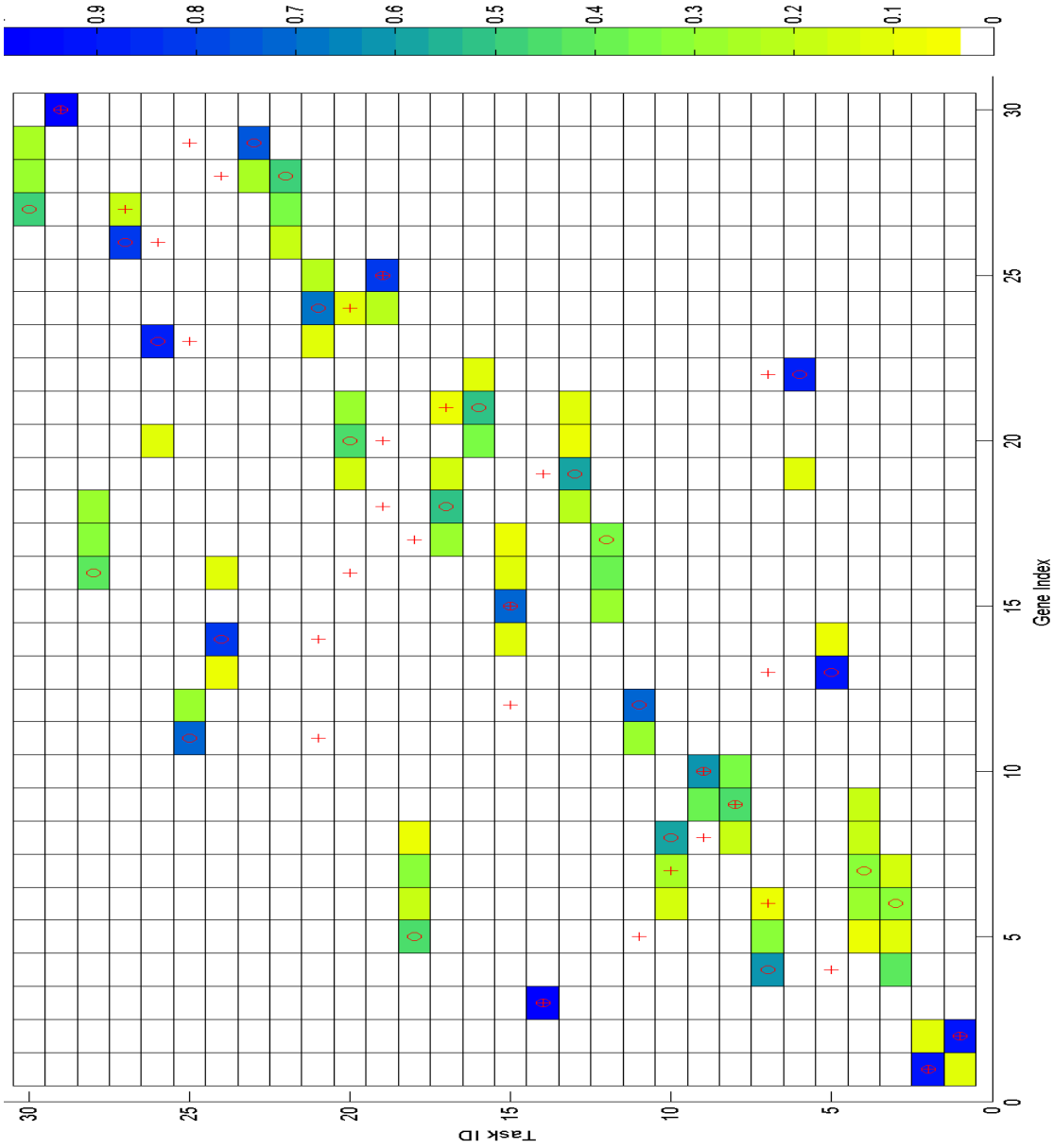



Figure 5.7: Distribution of IDs in genotypes of phenotypes/solutions to sub-problem $1_1^2$ obtained by GIBAR

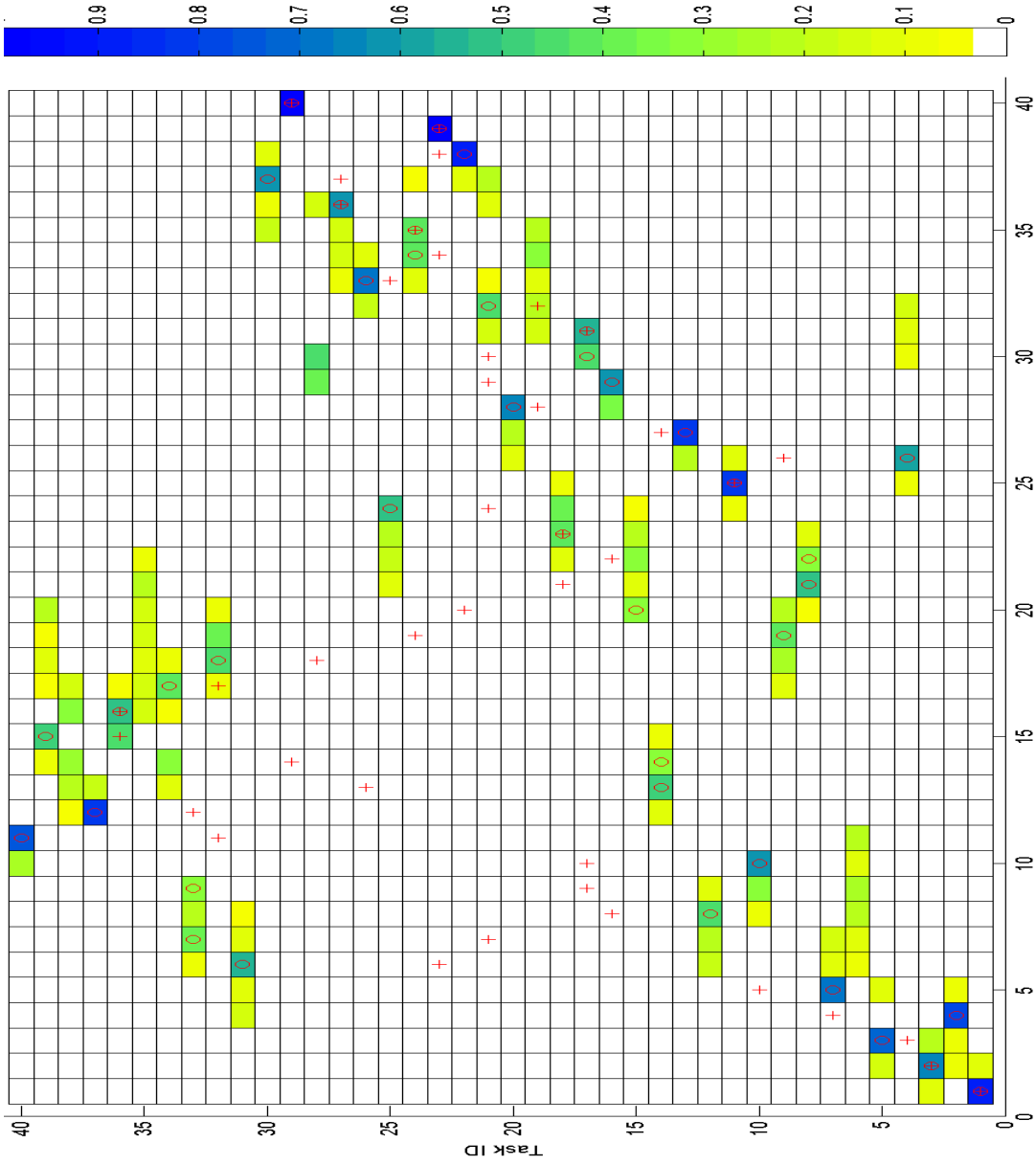



Figure 5.8: Distribution of IDs in genotypes of phenotypes/solutions to sub-problem $1^2_{11}$ obtained by GIBAR

### 5.4.6 Intuitive Explanation

Let us intuitively explain, using a particular case, the performance degradation of GIBAR. First, suppose that (a) there is no new task that appears from the first to the sixth SOSA of a MOE; (b) the set $S_0$ of genotypes which correspond to the non-dominated solutions obtained by GIBAR for solving sub-problem $q_0^2$ set in the original state (zeroth snapshot) of the MOE are those depicted in Figure 5.9 as points (dots) on the contour of a fitness landscape, points that are clustered around the label "0" whose location is also that of the centroid of the genotypes; (c) similar depiction and centroid labelling schemes are applied for the sets of genotypes that correspond to the NDSs of solutions obtained by GIBAR for solving sub-problems $q_1^2$ to $q_5^2$ set in the first to the fifth snapshot of the MOE respectively; (d) and the first and sixth snapshots of the MOE – where sub-problems $q_1^2$ and $q_6^2$ are set respectively – are nearly similar. Now, consider the expected set $S_6$ of genotypes that correspond to the non-dominated solutions which will be determined by GIBAR if GIBAR will solve sub-problem $q_6^2$ set in the sixth snapshot of the MOE. By supposition (d), they can be clustered near to $S_1$ ( the set of genotypes that correspond to the solutions to sub-problem $q_1^2$), such that, the centroids of these two clusters can be near (the distance between $S_1$ and $S_6$ can be measured using Hausdorff distance [57, 60]), as exemplified in Figure 5.9.

Second, consider the event where new tasks appear for the first time at the sixth snapshot of the recently considered MOE. Based on the investigations in Sections 5.4.2, 5.4.3 and 5.4.5, the set $T_6$ of genotypes which correspond to the non-dominated solutions to sub-problem $q_6^2$ that are expected to be determined by GIBAR will be those depicted in Figure 5.10, genotypes that form three clusters whose centroid is located at label "6". Considering that there is no new task that appears from the first to the fifth SOSA of the MOE, the sets $S_k$, $0 \leq k < 6$, of genotypes that correspond to the non-dominated solutions to sub-problems $q_0^2$ to $q_5^2$, each may form only one cluster.

Now, GIBAR inserts new genes, that correspond to the new tasks, to each genotype in $S[6] = \cup_{k=0}^{5} S_k$; supposing $N_c = 10$ in Equation 5.8. Using the result of the investigations

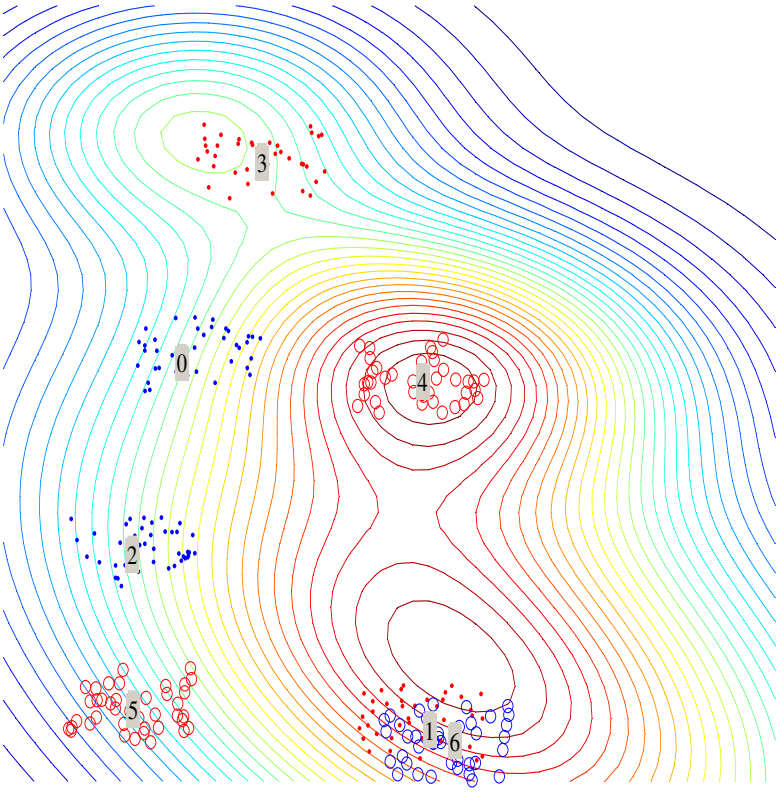



Figure 5.9: Clusters of solutions to various sub-problems without new tasks

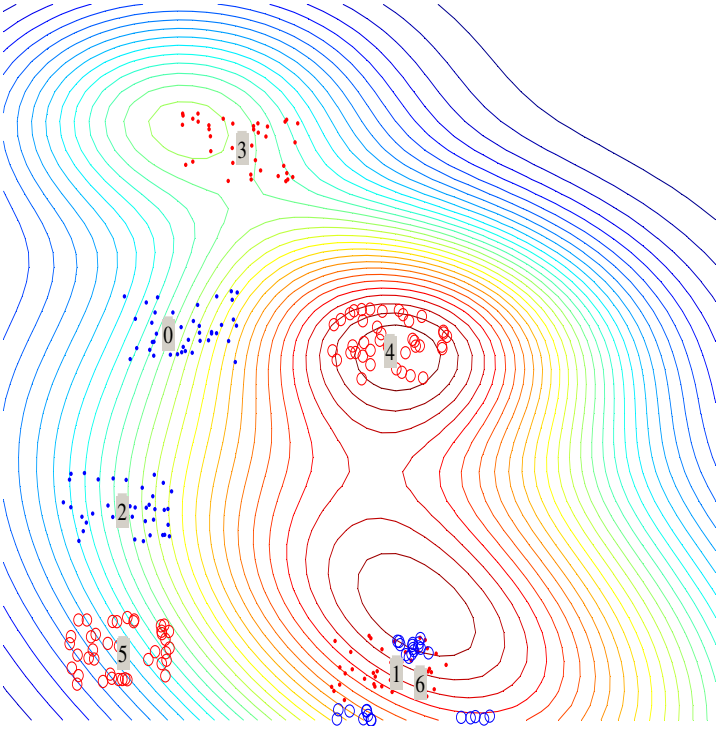



Figure 5.10: Clusters of solutions to various sub-problems with one sub-problem set in the sixth snapshot with new tasks

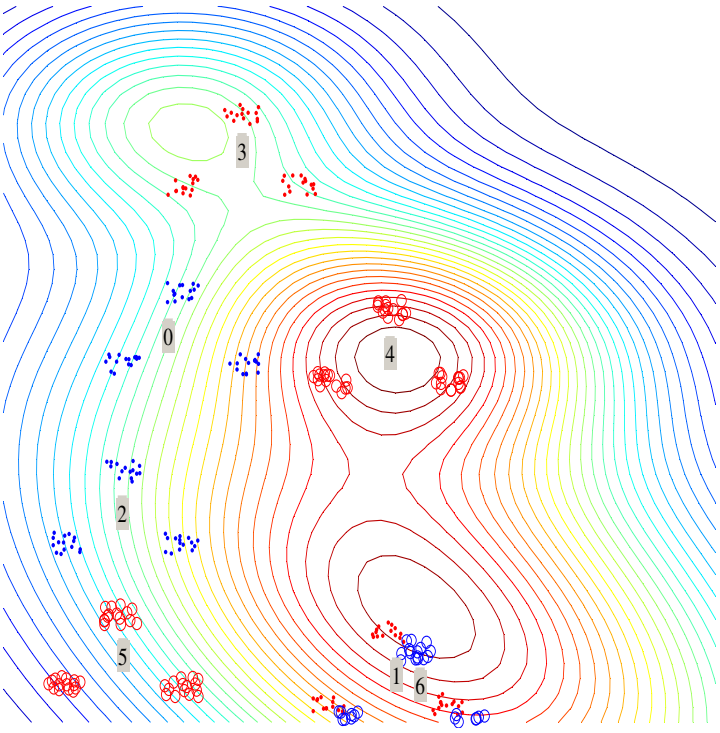



Figure 5.11: Clusters of solutions to various sub-problems with gene-inserted corresponding genotypes

in Sections 5.4.2, 5.4.3 and 5.4.5, a set $T_0$ comprised of the gene-inserted genotypes from $S_0 \in S\,[6]$ could form three clusters such as those depicted in Figure 5.11 where the centroid of these clusters is far from any of them. A similar scenario can be observed in the sets $T_1$ to $T_5$ comprised of gene-inserted genotypes from $S_1$ to $S_5$ respectively.

Based on Equations 5.4 and 5.5, GIBAR prepares an initial population which contains the centroids, $R_0$ to $R_5$, of the sets $T_0$ to $T_5$ (defined above) of the gene-inserted genotypes from the sets $S_0$ to $S_5$ respectively. This initial population is used by GIBAR and is intended to determine the set $T_6$ of genotypes that correspond to the NDS of high-quality solutions to sub-problem $q_6^2$ set in a snapshot with new tasks. Based on the investigations in Sections 5.4.2, 5.4.3 and 5.4.5, the centroids $R_0$ to $R_5$ could be far from the centroid of $T_6$, as depicted in Figure 5.11.

Now, given a fixed number $E$ of evolutionary cycles, GIBAR could be unable to evolve the initial population to produce genotypes which are near to any genotype in $T_6$ that corresponds to an expected high-quality solution to sub-problem $q_6^2$. Thus, GIBAR could

only produce low-quality solutions when solving sub-problem $q_6^2$ set in a snapshot with new tasks. However, based on the investigations in Sections 5.4.2, 5.4.3 and 5.4.5, if there is no new task that appears at and prior to the sixth SOSA of the MOE, the centroid of genotypes from $S_1$ could be near to that of $S_6$, as exemplified in Figure 5.9. Consequently, GIBAR with $E$ number of evolutionary cycles could evolve the initial population with this near (instead of the far) centroid to produce solutions near to $S_6$, i.e. high-quality solutions. Thus, the performance of GIBAR for solving a sub-problem set in a snapshot in the MOE with new tasks will be lower than that of the snapshot without current and previous new tasks.

In the researches of [27, 28, 127, 164], gene insertion was shown to be beneficial for searching high quality solutions to some problems. However, an implicit memory-based approach was not applied in these researches, such that the benefit does not necessarily hold in the performance of GIBAR for solving problems from $\mathcal{P}^2$ that have time-varying total number of tasks. The resolution to the performance degradation of GIBAR will be explained in the next section.

## 5.5 Resolution of Effects

Before proceeding to discuss the approach that can possibly resolve the above-explained gene insertion side-effect, let us define the precedence order of a task. This order is the maximum number of directed links which connect a task to the start of a given PNT following the reversed direction of the links. For example, task 16 in Figure 4.1 is of seventh precedence order.

Consider a set of gene-inserted genotypes that correspond to GIBAR-determined non-dominated solutions to the sub-problem that is set in a snapshot with new tasks. As investigated in Section 5.4, the abrupt change of ID along each element of the set of genotypes could be the cause of the performance degradation of GIBAR. Thus, the chosen potentially-resolving approach for the degradation is to one-to-one map IDs of tasks – that

belong to a similar precedence order – to values that are as near to each other as possible and unique among mapped values of all IDs. Let the function $\mathcal{F}(I_d)$ represents this ID-mapping operation, where $I_d$ is the ID to be mapped. For example, second precedence ordered tasks in Figure 4.1, such as 4, 6, 7, 8, 9, 10, 12, 14 and 38 are mapped to themselves, except task 38 mapped to 15.

As will be explained in Section 5.6, the mapping operation $\mathcal{F}$ and GIBAR are components of McBAR. To manifest a sample effect of $\mathcal{F}$, McBAR is applied to solve the sub-problem $1^2_{11}$ of problem $1^2$ from $\mathcal{L}^2$ (defined in Section 4.2). The distribution of IDs in the elements of the set $G$ of genotypes which correspond to non-dominated McBAR-determined solutions to sub-problem $1^2_{11}$ is illustrated in Figure 5.12, which is of similar format as that of Figure 5.8 (described in Section 5.4). Figure 5.12 illustrates the rounded mean – represented by the "+"-marked square – to be near to the highest percentage count ID – represented by the "o"-marked square – at any gene index, in general. Thus, by the definition in Section 5.4.5, the centroid of the distribution is near to the highly popular cluster. In contrast, the rounded mean at any given gene index in Figure 5.8 is generally far from the high percentage count ID at the same index. Thus, the centroid of the distribution illustrated in Figure 5.8 is far from the highly popular cluster. Note that the distribution illustrated in Figure 5.8 is that of IDs in genotypes which correspond to GIBAR-determined non-dominated solutions to the same sub-problem from where the distribution in Figure 5.12 is derived. These empirical results show that the action of the mapping $\mathcal{F}$ of IDs (which resulted in the distribution in Fig. 5.11) can move a centroid closer to a highly popular cluster in the same distribution. This action is the reverse of the gene-insertion side-effect (investigated in Section 5.4) which can degrade the performance of GIBAR. Thus, it could be expected that McBAR can resolve this performance degradation, an expectation that will be fulfilled in Chapter 6.

Note that although a task ID may be transformed by $\mathcal{F}$, the task that it represents remains the same. For example, the starting time of and the type of resource utilised by a task are not affected by $\mathcal{F}$, except the ID of the task utilised by the resource is also

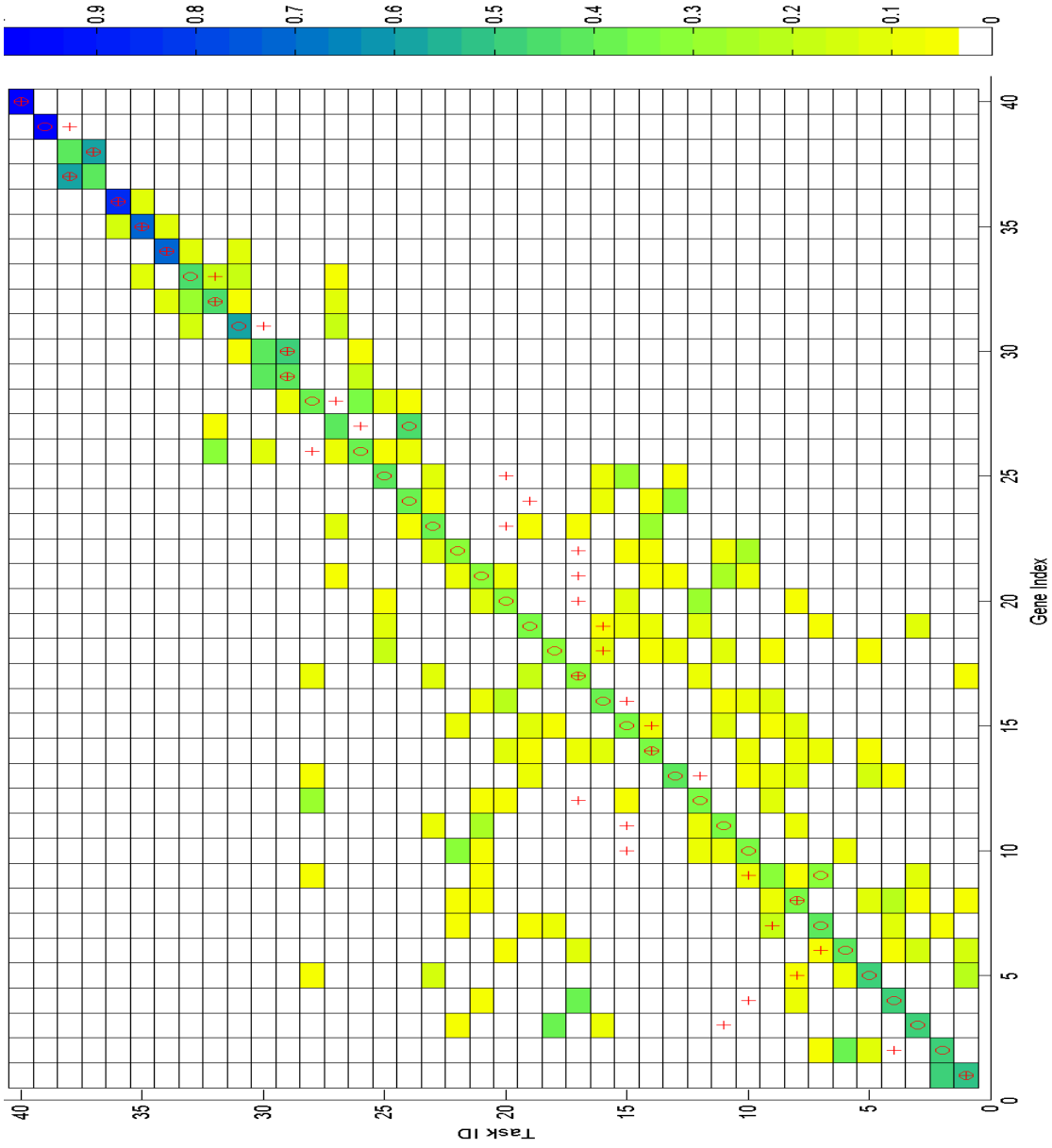



Figure 5.12: Distribution of mapped IDs in genotypes of phenotypes/solutions to sub-problem $1^2_{11}$ obtained by McBAR

mapped by $\mathcal{F}$. Further, the node IDs in the PNT are also mapped by $\mathcal{F}$.

## 5.6 Algorithm of McBAR

This section elaborates on the algorithm of McBAR. First, the computing system utilised by McBAR is described in Section 5.6.1. Then Section 5.6.2 explains the algorithm of McBAR for determining solutions to a sub-problem set in a snapshot where new tasks appear for the first time. Section 5.6.3 investigates the algorithm of McBAR for determining solutions to sub-problems set in snapshots with no new task but succeeding the snapshot where new tasks appear for the first time. Section 5.6.4 describes the algorithm of McBAR for determining solutions to sub-problems set in snapshots where new tasks appear for the second or later times. Sections 5.6.5 and 5.6.6 describe other components of McBAR. And then, the algorithm of McBAR is summarised in Section 5.6.7.

### 5.6.1 Computing System

Let $\mathcal{S}$ be the computing system used in the thesis to solve sub-problems of problems from $\mathcal{M}$. One attribute of any resource in the MOE considered in the thesis is the ID of task which utilises it. Thus, in system $\mathcal{S}$, task IDs are part of the information about (a) the resources, (b) genotypes used in the evolutionary process of McBAR to solve the sub-problems, and (c) each PNT in the snapshots where the sub-problems are set. Further, the PNT in the last snapshot is simply copied to the current snapshot if no new task appears at the current snapshot.

McBAR inherits from GIBAR. The insertion by McBAR of new genes into genotypes is accompanied by the insertion of new nodes, with IDs of the new tasks, into the PNT in the snapshot with new tasks. And, the mapping $\mathcal{F}$ of task IDs in each gene-inserted genotype is accompanied by the mapping $\mathcal{F}$ of task IDs in the node-inserted PNT and in the resources utilised by the tasks. When all IDs in system $\mathcal{S}$ are transformed by the

mapping, system $\mathcal{S}$ is considered to be in mapped mode.

### 5.6.2 First Appearance of New Tasks

Suppose McBAR is utilised to solve the sub-problems $p_k^2$s of problem $p^2$ from $\mathcal{P}^2$, for all $t_o(\tau) \leq k < \tau$; the sub-problems are set in the snapshots taken prior to the $\tau^{th}$ SOSA of a MOE. Further, no new tasks appear prior to the $\tau^{th}$ SOSA. In this case, McBAR simply applies GIBAR to determine the non-dominated solutions to the sub-problems $p_k^2$s, for all $0 \leq k < \tau$. Note that $t_o(t) \geq 0$ based on Equation 5.6. Suppose the set $G(k)$ of genotypes correspond to the non-dominated solutions to the sub-problems $p_k^2$s. Thus, after solving $p_{\tau-1}^2$, $G[\tau]$ (defined as a set of $G(k)$s in Equation 5.8) can be determined.

Now, when new tasks appear at the $\tau^{th}$ SOSA of the MOE, new genes which correspond to the new tasks are inserted by McBAR into each genotype from $G[\tau]$ to obtain $A[\tau] = \cup_{k=t_o(\tau)}^{\tau-1} A(k)$ where $A(k)$ is comprised of the gene-inserted genotypes derived from $G(k)$, for all $t_o(\tau) \leq k < \tau$. New nodes which correspond to the new tasks are also inserted to the PNT that exists prior to any change in the MOE. Next, the system $\mathcal{S}$ is set to mapped mode, i.e. the mapping function $\mathcal{F}$ is applied to every task ID in all in-use resources in the MOE, in all nodes of the gene-inserted PNT and in each gene-inserted genotype in $A[\tau]$, to obtain,

$$B[\tau] = \bigcup_{k=t_o(\tau)}^{\tau-1} B(k). \tag{5.9}$$

where $B(k)$ contains ID-mapped gene-inserted genotypes derived from $A(k) \subset A[\tau]$. Then, the centroid of each $B(k) \subset B[\tau]$ is computed and repaired. However, the centroid repairer $\mathcal{R}$ inherited by McBAR from GIBAR is revised (described in Section 5.6.5), intending to further increase the performance of McBAR. An initial population is then formed (explained in Section 5.6.6), comprised of the repaired centroids (derived from ID-mapped gene-inserted genotypes from $B(k) \subset B[\tau]$, for all $t_o(\tau) \leq k < \tau$), other ID-mapped elements of $B(\tau - 1)$ (found in $B[\tau]$), and SSGS-generated genotypes. Based on the SSGS algorithm explained in Section 2.3, the generated genotypes have mapped

IDs due to SSGS using the node-inserted ID-mapped PNT in the $(\tau-1)^{th}$ snapshot of the MOE. Note that all genotypes of the initial population have mapped IDs.

Under system $\mathcal{S}$ in mapped-mode, in each cycle of the evolutionary process of McBAR the ID-mapped evolutionary operator-produced genotypes which originated from the initial population are inputted to the Schedule Formation method (described in Section 5.1.4). Further, the mapped tasks IDs in resources and the ID-mapped PNT in the $(\tau-1)^{th}$ snapshot of the MOE, and the inputted ID-mapped genotypes are then utilised by the method to obtain schedules with mapped IDs. Thus, at the end of the evolutionary process, the evolved genotypes and their corresponding phenotypes/schedules have mapped IDs. Because of these mapped IDs, the schedules are not as yet the solutions to sub-problem $p_\tau^2$ set in the snapshot with the new tasks.

Let $B(\tau)$ be a subset of the evolved ID-mapped genotypes which correspond to the non-dominated schedules. Now, copies of the ID-mapped genotypes in $B(\tau)$ and the ID-mapped PNT are made. Then, IDs in each of the copies are unmapped. The ID-unmapped genotypes derived from $B(\tau)$ are processed in the Schedule Formation method which utilises the ID-unmapped PNT copy. The processing yields the non-dominated solutions to sub-problem $p_\tau^2$ set in the $\tau^{th}$ snapshot where the first new tasks appear. The series of actions of copying, unmapping and processing in Schedule Formation method is referred to as *Solution Production*.

The Solution Production is only performed when required, for example, for a decision-maker to view the solutions (schedules) to sub-problem $p_\tau^2$. However, the evolved ID-mapped genotypes in $B(\tau)$ are utilised for finding solutions to sub-problems set in the snapshots taken after the $\tau^{th}$ SOSA of the MOE.

### 5.6.3 No New Task in Succeeding Number of Snapshots

Suppose no new tasks appears from the $(\tau+1)^{th}$ to the $(\eta-1)^{th}$ SOSA of the last-mentioned MOE; and the second batch of new tasks occurs at the $\eta^{th}$ SOSA of the MOE.

As supported in Section 5.3, the gene-insertion is performed only due to the occurrence of new tasks. Thus, there is no need to insert genes into each genotype in the populations $B(\tau+1)$ to $B(\eta-1)$ which correspond, respectively, to the non-dominated sets of solutions to the $p^2_{\tau+1}$ to $p^2_{\eta-1}$ sub-problems of problem $p^2$ from $\mathcal{P}^2 \subset \mathcal{M}$. The non-insertion of genes is depicted in Figure 5.3 by the absence of downward-pointing arrows at the $(\tau+1)^{th}$ to the $(\eta-1)^{th}$ SOSAs of the MOE.

Now, although the genotypes in the population $B(\tau)$ of genotypes, which correspond to the non-dominated solutions to sub-problem $p^2_\tau$, are evolutionarily derived from the initial population with individuals derived from the gene-inserted genotype populations in Equation 5.9, they had not undergone the gene-insertion process. This non-insertion is the consequence of the rule explained in Section 5.3 when no new task appears from the $(\tau+1)^{th}$ to the $(\eta-1)^{th}$ SOSAs. Thus, there is no downward-pointing arrow at $\tau^{th}$ SOSA in the figure.

To solve the sub-problem $p^2_\sigma$, where $\tau < \sigma < \eta$, an initial population is formed using the populations in,

$$B\left[\sigma\right] = \bigcup_{k=t_o(\sigma)}^{\sigma-1} B(k), \tag{5.10}$$

where $t_o(\sigma)$ is defined in Equation 5.6; in accordance with Figure 5.3, $B(k)$ is a population of evolved ID-mapped gene-inserted genotypes – if $t_o(\sigma) \leq k < \tau$ – or a population of evolved ID-mapped un-inserted genotypes – if $\tau < k < \sigma$; and $B(k)$ is the population of genotypes that correspond to non-dominated solutions (schedules) to the sub-problem $p^2_k$, for all $t_o(\sigma) \leq k < \sigma$. Note that $B\left[\sigma\right]$ is a mix of populations of inserted and un-inserted genotypes. Next, the initial population with elements derived from $B\left[\sigma\right]$ is evolved by McBAR to obtain a population $E(\sigma)$ of evolved genotypes, where $B(\sigma) \subseteq E(\sigma)$ is a set of evolved genotypes which, when acted upon by the Solution Production, will yield all the non-dominated solutions to the sub-problem $p^2_\sigma$. Note that system $\mathcal{S}$ is at mapped mode during this evolutionary process.

### 5.6.4 Other Batches of New Tasks

Let us now consider the $\eta^{th}$ SOSA of the last-considered MOE when the second batch of new tasks appears. Let $B[\eta]$ be a set of populations $B(k)$, where $t_o(\eta) \leq k < \eta$, and each contains either ID-mapped gene-inserted genotypes or ID-mapped gene un-inserted genotypes, as was the case in Equation 5.10. Further, $B(k)$ is a set of genotypes which correspond to non-dominated solutions to sub-problem $p_k^2$. To solve sub-problem $p_\eta^2$ set in the $\eta^{th}$ snapshot of the MOE, first, the ID-mapped genotypes in $B[\eta]$ are unmapped to obtain a collection $D[\eta]$ of populations $D(k)$, $t_o(\eta) \leq k < \eta$, of ID-unmapped genotypes. Note that, as implied in Section 5.3, there is no unmapping of IDs in genotypes of $B[\eta]$ during the search for solutions to sub-problems $p_{\tau+1}^2$ to $p_{\eta-1}^2$, such that, the genotypes in $B[\eta]$ are mapped. The ID unmapping is also applied to task IDs in resources in the MOE and in the PNT in the $(\eta-1)^{th}$ snapshot of the MOE. Thus, system $\mathcal{S}$ is restored to unmapped mode. Second, new genes that correspond to the second batch of new tasks are inserted into the unmapped genotypes in $D[\eta]$ to obtain a collection $E[\eta]$ of populations $E(k)$, $t_o(\eta) \leq k < \eta$, of ID-unmapped gene-inserted genotypes. This second batch of gene insertion is depicted in Figure 5.3 as upward-pointing arrows. New nodes, which correspond to the new tasks, are inserted into the ID-unmapped PNT. Third, a different mapping function is applied to IDs in each genotype in $E[\eta]$ to obtain,

$$F[\eta] = \bigcup_{k=t_o(\eta)}^{\eta-1} F(k), \tag{5.11}$$

which is a collection of populations $F(k)$, $t_o(\eta) \leq k < \eta$, of ID-remapped gene-inserted genotypes. The different mapping function is still intended to minimise ID discontinuity along involved genotypes. This mapping is also applied to the task IDs in the resources and in the ID-unmapped node-inserted PNT. Thus, system $\mathcal{S}$ is restored to mapped mode once again. Fourth, the collection $F[\eta]$ of population of ID-remapped genotypes is used to form an initial population of ID-mapped genotypes which are then evolved by McBAR to obtain the population $C(\eta)$ of evolved ID-mapped genotypes. Note again that system

$\mathcal{S}$ is in mapped mode during the evolution of the initial population. Fifth, when required, the evolved genotypes in $C(\eta)$ are used in the Solution Production to obtain the non-dominated solutions (schedules) to the $p_\eta^2$ sub-problem. The successive processes of (a) restoring system $\mathcal{S}$ to unmapped mode; (b) insertion of new genes which correspond to new tasks; and (c) reverting $\mathcal{S}$ to mapped mode, are repeated every time a batch of new tasks appear in the MOE after the first batch.

Based on the discussion until this point, system $\mathcal{S}$ is always in mapped mode when McBAR evolves initial populations to determine solutions to sub-problems set at or after the snapshot of the MOE taken immediately after the first batch of new tasks appeared. In addition, the formation of the initial population uses the genotype populations that correspond to sets of non-dominated solutions.

### 5.6.5 Centroid Repair

As briefly considered in Section 5.6.2, the centroid repairer $\mathcal{R}$ of GIBAR (inherited from CBAR) is revised so as to enhance the performance of McBAR. Let us now explore the revised repairer labelled $\mathcal{N}$ which perturbs any given centroid/genotype as little as possible such that, intuitively, the essence of this genotype as a centroid is diminished the least.

First, let us recall the action of $\mathcal{R}$ explained in Section 5.1.1. An ID $\rho$ of a centroid $C$ is allowed to be appended to a genotype $R$ (which starts from empty) if it satisfies the appending rule. Otherwise, the function $\mathcal{R}$ of CBAR randomly picks a different ID $\rho'$, which satisfies the appending rule, from $R^c = C - R$ and then appends this to $R$, where $R^c$ is the complementary genotype of $R$. Now, the function $\mathcal{N}$ of McBAR differs from $\mathcal{R}$ as follows: instead of randomly picking an element of $R^c$, the element of $R^c$ that is nearest in value to $\rho$, and satisfies the appending rule, is the one appended to $R$. If two IDs from $R^c$ are equidistant to $\rho$, and satisfy the appending rule, one of these is randomly picked for appending to $R$. The appending rule uses the node-inserted ID-mapped PNT in the $(t-1)^{th}$ snapshot of a MOE, if $t$ is greater than or equal to $\tau$ which is the SOSA of the MOE at which new tasks appear for the first time. If $t < \tau$ the appending rule uses the

PNT in the original state of the MOE.

As an example, consider the centroid $C = \langle 14, 1, 2, 7, 9, 3, 11, \ldots \rangle$; its first gene/ID, 14; the current $R = \emptyset$; and $R^c = C$. As demonstrated in Section 5.1.1, ID 14 does not satisfies the appending rule. Using the PNT illustrated in Figure 2.9, among those from $R^c$ that satisfies the appending rule; ID 3 is nearest to 14. Consequently, $R = \langle 3 \rangle$ and $R^c$ becomes $\langle 14, 1, 2, 7, 9, 11, \ldots \rangle$. ID 1 of $C$ is considered next for appending to $R$. It satisfies the appending rule such that $R = \langle 3, 1 \rangle$. ID 2 of $C$ is considered next and also satisfies the appending rule, such that $R = \langle 3, 1, 2 \rangle$.

Let the first three genes of $C$ form the vector $V = \langle 14, 1, 2 \rangle$; and the last $R$ be treated as vector. The last $R$, produced through the $\mathcal{N}$ function, has a Euclidean distance of 11 to $V$; while $R = \langle 1, 14, 2 \rangle$, produced (as explained in Section 5.1.1) through the $\mathcal{R}$ function, has a distance of 18.38 to $V$. This result supports the lesser perturbation effected by $\mathcal{N}$ than by $\mathcal{R}$ on repairing centroids.

### 5.6.6 Initial Population for a Current Static Sub-Problem

As explained in Section 5.6.2, the sub-problem $p_s^2$ of problem $p^2$ from $\mathcal{P}^2$ set in the $s^{th}$ snapshot of a MOE taken prior to the first appearance of new tasks is solved by McBAR by applying GIBAR (inherited by McBAR) with the centroid repairer $\mathcal{R}$ replaced by $\mathcal{N}$. Now, suppose that new tasks first appear at the $\tau^{th}$ SOSA of the MOE. And, $B\,[t]$ (defined through Equation 5.9) is a collection of genotype populations $B(k)$, where $t_o(t) \leq k < t$; $t \geq \tau$; and $t_o(t)$ is defined in Equation 5.6. Considering the explanation in Section 5.6.4, $B(k)$ is a population of gene-inserted ID-mapped genotypes (e.g. $B(k)$s in Equation 5.9 and $F(k)$s in Equation 5.11) or a population of un-inserted ID-mapped genotypes (e.g. $B(k)$s in Equation 5.10). Further, genotypes in $B(k)$ correspond to the non-dominated solutions to the sub-problem $p_k^2$. The collection $B\,[t]$ is used to form an initial population,

$$M_0(t) = C(t) \ \cup \ Rnd(t) \ \cup \ Mchs(t-1), \tag{5.12}$$

where,

1. $C(t)$ is a set of centroids,
$$C(t) = \bigcup_{k=t_o(t)}^{t-1} R(k); \tag{5.13}$$

2. $R(k)$ is the $\mathcal{N}$-repaired centroid of $B(k) \subseteq B[t]$;

3. $Rnd(t)$ is the set of genotypes generated through SSGS which utilises the ID-mapped and node-inserted (nodes correspond to new tasks) PNT in the $t^{th}$ snapshot of the MOE. $|Rnd(t)| = P - |C(t)| - 1$. Being produced through SSGS (which endows stochasticity to its produced genotypes as explained in Section 5.1.2), the elements of $Rnd(t)$ are referred to as random components. As noted in Section 2.4.2, memory-based EA approaches need to diversify the genotypes at stages of their evolutionary cycles. In McBAR, this diversification is implemented through the random components, thereby justifying the inclusion of $Rnd(t)$ in Equation 5.12.

4. $P$ is a fixed size of the initial population;

5. $Mchs(t-1)$ is the ID-mapped gene-inserted genotype derived from $B(t-1)$ that corresponds to the chosen schedule from the population $P(t-1)$ of non-dominated solutions to the sub-problem $p^2_{t-1}$.

Based on the enumerated definitions above, all genotypes in the initial population $M_0(t)$ have mapped IDs. This initial population is then evolved (as explained in Section 5.1.5) to determine an ID-mapped evolved population. When necessary, this evolved population is fed to the Solution Production method to obtain a set $P(t)$ of schedules as solutions to the sub-problem $p^2_t$.

### 5.6.7 Summary of Algorithm

The summary of the algorithm of McBAR is depicted in Figure 5.13.. The sub-problem $p^2_t$ of problem $p^2$ from $\mathcal{P}^2$ set in the $t^{th}$ snapshot of a MOE taken prior to the first

appearance of new tasks is solved by McBAR by applying GIBAR with the centroid repairer $\mathcal{R}$ replaced by $\mathcal{N}$. At the first appearance of new tasks, at $t = \tau$, new genes are inserted then IDs are mapped to set the system $\mathcal{S}$ to mapped mode. Next, centroids are computed and then repaired; initial population is formed using these centroids; and NSGA-II evolves this population under system $\mathcal{S}$ in mapped mode. At SOSAs where there is no new tasks and after the first increase, GIBAR is applied under system $\mathcal{S}$ in mapped mode. Lastly, if it is desired to present schedules, say to a commanding officer, current genotypes and precedence network of tasks are copied; IDs in these genotypes and network are unmapped; Schedule Formation method is applied to form schedules for the presentation.

The sub-algorithms of McBAR applied at or after the first appearance of new tasks are the following:

1. Gene insertion
2. ID Mapping operation $\mathcal{F}$
3. Compute unrepaired centroid using mean in Equation 5.1
4. Minimal repairer $\mathcal{N}$ of centroids
5. Initial population as defined in Equation 5.12
6. NSGA-II (explained in Section 5.1.5) used to evolve the initial population
7. Maintenance of system $\mathcal{S}$ at mapped mode
8. Preservation of ongoing and finished tasks in a chosen schedule (described in Section 5.1.6)
9. SSGS to form $Rnd(t)$ in Equation 5.12

The algorithm of McBAR is similar to that of GIBAR derived from CBAR (described in Section 5.1.5), except that Items 1 to 5 are applied to form an initial population; and that system $\mathcal{S}$ is maintained at mapped mode, i.e., Item 7.

```
Procedure McBAR
Begin
    For t=0 to τ − 1
        Apply GIBAR with the centroid repairer ℛ replaced by 𝒩
    End
    For t=τ to number of SOSAs
        If no increase in total number of tasks
            Apply GIBAR with the centroid repairer ℛ replaced by 𝒩
                under system 𝒮 in mapped mode
        Else
            If first increase in total task number
                Insert new genes
                Map IDs to put system 𝒮 to mapped mode
            Else
                Unmap IDs to put system 𝒮 to unmapped mode
                Insert new genes
                Restore system 𝒮 to mapped mode
            End
            Compute and then repair centroids
            Form initial population
            Evolve using NSGA-II under system 𝒮 in mapped mode
        End
        If presenting actual schedule
            Copyevolved genotypes and current Precedence
                Network of Tasks
            Unmap IDs in copied evolved genotypes and
                current Precedence Network of Tasks
            Apply Schedule Formation method to form schedules
            Output schedule
        End
    End
End
```

Figure 5.13: Algorithm of McBAR

The *core algorithm* of McBAR is composed of the above algorithms except for the use of the statistical mean to determine the unrepaired centroid (Item 3); the minimal repairer $\mathcal{N}$ (Item 4); and the $Rnd(t)$ in the initial population (Item 5) in Equation 5.12. The exempted sub-algorithms in the algorithm of McBAR will be replaced with other sub-algorithms to form McBA, MedianBAR and McBAS, which are elements from $\mathfrak{T}$ noted in Section 1.2.6.

## 5.7 MedianBAR

The centroid expressed in Equation 5.1 is a statistical mean of IDs. Note that mean is one type of tendency in statistics. It is of interest to determine the performance of McBAR when the centroid is a median of IDs, in which case the centroid is renamed as medoid and McBAR as *MedianBAR*. The algorithm of MedianBAR is the core algorithm of McBAR plus its utilised centroid as a median; centroid repairer as $\mathcal{N}$; and initial population in Equation 5.12.

The $i^{th}$ gene of a medoid $M(t)$ is:

$$M_i(t) = \begin{cases} s_i^{K+1}(t) & P \text{ is odd} \\ \left\lfloor \frac{s_i^K(t)+s_i^{K+1}(t)}{2} \right\rfloor & P \text{ is even,} \end{cases} \tag{5.14}$$

where $s_i^k$ is an element of the ordered set $S_i(t) = \left\langle s_i^1(t), s_i^2(t), \ldots, s_i^P(t) \right\rangle$; $S_i(t) = sort(X_i(t))$; $X_i(t) = \left\{ x_i^1(t), x_i^2(t), \ldots, x_i^P(t) \right\}$; $sort$ is a sorting operation; $x_i^j(t)$ is the task ID in the $i^{th}$ gene of the $j^{th}$ genotype in a population of size $P$; $K = \lfloor P/2 \rfloor$; and $t$ is the order of taking the snapshot where a sub-problem is set whose solutions/phenotypes correspond to genotypes in $P$.

## 5.8 EDA on $\mathcal{P}^2$ Problem

The EDA algorithm described in Section 2.6 is innovated in order to solve problems from $\mathcal{M}$ and consequently labelled as EDA/$\mathcal{P}^2$. EDA/$\mathcal{P}^2$ determines a set of solutions to sub-problem $p_0^2$ of problem $p^2$ from $\mathcal{P}^2$ by undertaking the following steps, starting with EDA cycle $i = 0$ and SOSA $t = 0$:

1. The probability matrix in Equation 2.16 is relabelled as $P_i(t)$ to indicate the $t^{th}$ SOSA of a MOE where the problem is set. It is initialised to have equal entries of $1/Nt$, i.e. $P_i(t) = [1/Nt]$, where $Nt$ is the total number of tasks at the $t^{th}$ snapshot of the MOE. This step is the equivalent of EDA/$\mathcal{P}^2$ to step 1 in Figure 2.18.

2. The probability matrix $P_i(t)$ is then sampled, following the sampling scheme in Section 2.6, to form a population $G_i(t)$ of $P$ genotypes. This step is related to step 2 in Figure 2.18.

3. From $G_i(t)$, a set $S_i(t)$ of schedules is formed through the Schedule Formation scheme described in Section 5.1.4.

4. The cost and makespan of each schedule in $S_i(t)$ are utilised in NSGA-II's fast non-dominated sorting to obtain each schedule's Pareto rank and crowding distance (elaborated in Sections 2.5.3.2 and 2.5.3.3 respectively).

5. $Psel = \lceil\ \rho P\ \rceil$ genotypes are selected from $G_i(t)$, where $\lceil \bullet \rceil$ is a round-up operator; $0 < \rho \leq 1$; and $\rho$ is a predefined constant referred to as *percent of population*. This selection is performed following the scheme described in Section 2.5.3.4 and based on the Pareto rank and crowding distance of schedules in $S_i(t)$ that correspond to the genotypes. This step is related to step 3 in Figure 2.18.

6. These $Psel$ selected genotypes are included in $P - Psel$ SSGS-generated genotypes to form a new population $G_{i+1}(t)$ of size $P$. Note that, as explained in Section 5.1.2, SSGS-generated genotypes add diversity to the population to which they are

included. This inclusion is intended to remedy the loss of diversity of the population produced by EDA as the evolutionary cycle $i$ of EDA progresses [130].

7. A probability matrix $Q_{i+1}(t)$ is estimated from the new population $G_{i+1}(t)$. Following [178], the probability matrix to be used in the next EDA cycle is,

$$P_{i+1}(t) = \lambda Q_{i+1}(t) + (1 - \lambda)P_i(t), \tag{5.15}$$

where $0 < \lambda \leq 1$ is a chosen value and referred to as a *learning rate*. This equation implies that the information, embedded in $P_i(t)$, from the current EDA cycle $i$ is carried over (learned) to the next EDA cycle $i + 1$. This step is related to step 5 in Figure 2.18.

8. Steps 2 to 7 are repeated for a fixed number *Ncyc* of EDA cycles, except at the last EDA cycle where only steps 2 and 3 are executed successively. The cycle index $i$ is incremented at every end of the cycle.

   At the last generation, where $i = Ncyc$, the set $S_{Ncyc}(t)$ of schedules produced in step 3 is the set of solutions to the sub-problem $p_t^2$, the set relabelled as $P(t)$.

Steps 6 and 1 are replaced to determine solutions to sub-problem $p_t^2$ of problem $p^2$ from $\mathcal{P}^2$ when $t > 0$. Step 6 is replaced as follows: The *Psel* selected genotypes are included to $P - Psel - 1$ SSGS-generated genotypes and a chosen genotype to complete a population $G_{i+1}(t)$ of size $P$. The chosen genotype corresponds to the chosen schedule randomly picked from the set $Pnds(t - 1)$ of non-dominated solutions in $P(t - 1)$, i.e. the non-dominated solutions to sub-problem $p_{t-1}^2$. The finished and on-going tasks in the chosen schedule are preserved (as explained in Section 5.1.6) in the evolved schedules being solutions to the sub-problem $p_t^2$.

When there is no increase in the total number of tasks at the $t^{th}$ SOSA of the MOE, $t > 0$, step 1 is replaced as follows: Let,

$$G\,[t] = \bigcup_{k=t_o(t)}^{t-1} Gnds(k), \tag{5.16}$$

where $t_o(t)$ is defined in Equation 5.6; and $Gnds(k)$ is a set of genotypes that correspond to all non-dominated solutions in the set $P(k)$ of solutions to sub-problem $p_k^2$, $t_o(t) \leq k < t$. The probability matrix $P_i(t)$ is estimated from $G\,[t]$. In this way, solutions (e.g. those that correspond to $Gnds(k)$) to sub-problems set in past snapshots are utilised to search for solutions to the sub-problem set in the current ($t^{th}$) snapshot of the MOE.

If there is an increase in the total number of tasks, say at the $\tau^{th}$ SOSA of the MOE, $\tau > 0$, step 1 is replaced as follows: New genes that correspond to new tasks which appear at the $\tau^{th}$ SOSA are inserted into all genotypes in $G\,[\tau]$ of Equation 5.16 following the scheme in Section 5.3. From the gene-inserted $G\,[\tau]$, the $Nt \times Nt$ probability matrix $P_i(\tau)$ is estimated, where $Nt$ is the total number of tasks at the $\tau^{th}$ SOSA.

The itemised algorithm of EDA/$\mathcal{P}^2$ above implies that any genotype it processed has a fixed length during its EDA process, just like the evolutionary process of EA applied by CBAR and GIBAR. Thus, following the explanation in Section 5.2, the gene-insertion is a legitimate step.

## 5.9 Techniques

Let us now describe techniques in $\mathcal{T}$ other than GIBAR, McBAR, MedianBAR and EDA/$\mathcal{P}^2$:

1. Centroid-Based Adaptation with Minimal Repair (*CBAM*) differs from GIBAR only in using $\mathcal{N}$ instead of $\mathcal{R}$ to repair centroids. Based on the descriptions in Sections 5.1.5, 5.3 and 5.6 of the sub-algorithms of CBAR, GIBAR and McBAR, respectively, GIBAR differs from McBAR in using $\mathcal{R}$, in the mapping $\mathcal{F}$ of task IDs in genotypes,

and in the maintenance of system $\mathcal{S}$ at mapped mode. Considering that CBAM differs from GIBAR by using $\mathcal{N}$ which is also used by McBAR, then it differs from McBAR in the mapping $\mathcal{F}$, and in the maintenance of system $\mathcal{S}$ at mapped mode. Based on the implication drawn in the discussion in Section 5.6, the application of the mapping $\mathcal{F}$ is the cause of the maintenance of the system at mapped mode. Considering this catalysis, the fundamental difference between CBAM and McBAR is only in regard to $\mathcal{F}$.

2. Mapping of Task IDs for Centroid-Based Adaptation with Stochastic Repair (*McBAS*) differs from McBAR only in using the random centroid repair $\mathcal{R}$ instead of the minimal centroid repair $\mathcal{N}$.

3. Mapping of Task IDs for Centroid-Based Adaptation (*McBA*) differs from McBAR in randomly selecting genotypes from the set $Gnds(t-1)$ of genotypes that correspond to the non-dominated solutions to the last sub-problem $p^2_{t-1}$ of a problem $p^2$ from $\mathcal{P}^2$. The set of randomly selected genotypes replaces the set $Rnd(t)$ of SSGS-generated genotypes in the initial population Equation 5.12, where $t$ is the order of the snapshot where the current sub-problem $p^2_t$ is set.

4. NDS of Last Population (*NDLPOP*) differs from GIBAR in randomly selecting genotypes, from the set of genotypes $Gnds(t-1)$, to form the $C(t)$ component of the initial population in Equation 5.4. Note that the resulting $C(t)$ is no longer a set of centroids but rather of genotypes of $Gnds(t-1)$. Thus, NDLPOP differs from McBAR in being an explicit memory-based approach.

5. Random Immigrants (*RI*) creates an initial population composed of totally SSGS-generated genotypes for NSGA-II to evolve. The length of each genotype in this initial population is equal to the total number of tasks in the MOE that sets the sub-problem being solved by RI. This technique differs from other techniques in $\mathcal{T}$ in applying no rule in creating its initial population, except the rules followed by SSGS.

All techniques in $\mathcal{T}$ are in the same row as the label $\mathcal{T}$ in Table 5.1. In Chapters 8 and 9, several subsets of $\mathcal{T}$ will be analysed. Each symbol ✓ in Table 5.1 denotes that the technique whose label is located above it belongs to the subset whose label is located to its left. For example, the subset $\mathcal{U}$ is comprised of CBAM, McBA, McBAR, McBAS and MedianBAR. Techniques that apply the mapping function $\mathcal{F}$, such as McBAS, McBA, McBAR, and MedianBAR, are classified as variants (of McBAR) and the rest of the techniques in $\mathcal{T}$ as non-variants.

Table 5.1: Sets of techniques

| | Non-variants | | | | | Variants | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\mathcal{T}$ | RI | NDLPOP | EDA/$\mathcal{P}^2$ | GIBAR | CBAM | McBA | McBAR | McBAS | MedianBAR |
| $\mathcal{E}$ | ✓ | ✓ | × | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $\mathcal{K}$ | ✓ | ✓ | ✓ | × | ✓ | ✓ | × | ✓ | ✓ |
| $\mathcal{W}$ | ✓ | × | × | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| $\mathcal{U}$ | × | × | × | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| $\mathcal{D}$ | ✓ | ✓ | × | × | ✓ | ✓ | ✓ | ✓ | × |

In our preliminary investigations, the techniques from $\mathcal{T}$ other than EDA/$\mathcal{P}^2$ have high performance with an evolutionary process which has a selection rate of 0.5. Further, the performance of all techniques in $\mathcal{T}$ is found to stabilise before the $300^{th}$ generation of the evolutionary processes in these techniques. Thus, the evolutionary processes are terminated at this value. The population size, crossover rate and mutation rate of each technique in $\mathcal{T}$ other than EDA/$\mathcal{P}^2$ are determined in Section 8.2. Also determined in the same section are the values of the learning rate $\lambda$ and the percentage $\rho$ (respectively defined in Items 7 and 5 of the last enumeration in Section 5.8) at which the performance of EDA/$\mathcal{P}^2$ is high.

# Effectiveness of McBAR

## Chapter 6

This chapter presents the fulfilment of the first and second thesis' goals through the limited method briefly described in Section 1.4.1. It begins in Section 6.1 by defining averages useful for investigating the relative performance of one technique over another from $\mathcal{T}$. These averages are taken over various environmental conditions but at fixed given $\delta$ in Equation A.9 which influences the task duration changes in the conditions; with fixed given type of environmental changes; and with fixed given sub-problem of a given problem from Table 4.5. Section 6.2 provides information on the process of the limited method aimed at determining the relative performance of the techniques with respect to a parameter of the stochastic model for the change in duration of tasks in a MOE. Based on this relative performance, the sub-algorithms of McBAR are legitimised by the limited method in Section 6.3. In Section 6.4, the limited method reveals the relative performance of the techniques with respect to the type of changes that occur in the MOE. It manifests in Section 6.5 the dynamics of the relative performance with respect to the SOSA of the MOE. The conclusions and suggested future work are discussed in Section 6.6.

## 6.1 Averages

Let us discuss the averages utilised to analyse the performance of techniques from $\mathcal{T}$. The average $\mathbf{E}_d^{\delta}[dSC(A,B)]$ differential set coverage of technique $A$ over technique $B$ is used to compare the performance of the techniques for solving sub-problems of problems from $\mathcal{L}^2$, a comparison with respect to the value $d$ of $\delta$. The parameter $\delta$ is found in Equation

A.9 which models the change in duration of tasks in MOEs where the problems are set. The average $\mathbf{E}_d^{\delta}\left[dSC(A,B)\right]$ is defined as,

$$\mathbf{E}_d^{\delta}\left[dSC(A,B)\right] = \sum_{i=1}^{Ns}\frac{1}{Ns}\sum_{j=1}^{Nc}\frac{1}{Nc}\sum_{k\ \in In(t)}\frac{1}{|In(t)|}dSC(A,B,i,j,k). \tag{6.1}$$

where $dSC(A,B,i,j,k)$ is the differential set coverage $dSC(A,B)$ (defined in Equation 2.8) of technique $A$ over technique $B$ determined for sub-problem $k_j^2$ (of the problem $k^2$ from $\mathcal{L}^2$, i.e. the problem labelled $k$ in Table 4.5); and at the $i^{th}$ simulation (explored in Section 4.4) of this problem. Further, $Ns$ is the number of simulations; and $Nc$ is the number of sub-problems in problem $k^2$, including $k_0^2$. Furthermore, $In(t)$ is a set of labels of problems in Table 4.5 that utilise a given value $d$ of $\delta$. For example, based on Table 4.5, $In(3.0) = \{1,2,3,7,8,9,13,14,15,19,20,21,25,26,27\}$.

The average $\mathbf{E}_t^{\tau}\left[dSC(A,B)\right]$ differential set coverage of technique $A$ over technique $B$ is used to compare the performance of the techniques from $\mathfrak{T}$ at different types $t$ of changes listed in Table 4.2. It is defined as,

$$\mathbf{E}_t^{\tau}\left[dSC(A,B)\right] = \sum_{i=1}^{Ns}\frac{1}{Ns}\sum_{k=1}^{Nn}\frac{1}{Nn}\sum_{j\in Soc(k,\tau)}\frac{dSC(A,B,i,j,k)}{|Soc(j,\tau)|}, \tag{6.2}$$

where $Nn = 30$ is the number of all problems with labels found in Table 4.5; and $Soc(k,t)$ is a set of SOSAs of the MOE where the problem labelled $k$ is set, the SOSAs at which type $t$ of changes occurs. To exemplify, consider problem labelled 1 which, based on Table 4.5, has a TSC $S_1$. In this TSC, SOSAs with type $t = 0$ of changes (task duration change only, based on Table 4.2) are in the set $Soc(1,0) = \{1,2,3,5,9,12\}$, based on Table 4.3. Note that zeroth SOSA is excluded in the definition of the averages.

The average $\mathbf{E}_{j,k}^{\sigma}\left[dSC(A,B)\right]$ differential set coverage of technique $A$ over technique $B$ is used to compare the performance of the techniques from $\mathfrak{T}$ at different simulations of sub-problem $k_j^2$. It is defined as,

$$\mathbf{E}_{j,k}^{\sigma}\left[dSC(A,B)\right] = \frac{\sum_{i=1}^{Ns} dSC(A,B,i,j,k)}{Ns}, \tag{6.3}$$

where all indices are already defined in this section.

Note that all of the averages are derived from differential set coverage which, as defined in Section 2.5.2, is a measure of the performance of one technique over another. The following definition is then taken: if any of the above-defined averages is greater than zero then technique $A$ performs better than technique $B$, in determining solutions to problems from $\mathcal{L}^2$ (problems whose labels are listed in Table 4.5), over the domain (e.g. type $t$ of environmental change) at which this average is taken.

## 6.2 Relative Performance Under Task Duration Change

The limited method applies each technique from $\mathcal{T}$ to solve each sub-problem of each problem from $\mathcal{L}^2$. The obtained solutions are utilised by this method to determine the average defined in Equation 6.1. And from this average, the limited method compares the performance of the techniques with respect to the parameter $\delta$ in Equation A.9 which models task duration changes in the MOE where the problems are set. The remaining discussions in this section explain the results of the actions of the limited method. For brevity, let the average $\mathbf{E}_t^{\delta}[\bullet]$ refer simply to performance in this and in the next section.

Table 6.1 (a) presents the performance $\mathbf{E}_t^{\delta}[dSC(A, B)]$ of technique $A$ over technique $B$ whose labels are found in the first column and first row of this table respectively. These performances are derived from the solutions determined by the techniques for solving each sub-problem of each problem from $\mathcal{L}^2$, problems that have task duration changes modelled by Equation A.9 with $\delta = 3.0$. Referring to the second row of this table, $\mathbf{E}_{3.0}^{\delta}[dSC(\text{GIBAR}, T)] > 0$ only when technique $T$ is either NDLPOP, RI or EDA/$\mathcal{P}^2$. Thus, the performance of GIBAR is better than those of NDLPOP, RI and EDA/$\mathcal{P}^2$. Note that the degree of performance is based on the definition in Section 6.1. Succeeding rows in this table demonstrate that: the performance of CBAM is inferior only to those

of the variants (defined in Section 5.9); the performance of NDLPOP is superior only to those of EDA/$\mathcal{P}^2$ and RI; the performance of RI is superior only to that of EDA/$\mathcal{P}^2$; the performance of EDA/$\mathcal{P}^2$ is inferior to all other techniques; the performance of McBA is inferior to those of other variants and superior to those of non-variants; the performance of McBAR is inferior to that of MedianBAR only; the performance of McBAS is inferior to those of McBAR and MedianBAR only; and the performance of MedianBAR is superior to those of all other techniques. Note however that the degree of MedianBAR's superiority to McBAR is small, $\mathbf{E}^{\delta}_{3.0}\left[dSC(\text{McBAR}, \text{MedianBAR})\right] = .01$, i.e. near zero. Data in this table showed the performance of the variants to be superior to those of the non-variants.

Table 6.1: Average differential set coverage $\mathbf{E}\left[dSC\delta\right]$ with (a) $\delta = 3.0$ (b) $\delta = 6.0$

| Techniques | GIBAR | CBAM | NDLPOP | RI | EDA/$\mathcal{P}^2$ | McBA | McBAR | McBAS | MedianBAR |
|---|---|---|---|---|---|---|---|---|---|
| GIBAR | N/A | -0.54 | 0.56 | 0.37 | 0.99 | -0.64 | -0.63 | -0.61 | -0.65 |
| CBAM | 0.54 | N/A | 0.01 | 0.23 | 0.99 | -0.35 | -0.36 | -0.33 | -0.38 |
| NDLPOP | -0.56 | -0.01 | N/A | 0.24 | 1.00 | -0.34 | -0.36 | -0.33 | -0.37 |
| RI | -0.37 | -0.23 | -0.24 | N/A | 1.00 | -0.47 | -0.49 | -0.45 | -0.50 |
| EDA/$\mathcal{P}^2$ | -0.99 | -0.99 | -1.00 | -1.00 | N/A | -1.00 | -1.00 | -1.00 | -1.00 |
| McBA | 0.64 | 0.35 | 0.34 | 0.47 | 1.00 | N/A | -0.07 | -0.01 | -0.09 |
| McBAR | 0.63 | 0.36 | 0.36 | 0.49 | 1.00 | 0.07 | N/A | 0.07 | -0.01 |
| McBAS | 0.61 | 0.33 | 0.33 | 0.45 | 1.00 | 0.01 | -0.07 | N/A | -0.08 |
| MedianBAR | 0.65 | 0.38 | 0.37 | 0.50 | 1.00 | 0.09 | 0.01 | 0.08 | N/A |

(a)

| Technique | GIBAR | CBAM | NDLPOP | RI | EDA/$\mathcal{P}^2$ | McBA | McBAR | McBAS | MedianBAR |
|---|---|---|---|---|---|---|---|---|---|
| GIBAR | N/A | -0.58 | 0.56 | 0.37 | 0.99 | -0.65 | -0.66 | -0.62 | -0.67 |
| CBAM | 0.58 | N/A | 0.05 | 0.28 | 1.00 | -0.34 | -0.36 | -0.31 | -0.37 |
| NDLPOP | -0.56 | -0.05 | N/A | 0.25 | 1.00 | -0.37 | -0.39 | -0.35 | -0.38 |
| RI | -0.37 | -0.28 | -0.25 | N/A | 1.00 | -0.50 | -0.51 | -0.46 | -0.51 |
| EDA/$\mathcal{P}^2$ | -0.99 | -1.00 | -1.00 | -1.00 | N/A | -1.00 | -1.00 | -1.00 | -1.00 |
| McBA | 0.65 | 0.34 | 0.37 | 0.50 | 1.00 | N/A | -0.09 | 0.00 | -0.08 |
| McBAR | 0.66 | 0.36 | 0.39 | 0.51 | 1.00 | 0.09 | N/A | 0.11 | 0.01 |
| McBAS | 0.62 | 0.31 | 0.35 | 0.46 | 1.00 | 0.00 | -0.11 | N/A | -0.07 |
| MedianBAR | 0.67 | 0.37 | 0.38 | 0.51 | 1.00 | 0.08 | -0.01 | 0.07 | N/A |

(b)

Table 6.1 (b) presents the performances of technique $A$ over technique $B$ whose labels are found at the first column and first row of this table respectively. This average is determined by the techniques from $\mathcal{T}$ for solving each sub-problem of each problem from $\mathcal{L}^2$, the problems that have task duration changes modelled by Equation A.9 with $\delta = 6.0$. Table 6.1 (b) illustrates that the performance between the techniques from $\mathcal{T}$ are generally similar to those expressed in Table 6.1 (a), except that the performance of McBAR is superior to those of all other techniques, especially on MedianBAR. Further, the performance of MedianBAR is inferior by a small degree to that of McBAR,

$\mathbf{E}_{6.0}^{\delta}[dSC(\text{MedianBAR}, \text{McBAR})] = -.01$. In view of these findings and considering that Tables 6.1 (a) and (b) correspond to $\delta = 3.0$ and $\delta = 6.0$, respectively, the performance of techniques in both tables are generally not affected by the two values of $\delta$.

Let the rank of superiority in performance of a technique over other techniques in $\mathcal{T}$ be the number of other techniques on both tables to which it is superior. The techniques in $\mathcal{T}$ arranged in descending rank of superiority are: McBAR, MedianBAR, McBAS, McBA, CBAM, GIBAR, NDLPOP, RI and EDA/$\mathcal{P}^2$ whose performances are superior to 15, 15, 12, ten, eight, six, four, two and zero other techniques in the two tables respectively.

## 6.3 Legitimisation

The fourth thesis' goal is to legitimise some sub-algorithms of McBAR. The legitimisation undertaken in this section is by the limited method and is as follows. As discussed in Section 1.4.4, this legitimisation is based on the principle that if technique $A$ differs from technique $B$ in regard to some components, and if, in general, $A$ performs better than $B$ in solving a given set of problems then the components of $A$ as *distinct* from $B$ are legitimate for $A$ to solve this set of problems. Consider a sample case of the legitimisation; and note that McBAR uses EA operators while EDA/$\mathcal{P}^2$ uses distribution sampling to produce offspring. If in solving most sub-problems of problems from $\mathcal{L}^2$, the performance of McBAR is better than that of EDA/$\mathcal{P}^2$ then the EA operators utilised in McBAR are legitimate components of McBAR to solve these sub-problems.

As implied in Sections 5.8 and 5.9, the sub-algorithmic differences of McBAR from techniques in $\mathcal{K} = \{\text{RI}, \text{NDLPOP}, \text{EDA}/\mathcal{P}^2, \text{CBAM}, \text{McBA}, \text{McBAS}, \text{MedianBAR}\}$ (defined through Table 5.1) are as follows:

1. McBAR fundamentally differs from CBAM only in using the mapping function $\mathcal{F}$.
2. McBAR differs from McBAS only in using minimal centroid repair $\mathcal{N}$ elaborated in Section 5.6.5.

3. McBAR differs from McBA in generating genotypes through SSGS to form the $Rnd(t)$ component of its initial population (defined in Equation 5.12), where a current sub-problem set in the $t^{th}$ SOSA of a given MOE.

4. McBAR differs from MedianBAR in using mean (defined in Equation 5.1) to compute for the centroid $R(k)$ in Equation 5.12.

5. McBAR differs from EDA/$\mathcal{P}^2$ in using mutation and crossover operators – to form the next generation offspring in its evolutionary process – instead of sampling and estimation of a probability matrix (described in Section 2.6). Further, it differs in the use of centroid.

6. McBAR differs from NDLPOP in being an implicit memory-based approach.

7. McBAR differs from RI in using Equation 5.12 to generate an initial population for NSGA-II to evolve instead of not using any rule except SSGS.

Let us now legitimise sub-algorithms of McBAR using the results in the last section and the just enumerated differences of McBAR to other techniques in $\mathcal{K}$. Based on the legitimisation principle, the superiority in performance of McBAR to CBAM (Item 1) legitimises its use of mapping $\mathcal{F}$ since it fundamentally differs from CBAM on this sub-algorithm. Its superiority in performance to McBAS (Item 2) legitimises its use of the minimum centroid repair $\mathcal{N}$ since it differs from McBAS on this sub-algorithm only. Its superiority in performance to McBA (Item 3) legitimises its use of the random component $S(t)$ in Equation 5.12 since it differs from McBA by this component only. As noted in Section 5.1.2, randomly generated individuals can diversify an initial population that includes them. Thus, the superiority in performance of McBAR over McBA supports the finding in [36] on the relevance of diversification of population in some EA evolutionary processes. The approximately equal performance of McBAR to MedianBAR (Item 4) does not legitimise its use of the mean to compute the centroid through Equation 5.1. The superiority in performance of McBAR to NDLPOP (Item 6) legitimises its use of the implicit memory-based approach since it differs from NDLPOP on this type of approach.

Its superiority in performance to EDA/$\mathcal{P}^2$ (Item 5) legitimises its combined use of EA operators and centroids since it fundamentally differs from EDA/$\mathcal{P}^2$ on these components.

Table 6.2 lists the average number of CPU cycles, on identical computing machines, required by techniques in $\mathcal{T}$ to determine solutions to sub-problems of problems from $\mathcal{L}^2$ (whose labels are listed in Table 4.5). This average is taken over a similar domain as in the average $\mathbf{E}_d^{\delta}[\bullet]$ in Equation 6.1. Based on the column with the heading $\delta = 3.0$, non-variants require lesser CPU cycles to determine the solutions than variants. Among the variants, MedianBAR requires the most number of CPU cycles to determine the solutions. This last result is expected since the computation of the median in MedianBAR is more algorithmically complex (mainly due to the sorting) than the computation of the mean in McBAR based on Equations 5.14 and 5.1 respectively. The relationships of the numbers of CPU cycles for the column heading $\delta = 3.0$ are also true for the column heading $\delta = 6.0$. Thus, the relationships are not affected by the two values of $\delta$.

Table 6.2: Average execution time

| Technique | δ = 3.0 | δ = 6.0 |
|---|---|---|
| RI | $4.28 \times 10^{10}$ | $4.45 \times 10^{10}$ |
| NDLPOP | $4.64 \times 10^{10}$ | $4.71 \times 10^{10}$ |
| EDA/$\mathcal{P}^2$ | $4.82 \times 10^{10}$ | $4.92 \times 10^{10}$ |
| GIBAR | $5.01 \times 10^{10}$ | $5.08 \times 10^{10}$ |
| CBAM | $5.02 \times 10^{10}$ | $5.10 \times 10^{10}$ |
| McBAS | $5.05 \times 10^{10}$ | $5.12 \times 10^{10}$ |
| McBA | $5.28 \times 10^{10}$ | $5.32 \times 10^{10}$ |
| McBAR | $5.42 \times 10^{10}$ | $5.46 \times 10^{10}$ |
| MedianBAR | $6.60 \times 10^{10}$ | $6.72 \times 10^{10}$ |

McBAR requires less computational expense than MedianBAR to determine the solutions to all sub-problems of problems from $\mathcal{L}^2$; its performance is superior (as illustrated in Table 6.1(b)) to that of MedianBAR in solving problems from $\mathcal{L}^2$ whose task duration changes are modelled by Equation A.9 with $\delta = 6.0$; and its performance is superior (as illustrated in both Tables 6.1 (a) and (b)) to those of all techniques in $\mathcal{T}$, other than itself and MedianBAR. Despite the inferior (as illustrated in Table 6.1(a)) performance

of McBAR to MedianBAR in solving problems from $\mathcal{L}^2$ whose task duration changes are modelled by Equation A.9 with $\delta = 3.0$, the results on the average execution time and on the average $\mathbf{E}^{\delta}_{6.0}[\bullet]$ show McBAR to be the most versatile technique among all the techniques in $\mathcal{T}$.

## 6.4 Relative Performance Under Type of Changes

The limited method investigates the influence of types of changes in MOEs on the relative performance of techniques from $\mathcal{T}$, over each other, for solving problems from $\mathcal{L}^2$ set in the MOEs. This relative performance is measured in this section in terms of the average $\mathbf{E}^{\tau}_{t}[\bullet]$ defined in Equation 6.2. The following presents the results of the investigation.

Tables 6.3 to 6.5 present the relative performance of techniques from $\mathcal{T}$ measured through $\mathbf{E}^{\tau}_{t}[dSC(A, B)]$ where $A$ is the technique name under the column heading "Technique" and $B$ is the technique name in other table heading. Further, the integer under the column heading "Type" is the change type label $t$ enumerated in Table 4.2. From the first row of type 0 group of rows of Table 6.3, $\mathbf{E}^{\tau}_{0}[dSC(\text{GIBAR}, \text{McBA})]$ = -0.59, $\mathbf{E}^{\tau}_{0}[dSC(\text{GIBAR}, \text{McBAR})]$ = -0.58, $\mathbf{E}^{\tau}_{0}[dSC(\text{GIBAR}, \text{McBAS})]$ = -0.57, and $\mathbf{E}^{\tau}_{0}[dSC($ GIBAR, MedianBAR)] = -0.61. Thus, the performance of GIBAR is inferior to those of all the variants when the MOEs change in task duration only (type 0 in Table 4.2), the MOEs where problems from $\mathcal{P}^2$ are set. The performance of CBAM, NDLPOP, and RI is inferior to those of the variants at change type 0 based, respectively, on the second to the fourth rows of the type 0 group of rows.

The performance of the non-variants, in the type 5 group of rows in Table 6.4, is inferior to those of the variants when the change in the MOEs is of type 5 (simultaneous changes in task duration and number based on Table 4.2). The degree of inferiority is approximately of similar degree as to that found when the change in the MOEs is of Type 0. The performance of non-variants, in type 1 group of rows in Table 6.3, is inferior to that of variants when the change in the MOE is of type 1 (change in resource availability

Table 6.3: Average $\mathbf{E}_t^\tau[\bullet]$ performance under change types 0 to 2

| Type | Technique | GIBAR | CBAM | NDLPOP | RI | EDA/$\mathscr{P}^2$ | McBA | McBAR | McBAS | MedianBAR |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | GIBAR | N/A | -0.55 | 0.56 | 0.34 | 0.99 | -0.59 | -0.58 | -0.57 | -0.61 |
| | CBAM | 0.55 | N/A | 0.03 | 0.29 | 0.99 | -0.25 | -0.23 | -0.21 | -0.25 |
| | NDLPOP | -0.56 | -0.03 | N/A | 0.29 | 0.99 | -0.25 | -0.25 | -0.22 | -0.25 |
| | RI | -0.34 | -0.29 | -0.29 | N/A | 0.99 | -0.43 | -0.42 | -0.38 | -0.43 |
| | EDA/$\mathscr{P}^2$ | -0.99 | -0.99 | -0.99 | -0.99 | N/A | -0.99 | -1.00 | -1.00 | -1.00 |
| | McBA | 0.59 | 0.25 | 0.25 | 0.43 | 0.99 | N/A | -0.04 | 0.01 | -0.05 |
| | McBAR | 0.58 | 0.23 | 0.25 | 0.42 | 1.00 | 0.04 | N/A | 0.05 | -0.02 |
| | McBAS | 0.57 | 0.21 | 0.22 | 0.38 | 1.00 | -0.01 | -0.05 | N/A | -0.07 |
| | MedianBAR | 0.61 | 0.25 | 0.25 | 0.43 | 1.00 | 0.05 | 0.02 | 0.07 | N/A |
| 1 | GIBAR | N/A | -0.59 | 0.57 | 0.40 | 1.00 | -0.73 | -0.72 | -0.66 | -0.74 |
| | CBAM | 0.59 | N/A | 0.05 | 0.26 | 1.00 | -0.43 | -0.48 | -0.43 | -0.49 |
| | NDLPOP | -0.57 | -0.05 | N/A | 0.23 | 1.00 | -0.44 | -0.48 | -0.44 | -0.50 |
| | RI | -0.40 | -0.26 | -0.23 | N/A | 1.00 | -0.55 | -0.58 | -0.53 | -0.60 |
| | EDA/$\mathscr{P}^2$ | -1.00 | -1.00 | -1.00 | -1.00 | N/A | -1.00 | -1.00 | -1.00 | -1.00 |
| | McBA | 0.73 | 0.43 | 0.44 | 0.55 | 1.00 | N/A | -0.16 | -0.06 | -0.15 |
| | McBAR | 0.72 | 0.48 | 0.48 | 0.58 | 1.00 | 0.16 | N/A | 0.11 | 0.04 |
| | McBAS | 0.66 | 0.43 | 0.44 | 0.53 | 1.00 | 0.06 | -0.11 | N/A | -0.08 |
| | MedianBAR | 0.74 | 0.49 | 0.50 | 0.60 | 1.00 | 0.15 | -0.04 | 0.08 | N/A |
| 2 | GIBAR | N/A | -0.53 | 0.51 | 0.19 | 0.99 | -0.82 | -0.84 | -0.82 | -0.84 |
| | CBAM | 0.53 | N/A | 0.03 | 0.28 | 0.99 | -0.71 | -0.73 | -0.69 | -0.74 |
| | NDLPOP | -0.51 | -0.03 | N/A | 0.24 | 0.99 | -0.74 | -0.75 | -0.72 | -0.75 |
| | RI | -0.19 | -0.28 | -0.24 | N/A | 0.99 | -0.78 | -0.79 | -0.77 | -0.79 |
| | EDA/$\mathscr{P}^2$ | -0.99 | -0.99 | -0.99 | -0.99 | N/A | -1.00 | -1.00 | -1.00 | -1.00 |
| | McBA | 0.82 | 0.71 | 0.74 | 0.78 | 1.00 | N/A | -0.01 | 0.02 | -0.04 |
| | McBAR | 0.84 | 0.73 | 0.75 | 0.79 | 1.00 | 0.01 | N/A | 0.09 | -0.07 |
| | McBAS | 0.82 | 0.69 | 0.72 | 0.77 | 1.00 | -0.02 | -0.09 | N/A | -0.08 |
| | MedianBAR | 0.84 | 0.74 | 0.75 | 0.79 | 1.00 | 0.04 | 0.07 | 0.08 | N/A |

Table 6.4: Average $\mathbf{E}_t^\tau[\bullet]$ performance under Change Types 3 to 5

| Type | Technique | GIBAR | CBAM | NDLPOP | RI | EDA/$\mathscr{P}^2$ | McBA | McBAR | McBAS | MedianBAR |
|---|---|---|---|---|---|---|---|---|---|---|
| 3 | GIBAR | N/A | -0.62 | 0.58 | 0.14 | 1.00 | -0.90 | -0.92 | -0.89 | -0.92 |
| | CBAM | 0.62 | N/A | 0.07 | 0.24 | 1.00 | -0.68 | -0.76 | -0.70 | -0.75 |
| | NDLPOP | -0.58 | -0.07 | N/A | 0.16 | 1.00 | -0.71 | -0.78 | -0.73 | -0.76 |
| | RI | -0.14 | -0.24 | -0.16 | N/A | 1.00 | -0.78 | -0.82 | -0.78 | -0.82 |
| | EDA/$\mathscr{P}^2$ | -1.00 | -1.00 | -1.00 | -1.00 | N/A | -1.00 | -1.00 | -1.00 | -1.00 |
| | McBA | 0.90 | 0.68 | 0.71 | 0.78 | 1.00 | N/A | -0.24 | -0.10 | -0.20 |
| | McBAR | 0.92 | 0.76 | 0.78 | 0.82 | 1.00 | 0.24 | N/A | 0.14 | 0.05 |
| | McBAS | 0.89 | 0.70 | 0.73 | 0.78 | 1.00 | 0.10 | -0.14 | N/A | -0.06 |
| | MedianBAR | 0.92 | 0.75 | 0.76 | 0.82 | 1.00 | 0.20 | -0.05 | 0.06 | N/A |
| 4 | GIBAR | N/A | -0.57 | 0.58 | 0.40 | 1.00 | -0.21 | -0.27 | -0.21 | -0.26 |
| | CBAM | 0.57 | N/A | 0.02 | 0.24 | 1.00 | 0.16 | 0.09 | 0.15 | 0.09 |
| | NDLPOP | -0.58 | -0.02 | N/A | 0.24 | 1.00 | 0.16 | 0.07 | 0.11 | 0.08 |
| | RI | -0.40 | -0.24 | -0.24 | N/A | 1.00 | 0.03 | -0.06 | 0.01 | -0.03 |
| | EDA/$\mathscr{P}^2$ | -1.00 | -1.00 | -1.00 | -1.00 | N/A | -1.00 | -1.00 | -1.00 | -1.00 |
| | McBA | 0.21 | -0.16 | -0.16 | -0.03 | 1.00 | N/A | -0.09 | 0.01 | -0.10 |
| | McBAR | 0.27 | -0.09 | -0.07 | 0.06 | 1.00 | 0.09 | N/A | 0.10 | -0.01 |
| | McBAS | 0.21 | -0.15 | -0.11 | -0.01 | 1.00 | -0.01 | -0.10 | N/A | -0.07 |
| | MedianBAR | 0.26 | -0.09 | -0.08 | 0.03 | 1.00 | 0.10 | 0.01 | 0.07 | N/A |
| 5 | GIBAR | N/A | -0.60 | 0.59 | 0.09 | 1.00 | -0.68 | -0.68 | -0.64 | -0.68 |
| | CBAM | 0.60 | N/A | 0.03 | 0.18 | 1.00 | -0.36 | -0.42 | -0.35 | -0.40 |
| | NDLPOP | -0.59 | -0.03 | N/A | 0.15 | 1.00 | -0.39 | -0.43 | -0.38 | -0.42 |
| | RI | -0.09 | -0.18 | -0.15 | N/A | 1.00 | -0.46 | -0.50 | -0.45 | -0.49 |
| | EDA/$\mathscr{P}^2$ | -1.00 | -1.00 | -1.00 | -1.00 | N/A | -1.00 | -1.00 | -1.00 | -1.00 |
| | McBA | 0.68 | 0.36 | 0.39 | 0.46 | 1.00 | N/A | -0.11 | -0.01 | -0.10 |
| | McBAR | 0.68 | 0.42 | 0.43 | 0.50 | 1.00 | 0.11 | N/A | 0.14 | 0.02 |
| | McBAS | 0.64 | 0.35 | 0.38 | 0.45 | 1.00 | 0.01 | -0.14 | N/A | -0.11 |
| | MedianBAR | 0.68 | 0.40 | 0.42 | 0.49 | 1.00 | 0.10 | -0.02 | 0.11 | N/A |

number based on Table 4.2). The last inferiority is of higher degree when the change in the MOE is of type 0: for example, $\mathbf{E}_0^{\tau}\left[dSC(\text{CBAM}, \text{McBAS})\right] < \mathbf{E}_1^{\tau}\left[dSC(\text{CBAM}, \text{McBAS})\right]$.

The type 2 group of rows in Table 6.3, type 3 group of rows in Table 6.4, and type 6 group of rows in Table 6.5 demonstrate that the performance of the non-variants is inferior to that of the variants when the changes in the MOEs are of types 2, 3 and 6 respectively. This inferiority is greater when the change in the MOEs is of type 1. Note that change types 2, 3 and 6 involve change in the total number of tasks (based on Table 4.2). The above results manifest the superiority in performance of variants over non-variants when the changes in the MOEs involve change in total number of tasks, i.e. change type 2, 3, 5 and 6. However, the performance of the variants in the type 4 group of rows in Table 6.3 is less than CBAM, NDLPOP and RI when the change in the MOEs is of type 4 which, according to Table 4.2, denotes simultaneous changes in task duration and resource availability, i.e. a type of changes that does not involve change in the total number of tasks. Tables 6.3 to 6.5 illustrate the inferior performance of EDA/$\mathcal{P}^2$ to all other techniques in $\mathcal{T}$ with any type of changes in the MOEs. Further, they manifest the approximately similar relative performance of McBAR and MedianBAR over the other techniques in $\mathcal{T}$.

## 6.5 Dynamic Performance

The limited method also investigates the influence of the dynamics of MOEs on the average $\mathbf{E}_{j,k}^{\sigma}\left[\bullet\right]$ (defined in Equation 6.3) relative performance of techniques in $\mathcal{T}$ for solving problems from $\mathcal{L}^2$ set in these MOEs. Before pursuing this investigation, first, let us consider the following: (a) As demonstrated in Section 6.2, the average $\mathbf{E}_t^{\delta}\left[\bullet\right]$ relative performance of EDA/$\mathcal{P}^2$ for solving the problems is inferior to those of all other techniques in $\mathcal{T}$. (b) The average $\mathbf{E}_t^{\delta}\left[\bullet\right]$ performance of McBAR and MedianBAR is approximately similar. (c) Not shown, the dynamics of the average $\mathbf{E}_{j,k}^{\sigma}\left[\bullet\right]$ performance of GIBAR and CBAM relative to other techniques in $\mathcal{T}$ are not significantly different. Following

Table 6.5: Average $\mathbf{E}_t^{\tau}[\bullet]$ performance under Change Type 6

| Type | Technique | GIBAR | CBAM | NDLPOP | RI | EDA/$\mathcal{P}^2$ |
|---|---|---|---|---|---|---|
| 6 | GIBAR | N/A | -0.54 | 0.53 | 0.11 | 1.00 |
| | CBAM | 0.54 | N/A | 0.03 | 0.19 | 1.00 |
| | NDLPOP | -0.53 | -0.03 | N/A | 0.18 | 1.00 |
| | RI | -0.11 | -0.19 | -0.18 | N/A | 1.00 |
| | EDA/$\mathcal{P}^2$ | -1.00 | -1.00 | -1.00 | -1.00 | N/A |
| | McBA | 0.85 | 0.63 | 0.63 | 0.72 | 1.00 |
| | McBAR | 0.84 | 0.65 | 0.65 | 0.73 | 1.00 |
| | McBAS | 0.81 | 0.59 | 0.60 | 0.68 | 1.00 |
| | MedianBAR | 0.85 | 0.64 | 0.65 | 0.73 | 1.00 |

| Type | Technique | McBA | McBAR | McBAS | MedianBAR |
|---|---|---|---|---|---|
| 6 | GIBAR | -0.85 | -0.84 | -0.81 | -0.85 |
| | CBAM | -0.63 | -0.65 | -0.59 | -0.64 |
| | NDLPOP | -0.63 | -0.65 | -0.60 | -0.65 |
| | RI | -0.72 | -0.73 | -0.68 | -0.73 |
| | EDA/$\mathcal{P}^2$ | -1.00 | -1.00 | -1.00 | -1.00 |
| | McBA | N/A | -0.15 | -0.02 | -0.11 |
| | McBAR | 0.15 | N/A | 0.14 | 0.07 |
| | McBAS | 0.02 | -0.14 | N/A | -0.07 |
| | MedianBAR | 0.11 | -0.07 | 0.07 | N/A |

these observations EDA/$\mathcal{P}^2$, GIBAR and MedianBAR are excluded from the succeeding discussions. The techniques included in the succeeding discussions are members of the set $\mathcal{D} = \{\text{RI}, \text{NDLPOP}, \text{CBAM}, \text{McBA}, \text{McBAR}, \text{McBAS}\}$ (defined through Table 5.1).

For brevity, let the average $\mathbf{E}_{j,k}^{\sigma}[dSC(A, B)]$ be referred to simply as the relative performance of technique $A$ over $B$ in the remaining portion of this section. As the second preparatory step for the investigation of the limited method in this section, let us describe a certain figure format important in presenting the dynamics of the relative performance of the techniques from $\mathcal{D}$.

### 6.5.1 Figure Arrangement

The average $\mathbf{E}_{j,k}^{\sigma}[dSC(T, S)]$ relative performance of technique $T$ over $S$ is illustrated in Figure 6.1 for some sub-problems $k_j^2$ of problems $k^2$ ($k$s are listed in Table 4.5) from $\mathcal{L}^2$, where $0 \leq j \leq 12$. In Figure 6.1(a), the heading of the block of white small squares is the name of technique $T$, called the basis technique (e.g. McBAR). Further, each heading over coloured blocks is the name of the technique denoted as $S$. For example,

the blocks of squares in Figure 6.1(a) under the heading McBAS correspond to averages $\mathbf{E}^{\sigma}_{j,k}\left[dSC(\text{McBAR}, \text{McBAS})\right]$ for various combinations of indexes $j$ and $k$. This technique-heading correspondence applies to all other sub-figures of Figure 6.1.

In Figure 6.1(a), the row of blocks at a similar horizontal level as the label $S_i$ at the left of this figure corresponds to TSC $S_i$ listed in Table 4.5, where $1 \leq i \leq 3$. For example, the vertically middle blocks correspond to $S_2$. A coloured block in the figure is denoted by the $S_i$ label which is of similar level as this block, and also by the heading under which this block is found. For example, $S_2$-RI block is the vertically middle block under the RI column heading. Blocks at the same level as the $S_i$ label also correspond to an ordered set $N_i^{3.0}$ of problems from $\mathcal{L}^2$ where,

$$N_1^{3.0} = \langle 1, 7, 13, 19, 25 \rangle \,, \tag{6.4}$$

$$N_2^{3.0} = \langle 2, 8, 14, 20, 26 \rangle \tag{6.5}$$

and

$$N_3^{3.0} = \langle 3, 9, 15, 21, 27 \rangle \,. \tag{6.6}$$

For example, the middle blocks correspond to $N_2^{3.0}$. Inside a block at the same level as the $S_i$ label, the rows from bottom to top correspond to elements of $N_i^{3.0}$ respectively. For example, the bottom to top rows of $S_2$-RI block correspond to problems labelled 2, 8, 14, 20 and 26 respectively.

The horizontal coordinate of each small square in each block corresponds to the $j^{th}$ SOSA of the MOEs where the problems with labels from $N_i^{3.0}$ are set. Continuing the example above, the fourth small square from the left on the third row from the bottom of the $S_2$-RI block corresponds to the fourth SOSA of the MOE where problem 14 (the third element of $N_2^{3.0}$) is set, i.e. it corresponds to sub-problem $14_4^2$ (refer to Section 4.2.1 for notation). The colorbar at the rightmost of the figure maps colours in the squares to values. In the example, the colour of the small square that corresponds to

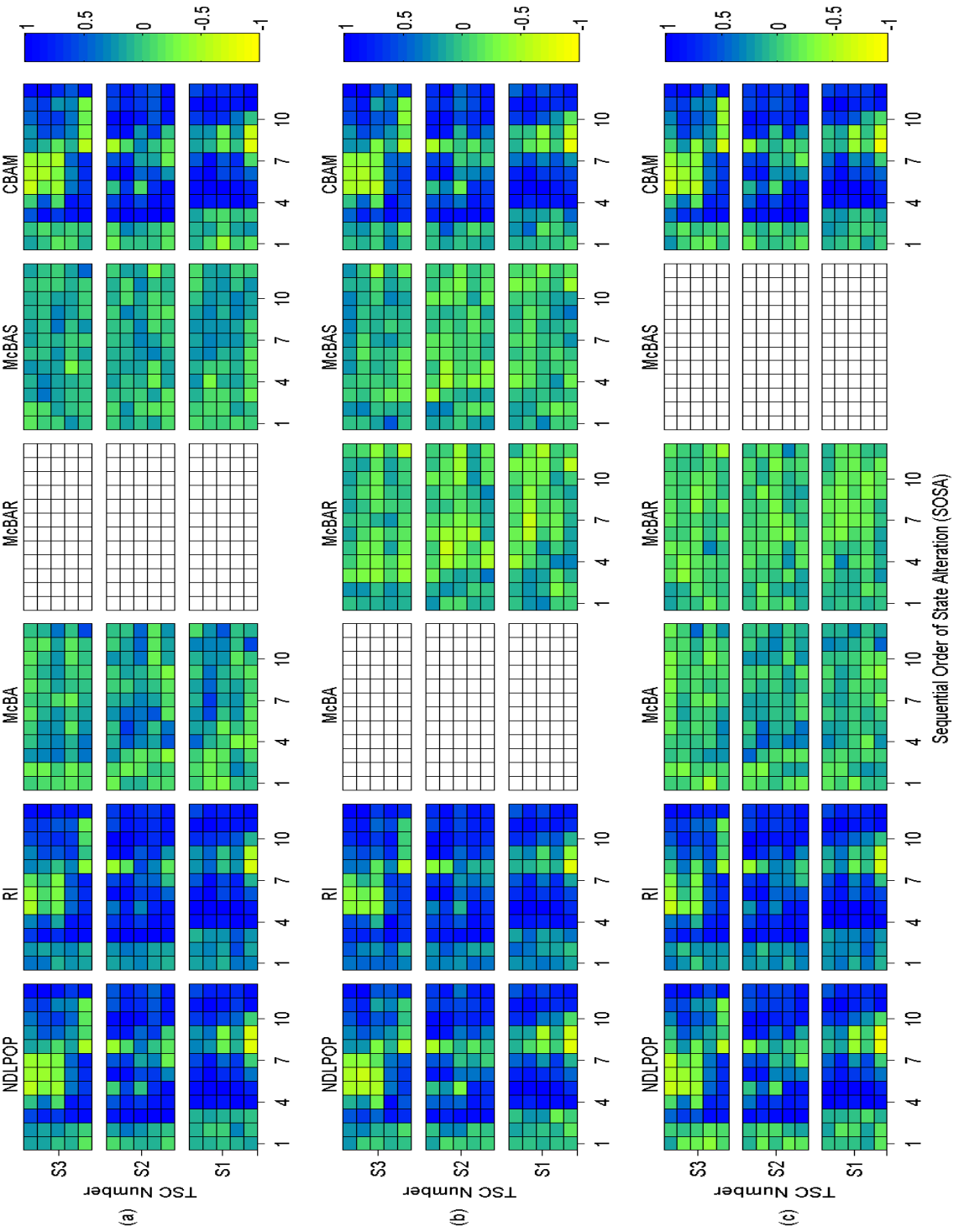



Figure 6.1: Dynamics of $\mathbf{E}^{\sigma}_{j,k}[\bullet]$ with $\delta = 3.0$ with (a) McBAR (b) McBA (c) McBAS as basis techniques

sub-problem $14_4^2$ represents $\mathbf{E}_{4,14}^{\sigma}\left[dSC(\text{McBAR}, \text{RI})\right]$. Considering that $\mathbf{E}_{j,k}^{\sigma}\left[dSC(T,S)\right]$ denotes average relative performance and the horizontal coordinate denotes SOSA, each row of small squares in a block expresses the dynamics of the average $\mathbf{E}_{j,k}^{\sigma}\left[dSC(T,S)\right]$ relative performance of a basis technique $T$ over another $S$.

In the $S_2$-RI block of Figure 6.1(a), the bottom to top rows of the vertical strip of small squares at the fourth SOSA correspond to sub-problems $2_4^2$, $8_4^2$, $14_4^2$, $20_4^2$ and $26_4^2$ (based from $N_2^{3.0} = \langle 2, 8, 14, 20, 26\rangle$). The types of changes in the MOEs, to where these sub-problems are set, are the elements of the vector,

$$\begin{aligned} V &= \langle Ct(2,4), Ct(8,4), Ct(14,4), Ct(20,4), Ct(26,4)\rangle \\ &= \langle 2,2,2,2,2\rangle\,, \end{aligned} \tag{6.7}$$

where Equation 4.3 is used to obtain the last line. This result implies that the rows in the vertical strip all correspond to change type 2 which, based on Table 4.2, denotes an increase in the total number of tasks in the MOEs. Now, using Equations 4.6 and 6.5, we obtain a compact form,

$$V = Ct\left(\langle 2, 8, 14, 20, 26\rangle, 4\right) = Ct\left(N_2^{3.0}, 4\right), \tag{6.8}$$

Following a similar approach and using Equation 4.7, $Nt\left(N_2^{3.0}, 4\right) = \langle 3, 4, 5, 6, 7\rangle$. To generalise, given a vertical strip at SOSA $j$ in a block at $S_i$ level, the types of changes and size of increase in the total number of tasks in the MOEs that correspond to the bottom to top rows of this strip can be determined from $Ct\left(N_i^{3.0}, j\right)$ and $Nt\left(N_i^{3.0}, j\right)$ respectively. Using Equation 4.8, $Pt\left(N_i^{3.0}\right) = \langle T_3, T_4, T_5, T_6, T_7\rangle$ which implies that the bottom to top rows correspond to the sub-problems constrained by the task-precedence networks $T_3$ to $T_7$ respectively.

Figures 6.1(b) and (c) differ from Figure 6.1(a) only in their corresponding basis techniques, which are McBA and McBAS respectively. For example, the fifth small square from the left (corresponds to the fifth SOSA) on the second row of the $S_3$-McBAR block in

Figure 6.1(b) corresponds to $\mathbf{E}^{\sigma}_{5,9}\left[dSC(\text{McBA, McBAR})\right]$ where index 9 is the problem label that corresponds to the second row, i.e. the second element of $N^{3.0}_{3}$. Note that, based on Table 4.5, all of problems from $\mathcal{L}^2$ found in $N^{3.0}_{1}$, $N^{3.0}_{2}$ and $N^{3.0}_{3}$ correspond to Equation A.9 with $\delta = 3.0$, such that the representation format of $\mathbf{E}^{\sigma}_{j,k}\left[\bullet\right]$ in Figures 6.1(a) to (c) is called the $F_{3.0}$ format.

The representation format of $\mathbf{E}^{\sigma}_{j,k}\left[\bullet\right]$ in Figures 6.2(a) to (c) only differs from that in Figures 6.1(a) to (c) by using,

$$N^{6.0}_{1} = \{4, 10, 16, 22, 28\}, \tag{6.9}$$

$$N^{6.0}_{2} = \{5, 11, 17, 23, 29\} \tag{6.10}$$

and

$$N^{6.0}_{3} = \{6, 12, 18, 24, 30\} \tag{6.11}$$

instead of $N^{3.0}_{1}$, $N^{3.0}_{2}$ and $N^{3.0}_{3}$ respectively. Note that the rows from the bottom to the top in blocks of Figure 6.2(a) to (c) correspond to problems from $\mathcal{L}^2$ with labels in $N^{6.0}_{1}$ to $N^{6.0}_{3}$ respectively. Based on Table 4.5, all of problems from $\mathcal{L}^2$ whose labels are found in sets $N^{6.0}_{1}$ to $N^{6.0}_{3}$ correspond to Equation A.9 with $\delta = 6.0$, such that the representation format of $\mathbf{E}^{\sigma}_{j,k}\left[\bullet\right]$ in Figures 6.2 (a) to (c) is called the $F_{6.0}$ format.

### 6.5.2 $S_1$-NDLPOP block

Let us now discuss the performance of techniques from $\mathcal{D} = \{\text{RI}, \text{NDLPOP}, \text{CBAM}, \text{McBA}, \text{McBAR}, \text{McBAS}\}$ for solving problems set in dynamic MOEs. In Figure 6.1(a), the first square from the left of the bottom row of $S_1$-NDLPOP block corresponds to sub-problem $1^2_1$ of problem $1^2$ from $\mathcal{P}^2$, based on the figure formatting explained above. This sub-problem is set in the first (indicated by the subscript in $1^2_1$) snapshot of a MOE. Based on the colorbar at the right side of the figure, the colour at the square denotes

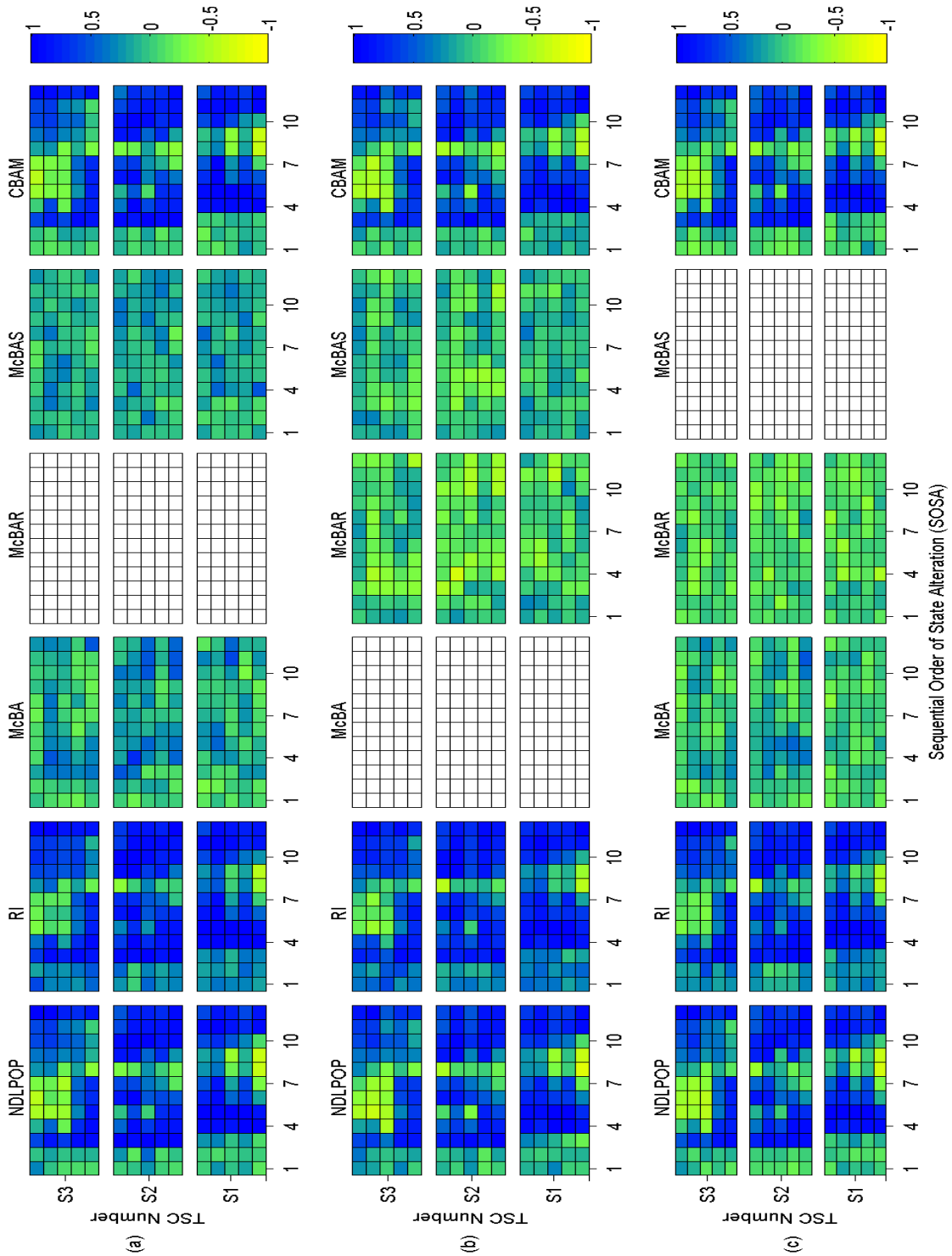



Figure 6.2: Dynamics of $\mathbf{E}^{\sigma}_{j,k}$ [•] with $\delta = 6.0$ with (a) McBAR (b) McBA (c) McBAS as basis techniques

$\mathbf{E}^{\sigma}_{j,k}\left[dSC(\text{McBAR}, \text{NDLPOP})\right] > 0$, where $j = 1$ (indicates first snapshot) and $k = 1$ (indicates label of problem $1^2$). Note that by Equation 4.3, $Ct(k, j) = 0$ denotes a type of change that involves a change in task duration only, based on Table 4.2. The result implies that the performance of McBAR is better ($\mathbf{E}^{\sigma}_{j,k}\left[\bullet\right] > 0$) than that of NDLPOP for solving the sub-problem $1^2_1$ of problem $1^2$ set in the MOE whose first SOSA has a change in task duration only.

The first to third vertical strips from the left of $S_1$-NDLPOP block show that $\mathbf{E}^{\sigma}_{S,Ns}\left[\right.$ $\left.dSC(\text{McBAR}, \text{NDLPOP})\right] > 0$, for all SOSAs $S$, $1 \leq S \leq 3$ and for all problems with labels $Ns \in N_1^{3.0}$ (labels which corresponds to squares that comprised one vertical strip at the $S_1$ blocks as explained in the last section). Considering that $Ct(Ns, S) = 0$, $\forall\ S, 1 \leq S \leq 3$ and $\forall\ Ns \in N_1^{3.0}$, then the result implies that McBAR performs better than NDLPOP for solving each of the first to third sub-problems of problems with labels in $N_1^{3.0}$; problems that are set in the MOEs whose first to third SOSAs have change in task duration only.

The fourth vertical strip of the $S_1$-NDLPOP block expresses an abrupt increase in the superiority of performance of McBAR over NDLPOP for solving the fourth sub-problems of problems with labels $Ns \in\ N_1^{3.0}$. The problems are set in MOEs whose fourth SOSAs have an increase in the total number of tasks only ( change type $2 = Ct(Ns, 4)$ based on Table 4.2). The fifth to seventh vertical strips of the block show that the superiority of McBAR over NDLPOP generally continues till the seventh sub-problems of the problems.

However, the first and third rows from the bottom of the eighth vertical strip of the $S_1$-NDLPOP block show that McBAR is inferior to NDLPOP for solving the eighth sub-problems of problems labelled 1 and 13 (which correspond to first and third rows, respectively, based on the figure formatting explained above). The problems are set in MOEs whose eighth SOSAs have change type $4 = Ct(1, 8) = Ct(13, 8)$, i.e. simultaneous changes in task duration and resource availabilities based on Table 4.2. The first and third rows from the bottom of the ninth vertical strip of the block show that the inferiority continues to the ninth sub-problems of the problems set in MOEs whose ninth SOSAs

have change type $0 = Ct(1, 9) = Ct(13, 9)$, the change in task duration only.

As implied above, the performance of McBAR is superior to that of NDLPOP for solving the first to third sub-problems of problems labelled 1 and 13 set in MOEs whose first to third SOSAs have change in task duration only. Thus, McBAR cannot be expected to be inferior to NDLPOP for solving the ninth sub-problems of problems labelled 1 and 13 set in the MOEs whose ninth SOSAs have similar types of changes, i.e. in task duration only. However, despite the similarity in the type of changes, the inferiority continues from the eighth (where the type of change is $4 = Ct(1, 8) = Ct(13, 8)$) to the ninth (where the type of change is $0 = Ct(1, 9) = Ct(13, 9)$) SOSAs of the MOEs. This suggests inertia in the performance of McBAR.

In contrast, the second, fourth and fifth rows from the bottom of the eighth and ninth vertical strips of $S_1$-NDLPOP block express the continuity of superiority in the performance of McBAR against NDLPOP for solving the eighth to the ninth sub-problems of problems labelled 7, 19 and 25 (which correspond to the rows respectively). These problems are set in MOEs whose eighth SOSAs have simultaneous changes in resource availability and task duration (change type 4 as explained above) and whose ninth SOSAs have change in task duration only (change type 0 as explained above).

Based on the tenth to the $12^{th}$ vertical strips of $S_1$-NDLPOP block, McBAR is generally superior in performance to NDLPOP for solving the tenth to the $12^{th}$ sub-problems of all problems with labels $Ns \in N_1^{3.0}$. These problems are set in MOEs whose tenth and $11^{th}$ SOSAs have changes type $5 = Ct(Ns, 10)$ and $3 = Ct(Ns, 11)$ for all $Ns \in N_1^{3.0}$, the change types that involve an increase in the total number of tasks. Further, the $12^{th}$ SOSAs of the MOEs has change type $0 = Ct(Ns, 12)$ for all $Ns \in N_1^{3.0}$, i.e. a change in task duration only.

Referring back to the fourth vertical strip of the $S_1$-NDLPOP block, the squares from bottom to top of this strip correspond to problems with labels 1, 7, 13, 19 and 25 (i.e. problems from $N_1^{3.0}$), respectively; problems that are set in MOEs whose fourth SOSAs have increase in the total number of tasks by 3, 4, 5, 6 and 7 (obtained from Equation 4.5

with $Nt(N_1^{3.0}, 4)$) respectively. Based on the colour of the squares in the strip and their corresponding increase in the total number of tasks, the superiority in performance of McBAR over NDLPOP for solving the fourth sub-problems (corresponding to the fourth vertical strip) of the problems is not affected by the size of the increase in the total number of tasks at the fourth SOSAs of the MOEs where the problems with labels in $N_1^{3.0}$ are set. Based on the discussions in Section 4.2.1, the sub-problems in problems with labels 1, 7, 13, 19 and 25 are constrained by PNTs illustrated in Figures 4.4, 4.1, 4.5 to 4.7 respectively. Thus, the superiority in performance of McBAR to NDLPOP is not affected by the type of PNT that constrained the sub-problems. An analogous conclusion can be drawn from the sixth vertical strip of the block, whose bottom to top rows correspond to 2, 4, 3, 2 and 1 (obtained from $Nt(N_1^{3.0}, 6)$) new tasks respectively; and from the tenth vertical strip whose bottom to top rows correspond to 2, 1, 1, 1 and 1 (obtained from $Nt(N_1^{3.0}, 10)$) new tasks respectively. Note, however, that the PNTs are related by being derived from the original PNT (illustrated in Figure 2.9) as explained in Section 4.2.1.

### 6.5.3 $S_2$-NDLPOP block

The first and second vertical strips of the $S_2$-NDLPOP block illustrate that $\mathbf{E}^{\sigma}_{S,Ns}\,[dSC(\,\text{McBAR}, \text{NDLPOP})] > 0,\ \forall\ S, 1 \leq S \leq 2$ and $\forall\ Ns \in N_2^{3.0} = \langle 2, 8, 14, 20, 26\rangle$. This implies that the performance of McBAR is better than that of NDLPOP for solving each of the first and second sub-problems of problems with labels in $N_2^{3.0}$; problems set in MOEs whose first and second SOSAs have change type $0 = Ct(Ns, S)$, $\forall\ S, 1 \leq S \leq 2$ and $\forall\ Ns \in N_2^{3.0}$, i.e. the change in task duration only.

Unlike the case of the $S_1$-NDLPOP block, the abrupt increase in the superiority in performance of McBAR over NDLPOP occurs at the third SOSA. This abruptness is expressed by the third vertical strip of the $S_2$-NDLPOP block. The bottom to top rows of this strip correspond (as explained in Section 6.5.1) to the problems with labels in $N_2^{3.0}$, respectively, set in the MOEs whose third SOSAs have change type $\langle 2, 2, 2, 2, 2\rangle = Ct(N_2^{3.0}, 3)$ which is an increase in the total number of tasks based.

Based on the rows of the eighth vertical strip of the $S_2$-NDLPOP block, the performance of McBAR is generally inferior to that of NDLPOP for solving the eighth sub-problem of all problems with labels $Ns \in N_2^{3.0}$. The problems are set in the MOEs whose eighth SOSAs have change type $4 = Ct(Ns, 8)$ which is a simultaneous changes in resource availability and duration of tasks. Unlike the case of the $S_1$-NDLPOP block, based on the ninth vertical strip of $S_2$-NDLPOP block, the inferiority is not continued to the next (ninth) SOSA of the MOEs; each undergoes change type $6 = Ct(Ns, 9)$ that involves an increase in the total number of tasks. Based on the ninth to $12^{th}$ vertical strips, the performance of McBAR remains superior to that of NDLPOP for solving sub-problems set at the ninth to $12^{th}$ snapshots of the MOEs.

In the third vertical strip of the $S_2$-NDLPOP block, the squares from bottom to top correspond respectively to the third sub-problem of problems with labels 2, 8, 14, 20 and 26 (labels from $N_2^{3.0}$), problems set in MOEs whose third SOSAs have 3, 4, 5, 6 and 7 (derived from $Nt(N_2^{3.0}, 6)$) new tasks respectively. Based on the colour of the squares and their corresponding increases in the total number of tasks, the superiority in performance of McBAR over NDLPOP for solving the sub-problems is almost constant with respect to the size of the increase in the total number of tasks. Following the explanation in the last section, this superiority is also almost constant with respect to the types of PNT that constrain the sub-problems. Analogous conclusions can be drawn from the sixth, ninth and tenth vertical strips of the $S_2$-NDLPOP block.

### 6.5.4 $S_3$-NDLPOP block

The first two vertical strips of the $S_3$-NDLPOP block express the dynamics, of the superiority in performance of McBAR over NDLPOP, being similar to that expressed by the first two vertical strips of the $S_2$-NDLPOP block. The third to fifth rows from the bottom of the fifth vertical strip of the $S_3$-NDLPOP block illustrate that the performance of McBAR is inferior to that of NDLPOP for solving the fifth sub-problems of problems labelled 15, 21 and 27 set in MOEs whose fifth SOSAs have change type 4

(obtained from $\langle 4,4,4\rangle = Ct(\langle 15,21,27\rangle, 5)$) which is a simultaneous change in task duration and resource availabilities (based on Table 4.2). This inferiority continues to be expressed at the third to the fifth rows from the bottom of the sixth and seventh vertical strips of the $S_3$-NDLPOP block. The problems that correspond to these rows are set in MOEs whose sixth and seventh SOSAs have change type 0 which is obtained from $\langle 0,0,0\rangle = Ct(\langle 15,21,27\rangle, 6)$ and $\langle 0,0,0\rangle = Ct(\langle 15,21,27\rangle, 7)$, i.e. a change in task duration. Based on the explanation of the last sub-section, this result suggests inertia on the relative performance of McBAR over NDLPOP.

The bottom row of the eighth vertical strip of the $S_3$-NDLPOP block also shows the inferiority in performance of McBAR over NDLPOP for solving the sub-problem of problem labelled 3 set in the MOE whose eighth SOSA has a change type $5 = Ct(3,8)$ which is a simultaneous change in resource availability and the total number of tasks. The bottom rows of the ninth to $11^{th}$ vertical strips express the continuity of this inferiority for solving the ninth to $11^{th}$ sub-problems of problem labelled 3. All other unaccounted squares in the $S_3$-NDLPOP block show the superiority in performance of McBAR over NDLPOP for solving sub-problems that correspond to these squares.

In the third vertical strip of the $S_3$-NDLPOP block, the squares from bottom to top correspond, respectively, to the third sub-problems of problems with labels 3, 9, 15, 21 and 27 ( labels from $N_3^{3.0}$) respectively; problems set in MOEs whose third SOSAs have 3, 4, 5, 6 and 7 (obtained from $Nt(N_2^{3.0}, 6)$) new tasks respectively. Based on the colour of the squares and their corresponding increases in the total number of tasks, the superiority in performance of McBAR over NDLPOP for solving the sub-problems is almost constant with respect to the size of the increase in the total number of tasks. Based on the explanation in Section 6.5.2, this superiority is also almost constant with respect to the type of PNT that constrains the sub-problems. Analogous conclusions can be drawn from the $12^{th}$ vertical strip of the $S_3$-NDLPOP block.

### 6.5.5 Other Blocks

Let us now consider the remaining coloured blocks in Figure 6.1(a). The $S_1$, $S_2$ and $S_3$-RI blocks are generally similar in colour variation (but not in magnitude) to those of the $S_1$, $S_2$ and $S_3$-NDLPOP blocks respectively. Likewise, the $S_1$, $S_2$ and $S_3$-CBAM blocks are generally similar in colour variations to those of the $S_1$, $S_2$ and $S_3$-NDLPOP blocks respectively. This implies that the variations in McBAR's superiority over RI and over CBMA are generally similar to the variation in McBAR's superiority over NDLPOP.

The notable exceptions to the last conclusion are as follows. The squares from the bottom to the top of the first and second vertical strips of the $S_2$-CBAM, $S_3$-CBAM, $S_2$-NDLPOP, $S_3$-NDLPOP, $S_2$-RI and $S_3$-RI blocks show McBAR as equally, slightly, and highly superior to CBAM, NDLPOP and RI, respectively. A similar exception can be seen in the performance of McBAR over CBAM, NDLPOP and RI in the squares from the bottom to the top of the first to the third vertical strips of the $S_1$-CBAM, $S_1$-NDLPOP and $S_1$-RI blocks respectively. Following similar reasoning as in Section 6.5.2, the types of changes that correspond to the squares are only in task durations.

The squares from bottom to top in the first vertical strip of the $S_1$-McBA block show slight superiority in the performance of McBAR over McBA in solving the first sub-problems of problems with labels found in $N_1^{3.0}$. The squares correspond to the labels respectively, and the problems are set in MOEs whose first SOSAs have change in task duration (a change type 0 which is obtained from $\langle 0,0,0,0,0\rangle = Nt(N_1^{3.0},1)$). Analogous conclusions can be drawn from the second and third vertical strips of the $S_1$-McBA block.

The fourth to the last vertical strips of the $S_1$-McBA block generally express the superiority in performance of McBAR over McBA for solving sub-problems that correspond to the squares in these strips. Recall from Section 5.9 that McBAR differs from McBA in its random component only. The superiority of McBAR to McBA suggests the effectiveness of this component. Based on the colour variation of the $S_1$-McBA block, generally similar performance variation are expressed in the $S_2$ and $S_3$-McBA blocks. Further, the variation

in McBAR's performance over McBAS expressed in the $S_1$, $S_2$ and $S_3$-McBAS blocks are generally similar to the variation in McBAR's performance over McBA expressed in the $S_1$, $S_2$ and $S_3$-McBA blocks. Note that, although the variations are generally similar, the magnitude of McBAR's performance over McBAS is greater than the magnitude of its performance over McBA.

### 6.5.6 Other Figures

Let us now discuss the performance variation, expressed in Figures 6.1 (b) and (c), of the techniques from $\mathcal{D}$. Note first that the blank block in Figure 6.1(b) is under the heading McBA. Thus, as explained in Section 6.5.1, McBA is the basis technique in this figure. The colour variations between blocks under NDLPOP, RI, CBAM and McBAS headings in Figure 6.1(a) and blocks under similar headings in Figure 6.1(b) are respectively generally similar which implies that the variations in performance, expressed in these blocks, of McBA (the basis technique) over NDLPOP, RI, CBAM and McBAS are generally similar to those of McBAR.

The basis technique in Figure 6.1(c) is McBAS. The colour variation between blocks under NDLPOP, RI and CBAM headings in Figure 6.1(a) and blocks under similar headings in Figure 6.1(c) are respectively generally similar which implies that the variations in performance, expressed in these blocks, of McBAS over NDLPOP, RI and CBAM are generally similar to those of McBAR.

Note that as the differential set coverage $dSC(A, B)$ in Equation 2.8 is anti-symmetric with respect to its arguments $A$ and $B$, so does the average $\mathbf{E}^{\sigma}_{S,Ns}[dSC(A, B)]$. For example, $\mathbf{E}^{\sigma}_{S,Ns}[dSC(\text{McBAR}, \text{McBA})] = -\mathbf{E}^{\sigma}_{S,Ns}[dSC(\text{McBA}, \text{McBAR})]$. The variations in the performance of McBAR over McBA expressed in $S_1$, $S_2$ and $S_3$-McBAR blocks of Figure 6.1(a) are respectively opposite (due to the anti-symmetry of $\mathbf{E}^{\sigma}_{S,Ns}[\bullet]$) to the variations in the performance of McBA over McBAR expressed in $S_1$, $S_2$ and $S_3$-McBA blocks of Figure 6.1(b). A similar relationship holds between McBAR and McBAS.

The variations of performance expressed in blocks of Figure 6.1 are respectively generally similar to those expressed in blocks of Figure 6.2. Considering that problems corresponding to rows in blocks of Figures 6.1 and 6.2 correspond respectively to $\delta = 3.0$ and $\delta = 6.0$, as discussed in Section 6.5.1, then the variations in the performance of the techniques from $\mathcal{D}$ are generally similar on the two amounts of $\delta$ in Equation A.9 which models change in duration of tasks associated with the figures.

## 6.6 Conclusions and Future Work

Under the limited method, the solutions to problems from $\mathcal{L}^2$ obtained by techniques from $\mathcal{T}$ demonstrated the performance of McBAR to be superior to that of the other techniques from $\mathcal{T}$, a performance defined in terms of an average of differential set coverage. This conclusion is true at any of the two values of $\delta$ used above as the parameter of Equation A.9 which models changes in task durations in MOEs where the problems are set, at any type (listed in Table 4.2) of simultaneous changes in the MOEs and at any form of PNT utilised in this chapter. For each problem from $\mathcal{L}^2$, the performance of McBAR generally becomes and remains superior, respectively, at and after the first occurrence of new tasks in the MOEs.

By comparing the performance of McBAR for solving problems from $\mathcal{L}^2$ to those of other techniques from $\mathcal{T}$, each of its sub-algorithms was supported as being legitimate. Based on the already-presented conclusions in this section, the first and fourth thesis' goals are accomplished through the limited method.

Now, the above conclusions are only true on problems from $\mathcal{L}^2$, and each has two conflicting objectives. The next chapter will investigate the performance of the extended core algorithm of McBAR for solving problems from $\mathcal{L}^3$, where each problem has three conflicting objectives.

As discussed in Section 1.4.1, the problems from $\mathcal{L}^2$, which have been considered in this chapter, have small or large consecutive sub-problem attribute difference magnitude

(CSADM) (explained in Section 1.4.1) and are few in number. The superiority of the performance and the legitimacy of the sub-algorithms of McBAR will be investigated in Chapter 8 on problems with small CSADM only, but in greater numbers than those considered in this chapter.

There are two future research directions envisioned in relation to the topics investigated in this chapter; they are to investigate the performance of McBAR (a) at other values of the above-considered $\delta$ and (b) at forms of PNT other than those considered in this chapter.

# Performance of McBAR on $\mathcal{L}^3$

## Chapter 7

The last chapter demonstrated the superior performance of the reactive-predictive McBAR for solving problems from $\mathcal{L}^2 \subset \mathcal{P}^2$, each of which has two conflicting objectives – the minimisation of schedule makespan and cost. This present chapter investigates the performance of the robust reactive-predictive McBAR-P (the extended core algorithm of McBAR) for solving problems from $\mathcal{L}^3 \subset \mathcal{P}^3$, each of which has three conflicting objectives – the two mentioned above and the probability of a schedule to become infeasible (defined in Section 1.2). It begins by elaborating the algorithm of McBAR-P in Section 7.1. Then the averages and models useful for investigating the characteristics of McBAR-P and McBAR are described in Section 7.2. Using some of them, the performance enhancing parameters of McBAR-P are determined in Section 7.3. McBAR-P and McBAR are compared in Sections 7.4 to 7.6 with respect to the following aspects: (a) the financial relevance of the solutions/schedules that McBAR and McBAR-P obtained for solving the problems from $\mathcal{L}^2$ and $\mathcal{L}^3$ respectively; (b) the difference in the schedule makespans; (c) and the difference in the schedule implementation costs. The variation in the results of the comparisons due to the dynamics of the MOE, in which the problems are set, is explored in Sections 7.5 and 7.6. The investigations are undertaken to manifest the significant qualities of McBAR-P despite its being an extension, i.e. to achieve the second thesis' goal as defined in Section 1.3. Finally, Section 7.7 gathers the conclusions and presents directions for the future research.

## 7.1 McBAR-Proactive

As discussed in Section 2.4.2, a *reactive-predictive* approach creates a baseline schedule which it revises every time change occurs in an environment in which this schedule is implemented [40]. However, the revision could incur penalties in breaching contracts made to material or labour suppliers aside from costs for material, fuel and labour for example and in executing schedule's remaining stages after this revision. A *Robust proactive* approach in scheduling, on the other hand, anticipates future events to create schedules robust to changes in the environment [104, 106, 117, 208]. However, when changes in the environment are extreme the created schedules could become infeasible [117]. Intuitively, the best way of making schedules could be to combine the robust proactive and reactive-predictive approaches to gain the benefits offered by both of these approaches. As noted in Section 2.4.2, the combination of these approaches is referred to as a robust reactive-predictive approach; McBAR-P is of this type.

McBAR-P differs from McBAR primarily in being robust proactive. To create schedules, it anticipates the implementation failures of these schedules due to the changes in task duration that can trigger violation of resource constraints (as explained in Section 2.7). These schedules are solutions to the sub-problems of problems from $\mathcal{L}^3$. In terms of the algorithm, McBAR-P differs from McBAR only in the following: if the current change in a MOE, in which a problem from $\mathcal{L}^3$ is set, is only on duration of its tasks, if the unused resources in the MOE are not required to be used by the tasks and if the constraints of the resources are not violated despite the current change, then McBAR-P will not revise the schedule implemented on the last state of the MOE. Otherwise, McBAR-P will revise the schedule, but still with anticipation of future resource constraint violations. On the contrary McBAR will revise the schedules it created regardless of change type in the MOE.

## 7.2 Models and Averages

The following subsections discuss a model and averages useful to demonstrate the characteristics of McBAR and McBAR-P for solving dynamic problems from $\mathcal{L}^2$ and $\mathcal{L}^3$, respectively. The model is of the dynamic characteristics of the EA-based McBAR and McBAR-P with respect to their evolutionary cycles. The averages are of the makespan and cost of the non-dominated schedules/solutions to the problems, solutions found by the McBAR and McBAR-P approaches.

### 7.2.1 Convergence Model

Suppose McBAR solves the static sub-problem $p_j^2$ of problem $p^2$ from $\mathcal{L}^2$. The evolutionary process of McBAR begins with an initial population (as elaborated in Section 5.6.6) whose NDS has elements that form a hypersurface of certain hypervolume. At each new cycle of this process, McBARP obtains (through the Solution Production method explained in Section 5.6) a new NDS whose elements could form a new hypersurface. As this process continues, the instantaneous hypervolume of the NDS could vary with the number $t$ of evolutionary cycles, as exemplified in Figure 7.1 where the horizontal axis is the evolutionary cycle. We then associate a hypervolume trajectory to a sub-problem. This hypervolume trajectory is modelled as,

$$y\left(t\right) = DC - \frac{e^{P(t)}}{t+1} \tag{7.1}$$

where $DC$ is a constant and is the hypervolume averaged over $t_\phi$ to $t_\omega$ evolutionary cycles; while,

$$P\left(t\right) = \sum_{n=0}^{D_g} c_n t^n \tag{7.2}$$

is a polynomial whose coefficients are $c_n$; and the polynomial degree $D_g$ is chosen to avoid under and over fitting of data $d\left(t\right)$ of hypervolume of non-dominated solutions determined by McBAR at the $t^{th}$ evolutionary cycle. The polynomial $P\left(t\right)$ fits $y\left(t\right)$ best to $d\left(t\right)$ in

the least square sense, i.e.,

$$\min_{c_n} \|ln\left|\left(DC - d\left(t\right)\right)\left(t+1\right)\right| - P\left(t\right)\|^2 \tag{7.3}$$

The convergence rate of hypervolume is defined as the number of cycles $\tau$ at which hypervolume is $\xi$ percent close to its asymptotic value,

$$\tau = y^{-1}\left(DC - e^{c_0}\left[1-\xi\right]\right) \tag{7.4}$$

where $\xi$ is determined to be 0.95. The topics discussed in this section also apply to McBAR-P that solves sub-problem $p_j^3$ of problem $p^3$ from $\mathcal{L}^3$. Note that the labels of problems $p^2$ and $p^3$ being solved by McBAR and McBAR-P, respectively, refer to the letter "p" which takes values from Table 4.5. Further, problem $p^3$ only differs from $p^2$ by having the additional minimisation of the probability of a schedule to become infeasible.

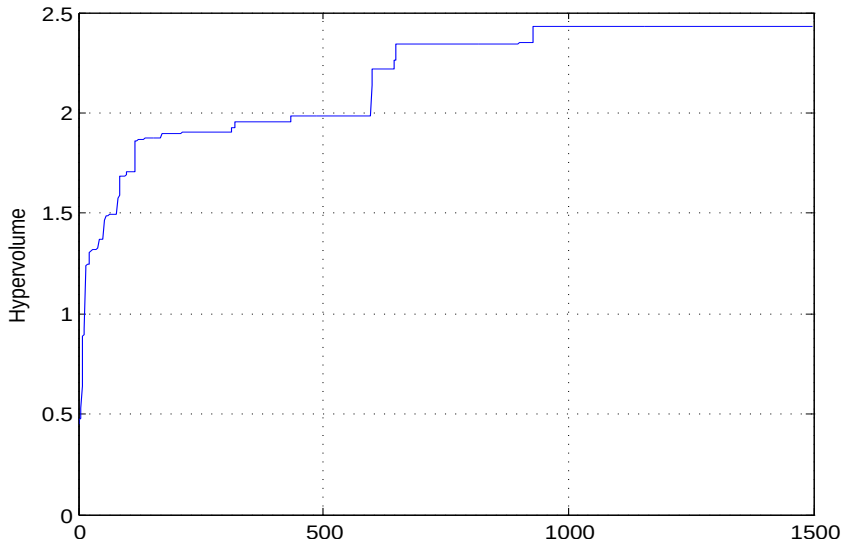



Figure 7.1: Sample evolution dynamics of hypervolume

### 7.2.2 Difference in Average Makespan and Cost

Let us now define averages related to the schedule cost of implementation and duration/ makespan. The makespans of schedules as solutions to static sub-problems $x_e^2$ and $x_e^3$ are denoted as $\tau^r(x,e)$ and $\tau^p(x,e)$ respectively; these sub-problems are set in the $e^{th}$

snapshot of a MOE. Let $\mathbf{E}_d[\bullet]$ be an averaging operator over non-dominated solutions to the problem related to its argument. For example, $\mathbf{E}_d[\tau^p(x,e)]$ is the average makespan of non-dominated solutions to $x_e^3$. The average $\mathbf{E}_d[\bullet]$ is in turn averaged $\mathbf{E}_s[\bullet]$ over MOE simulations where we define a compound average operator, $\mathbf{E}_{d,s}[\bullet] = \mathbf{E}_s[\mathbf{E}_d[\bullet]]$. Let us consider the difference,

$$dM_s(x,e) = \mathbf{E}_{d,s}[\tau^r(x,e)] - \mathbf{E}_{d,s}[\tau^p(x,e)] \tag{7.5}$$

A positive value of this difference implies that, on average over the non-dominated solutions and MOE simulations, the makespans of schedules as non-dominated solutions to $x_e^2$ determined by McBAR is longer than the makespans of non-dominated solutions to $x_e^3$ determined by McBAR-P.

The total cost $f_{cs}$ of moving resources between locations of tasks in a MOE is defined in Equation A.8. Let $\gamma^r(x,e)$ and $\gamma^p(x,e)$ be the total costs of moving tasks in order to implement schedules as solutions to static sub-problems $x_e^2$ and $x_e^3$. Let us consider the difference,

$$dC_s(x,e) = \mathbf{E}_{d,s}[\gamma^r(x,e)] - \mathbf{E}_{d,s}[\gamma^p(x,e)] \tag{7.6}$$

A positive value of this difference implies that, on average over non-dominated solutions and MOE simulations, the costs of schedules as non-dominated solutions to $x_e^2$ determined by McBAR is higher than the costs of non-dominated solutions to $x_e^3$ determined by McBAR-P.

The difference $dC_s(x,e)$ compares costs, a comparison that resembles what had been done in [199] where the difference in cost of executing a ship berthing schedule created under a robust proactive approach is compared to that of a reactive-predictive approach. This type of comparison is relevant to demonstrate the significance of McBAR-P.

## 7.3 Optimal Parameters

The crossover and mutation rates of the evolutionary process of McBAR-P are determined as follows: Consider the Cartesian product,

$$C = \chi \otimes \mu \equiv \{\chi_i, \mu_j\} \tag{7.7}$$

where $i$ and $j$ are the row and column indices of $C$ respectively; $\chi = [.1, .2, \ldots, .9]$ and $\mu = [0.01\,,\ 0.0588,\ 0.1075,\ 0.1563,\ 0.2050,\ 0.2538,\ 0.3025,\ 0.3513,\ 0.4000,\ 0.45,\ 0.5,\ 0.55,\ 0.6,\ 0.65,\ 0.7,\ 0.75,\ 0.8,\ 0.850]$. For a given element $(\chi_i, \mu_j) \in C$, the evolutionary process of McBAR-P is endowed with a crossover rate of $\chi_i$ and a mutation rate of $\mu_j$ to determine the population $P_{28}(e, s, \chi_i, \mu_j)$ of non-dominated solutions to sub-problem $28_e^3$ (refer to notation in Section 1.2) set in the $s^{th}$ MOE simulation. Then the hypervolume $\mathcal{H}\{P_{28}(e, s, \chi_i, \mu_j)\}$ of the population $P_{28}(e, s, \chi_i, \mu_j)$ is determined using the method explained in Section 2.5.1. The hypervolume $\mathcal{H}\{P_{28}(e, s, \chi_i, \mu_j)\}$ is determined for each of the 30 MOE simulations; and for each of the snapshots taken at the zeroth to the $12^{th}$ SOSA of the MOE. The determined hypervolumes are then averaged $\mathbf{E}_{e,s}[\bullet]$ over the snapshots and the MOE simulations to obtain the average hypervolume,

$$\overline{\mathcal{H}}_{28}\{\chi_i, \mu_j\} = \mathbf{E}_{e,s}[\mathcal{H}\{P_{28}(e, s, \chi_i, \mu_j)\}] \tag{7.8}$$

This average is computed for every element of $C$. The element $(\chi^{max}, \mu^{max})$ that corresponds to the maximum average hypervolume,

$$(\chi^{max}, \mu^{max}) = \max_{(\chi,\mu)\ \in\ C_p} \overline{\mathcal{H}}_{28}\{\chi_i, \mu_j\} \tag{7.9}$$

has its crossover $\chi^{max}$ and mutation $\mu^{max}$ rates used in the evolutionary process of McBAR-P to determine solutions to all sub-problems of all problems from $\mathcal{L}^3$, solutions considered in the remaining portion of the thesis.

The problem labelled 28 in Table 4.5 is from $\mathcal{L}^3$ and is the chosen problem to determine

the hypervolume-optimising rates due to its (a) the sequence of changes labelled $S_1$ that has the earliest successive changes (from the sixth to the eighth SOSA as can be inferred from Table 4.3) in resource availability and in the total number of tasks among all of the TSCs in Table 4.3; (b) TNIS of $T_7$ that has the most number of new tasks among all of the types of TNIS in Table 4.4; and (c) the use of $\delta$ equal to 6.0 for the task duration change model in Equation A.9, an amount of $\delta$ that is largest among considered values of $\delta$. All these extreme environmental changes will intuitively give McBAR-P the most difficulty in searching for the NDS of solutions that has a large average hypervolume. Considering that these hypervolume-optimising rates are derived from the performance of McBAR-P for solving problem 28 set in extreme environmental changes, it is legitimate to assume that if the evolutionary process of McBAR-P uses the hypervolume-optimising rates then McBAR-P will also determine an optimal average hypervolumed NDS of solutions to any sub-problem of any problem from $\mathcal{L}^3$ set in MOEs with less extreme changes than that of problem 28. This assumption is considered to justify problem 28 as the basis of the search of the hypervolume-optimising rates.

The average hypervolume $\overline{\mathcal{H}}_{28}\{\chi_i, \mu_j\}$ in Equation 7.8 is a function of crossover $\chi_i$ and mutation $\mu_j$ rates, where $(\chi_i, \mu_j) \in C_p$ (refer to Equation 7.7) indexed $i$ and $j$. This function is plotted in Figure 7.2(a) where $\chi$ and $\mu$ are the vertical and horizontal axis respectively. From the figure, $\overline{\mathcal{H}}_{28}\{\chi_i, \mu_j\}$ is maximum at crossover rate $\chi^{max}$ of .7 and mutation rate $\mu^{max}$ of .3025. These parametric values are used by McBAR-P to obtain the solutions to sub-problems of problems from $\mathcal{L}^3$ considered in the remaining portion of the thesis. Further, the figure shows high average hypervolume $\overline{\mathcal{H}}_{28}\{\chi_i, \mu_j\}$ at $.2 < \mu < .7$ and at a wide range of $\chi$. This result implies a need of a high mutation rate for McBAR-P to have a high performance (in terms of hypervolume) for solving the sub-problem $28^3_e$, where $0 \leq e \leq 12$. The figure also shows a less influential crossover rate for McBAR-P to determine large average hypervolumed non-dominated solutions to the sub-problems.

Aside from the average hypervolume, an average convergence rate is also analysed at each element of $C$ in Equation 7.7. For a given element $(\chi_i, \mu_j) \in C$ of Equation 7.7,

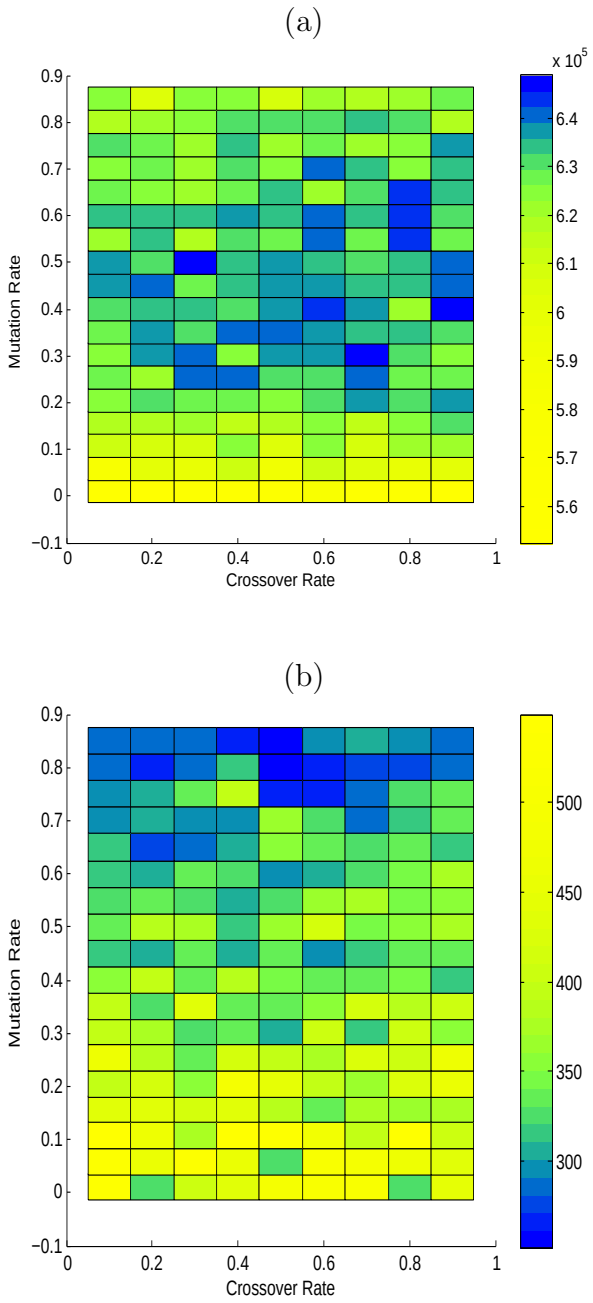



Figure 7.2: Crossover and mutation rate as a function of (a) hypervolume (b) convergence rate

the evolutionary process of McBAR-P is endowed with crossover and mutation rates of $\chi_i$ and $\mu_j$, respectively, to determine the solutions to sub-problem $28_e^3$ (refer to notation in Section 1.2) set in the $s^{th}$ MOE simulation. In the process of determining the solutions, the convergence rate $\tau_{28}(e, s, \chi^i, \mu^j)$ is also determined following the approach explained in Section 7.2.1. This convergence rate is determined for each of the 30 MOE simulations

described in Section 4.4; and for each of the snapshots taken at the zeroth to the $12^{th}$ SOSA of the MOE. The determined convergence rates are then averaged $\mathbf{E}_{e,s}[\bullet]$ over the snapshots and over the MOE simulations to obtain the average convergence rate,

$$\overline{\mathcal{C}}_{28}\{\chi_i, \mu_j\} = \mathbf{E}_{e,s}[\tau_{28}(e, s, \chi_i, \mu_j)] \tag{7.10}$$

This average convergence rate is a function of crossover $\chi_i$ and mutation $\mu_j$ rates of McBAR-P. This function is illustrated in Figure 7.2(b) which shows a small average convergence rate as the mutation rate increases. Based on the definition of convergence rate in Section 7.2.1, this result implies that as the mutation rate utilised by McBAR-P increases the number of evolutionary generations in McBAR-P decreases, i.e., the number required for the average hypervolume to reach a determined asymptote. This hypervolume is of the non-dominated solutions to sub-problems $28_e^3$ determined by McBAR-P. And, it is averaged over the snapshot $e$, where $0 \leq e \leq 12$, and over 30 MOE simulations. The figure shows the crossover rate to be less significant on the convergence rate of the average hypervolume.

## 7.4 Relevance of Proactiveness

To be able to determine the dominance of a solution to a sub-problem of a problem from $\mathcal{L}^2$ over a solution to a sub-problem of a problem from $\mathcal{L}^3$, the definition in Section 2.5 of the dominance operator $\preceq$ (used to determine set coverage) requires that these two sub-problems have a similar number of objectives. However, for any given problem labelled $p$ in Table 4.5 and any snapshot $e$, the number of objectives of sub-problem $p_e^2$ is different to that of $p_e^3$. On the other hand, based on the algorithm explained in Section 2.5.1, the computation of a hypervolume requires a common reference point (e.g. "$\diamond$" in Figure 2.12(f)) which cannot be established for the solutions to the sub-problems $p_e^2$ and $p_e^3$ due to the different number of objectives in these problems. Thus, the performances of McBAR and McBAR-P for solving the sub-problems $p_e^2$ and $p_e^3$, respectively, cannot

be compared using set coverage or hypervolume. Consequently, McBAR and McBAR-P are compared with respect to the following aspects: (a) the financial relevance of the solutions/schedules that McBAR and McBAR-P obtained in this section; (b) the difference in the makespans of the solutions/schedules considered in Section 7.5; (c) and the difference in the implementation cost of the solutions/schedules explored in Section 7.6.

Let us now explore the relevance of the robust reactive-predictive McBAR-P approach. Consider a schedule as the solution to sub-problem $12_5^3$ (fifth sub-problem of problem 12 from $\mathcal{L}^3$) depicted in Figure 4.2, and determined through the McBAR-P approach. Based on Table 4.5, problem 12 of $\mathcal{L}^3$ has $S_3$ TSC which, based on Table 4.3, has type 4 of changes (simultaneous changes in task duration and resource availability based on Table 4.2) at the fifth snapshot (taken at 13 time units) in which sub-problem $12_5^3$ is set. Now, at the next (sixth) SOSA of the MOE, based on Table 4.3, there is a change in task duration only (type 0 based on Table 4.2). A schedule as solution to sub-problem $12_6^3$ (set in the sixth snapshot of the MOE) is depicted in Figure 4.3. As is evident from Figure 4.2, task 17 does not overlap in time with task 16 while in Figure 4.3 it does, due to the task duration change. A similar relationship holds between pairs of tasks 32 and 15. In Figure 4.3, tasks in any of these pairs simultaneously utilise a similar type of resource at the period they overlap. However, there is no violation of any resource constraint (listed in Section 4.2), and no unused resource in the fifth snapshot is utilised in the sixth snapshot despite the overlap. Thus, based on the algorithm of McBAR-P explained in Section 7.1, the schedule for the fifth snapshot is retained for the sixth snapshot of the MOE. As pointed out above, the revision of the schedule can incur high cost, e.g. the penalty due to contract infringement with material and labour suppliers [191], such that the retention of the schedule despite environmental changes could be cost-saving. Thus, the demonstrated retention of the schedule supports the financial relevance of McBAR-P.

The correspondence between TSCs and problems from $\mathcal{L}^3$ is found in Table 4.5. For a given TSC, the number of moments (labelled in Table 4.3 by SOSA) at which the MOE

undergoes changes of type zero (i.e., change only on task duration, based on Table 4.2) is counted, the MOE in which the problem that corresponds to the TSC is set. The sum of this count, over all TSCs that correspond to all problems from $\mathcal{L}^3$, is 180. Now, each MOE is simulated (described in Section 4.4) 30 times. On the average over these simulations, there are only 2.4 moments – among the 180 moments, i.e. 1.33% – whose corresponding schedules are retained. These schedules are non-dominated solutions to sub-problems set in snapshots taken at the counted moments of changes. If the above-stated cost penalty per schedule revision is extremely high, the McBAR-P approach could still be relevant despite its rare schedule retention rate; this supports the relevance of McBAR-P.

## 7.5 Difference in Makespan

As explained in Section 7.2.2, $dM_s(x, e)$ is the difference between the average makespans of non-dominated solutions/schedules determined by McBAR and the average makespans of non-dominated solutions/schedules determined by McBAR-P for solving $x_e^2$ and $x_e^3$ sub-problems, respectively, where $x_e^K$ (refer to Section 1.2 for notation) is set in the $e^{th}$ snapshot of a MOE and is a sub-problem of problem $x^K$ from $\mathcal{L}^K$, $K = 2, 3$. The $F_{6.0}$ formatted (explained in Section 6.5.1) Figure 7.3(a) is a plot of $dM_s(x, e)$ at all pairs of problems $x$ and SOSAs $e$, where problem $x$ is in Table 4.5 and has task duration variation that follows Equation A.9 with $\delta = 6.0$; and $0 \leq e \leq 12$. It shows that at 92.8% of the pairs $dM_s(x, e)$ is negative. Based on Equation 7.5, at 92.8% of the pairs, the average $\mathbf{E}_{d,s}[\bullet]$ makespans of schedules obtained through McBAR is less than that of schedules obtained through McBAR-P. The difference $dM_s(x, e)$ is more negative after the sixth SOSA of the MOE than otherwise, regardless of problems in the pairs. After the sixth SOSA, based on Table 4.3 in conjunction with Table 4.2, severe (defined in Section 4.2.1) types of changes that occur in the MOE are prevalent. The $F_{3.0}$ formatted Figure 7.3(b) shows an approximately similar $dM_s(x, e)$ plot to that of $F_{6.0}$ formatted Figure 7.3(a). It shows that at 95.6% of its corresponding pairs $dM_s(x, e)$ is negative. Note that problems which correspond to rows in Figures 7.3(a) and (b) used $\delta$ in Equation A.9 equal to 3.0

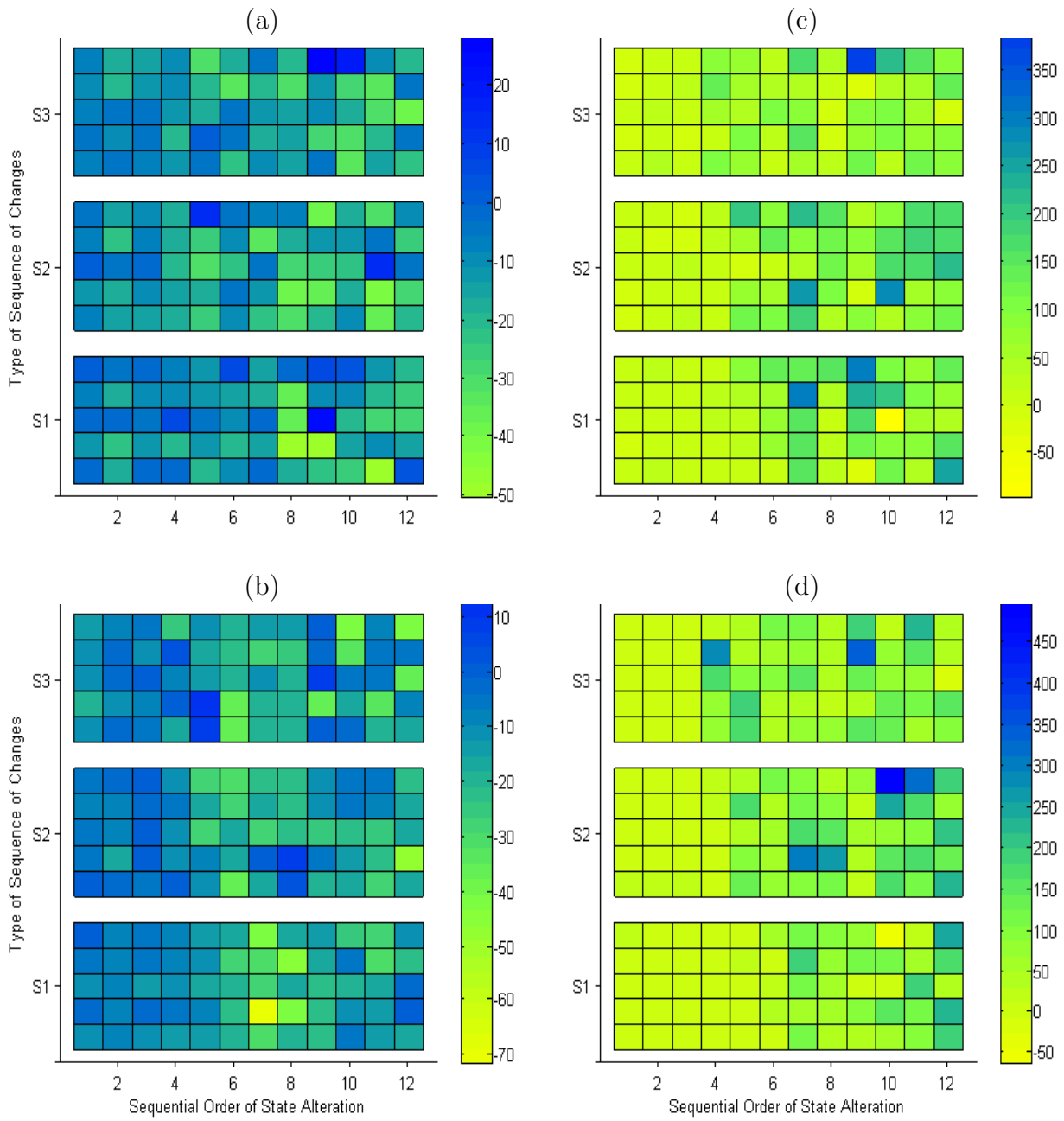



Figure 7.3: Dynamics of average difference in (a) makespan with $\delta = 6.0$ (b) makespan with $\delta = 3.0$ (c) cost with $\delta = 6.0$ (d) cost with $\delta = 3.0$

and 6.0 respectively. Thus, regardless of the $\delta$s, TSCs and TNISs used to characterise the sub-problems $x_e^2$ and $x_e^3$, the average $\mathbf{E}_{d,s}[\bullet]$ makespans of schedules obtained through McBAR is generally shorter than those of schedules obtained through McBAR-P.

Let us intuitively explain the recent results. In the schedule depicted in Figure 7.4(a), each horizontal strip represents a task; the subscript of the strip label $T_j$ is the task ID; the horizontal location of the strip's left edge signifies starting time of the task; and the

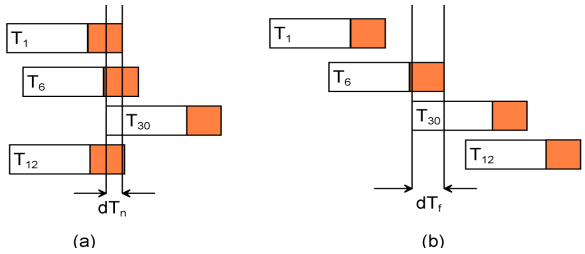



Figure 7.4: Schedule with tasks' starting time (a) near (b)

length of the white portion of the strip represents task duration prior to a change in a MOE, while the length of the other portion represents the amount of increase in the task duration during the change in this MOE. Further, tasks represented by all the strips utilise similar resource types. Now, suppose the starting times of the tasks are close to each other. Based on the figure, when duration of the tasks change, more of these tasks could overlap in time (around $dT_n$ in the figure). With this overlap, tasks simultaneously utilise resources of similar type, a simultaneity that could violate the constraint on these resources. One way to avoid this violation is to set the starting time of the tasks far from each other, such as those depicted in Figure 7.4(b). This figure depicts that when durations of the tasks change, few of these tasks could overlap in time (around $dT_f$ in the figure). Considering that the McBAR-P approach anticipates avoiding resource constraint violation, the schedule it determines could be akin to that in Figure 7.4(b) where task starting times are far from each other. On the other hand, a schedule determined through McBAR, which is determined without the anticipation intended to avoid the violation, need not separate task starting times far from one another, such as those in Figure 7.4(a). The schedule in Figure 7.4(b) has a longer makespan than the schedule in Figure 7.4(a). Thus, schedules determined through McBAR-P can be expected to have longer makespans than those determined through McBAR.

## 7.6 Difference in Cost

As discussed in Section 7.2.2, $dC_s(x, e)$ is the difference between the average implementation costs of non-dominated solutions/schedules determined by McBAR and the average

implementation costs of non-dominated solutions/schedules determined by McBAR-P in solving $x_e^2$ and $x_e^3$ sub-problems respectively. The $F_{6.0}$ formatted Figure 7.3(c) is a plot of $dC_s(x, e)$ at all pairs of problems $x$ and SOSAs $e$ where the problems label are from Table 4.5 and the problems have task duration variations that follow Equation A.9 with $\delta = 6.0$; and $0 \leq e \leq 12$. It shows that at 88.8% of the pairs $dC_s(x, e)$ is positive. Based on Equation 7.6, at 88.8% of the pairs, the average $\mathbf{E}_{d,s}[\bullet]$ implementation cost of schedules obtained through McBAR is more costly than that of schedules obtained through McBAR-P. The difference $dC_s(x, e)$ is more positive after the fifth SOSA of the MOEs than otherwise, regardless of problems in the pairs. These MOEs are where the problems are set.

After the fifth SOSA, based on Table 4.3 in conjunction with Table 4.2, severe types of changes that occur in the above MOE are prevalent. The $F_{3.0}$ formatted Figure 7.3(d) shows an approximately similar $dC_s(x, e)$ plot to that of Figure 7.3(c). It shows that at 90% of its corresponding problem-SOSA pairs $dC_s(x, e)$ is positive. Note that problems that correspond to rows in Figures 7.3(c) and (d) used $\delta$ in Equation A.9 that are equal to 3.0 and 6.0 respectively. Thus, regardless of the $\delta$s, TSCs and TNISs used to characterise the sub-problems $x_e^2$ and $x_e^3$, the average $\mathbf{E}_{d,s}[\bullet]$ implementation costs of schedules obtained through McBAR is more costly than that of schedules obtained through McBAR-P.

## 7.7 Conclusions and Future Work

The research presented in this chapter showed that McBAR-P determines some schedules with enough robustness whereby these schedules can be retained, without violating any constraint and still with high quality, at some changes in the MOE in which these schedules are implemented. If there is a high penalty for revising schedules, the manifested ability of McBAR-P is financially relevant.

The problems from $\mathcal{L}^2$ and $\mathcal{L}^3$ with the same label (e.g. $5^2$ and $5^3$ respectively) listed in Table 4.5 differ only in their number of objectives and being solved by McBAR and

McBAR-P, respectively, in this chapter. In addition to the last-considered characteristic, McBAR-P creates schedules with less implementation cost, but with longer durations, than those created by McBAR for solving any sub-problem of any problem with the same label found in Table 4.5.

As in the last chapter, the conclusions presented are true at any of the two values of $\delta$ used as the parameter of Equation A.9 which models changes in task durations of the problems with labels listed in Table 4.5; true at any type of simultaneous changes in the MOEs in which the problems are set; and true at any form of PNT utilised in this chapter. Based on the conclusions presented in this section, the second goal of the thesis is accomplished.

There are two suggested future directions to follow up the research undertaken in this chapter: the investigation of the performance of McBAR at other values of the above-considered $\delta$, and at forms of PNT other than those considered in this chapter.

# Response Surface Methodology (RSM)

## Chapter 8

The investigations undertaken in Chapters 6 and 7 focused on the problems from the sets $\mathcal{L}^2$ and $\mathcal{L}^3$ sets that have few elements. The conclusions on the characteristics of some techniques from $\mathcal{T}$ drawn from these investigations are only valid for these few elements. To obtain widely valid conclusions ( to be proven in Section 8.3.7) on the characteristics, the general method is utilised and explored in this chapter, which is organised as follows. Section 8.1 explains a method that utilises the modified *Method of Steepest Ascent* (MSA) (explained in Section 2.8.6). This method is applied in Section 8.2 to determine the performance-enhancing EA parametric values of EA-based techniques from $\mathcal{T}$. Some performance-enhancing parameters of EDA/$\mathcal{P}^2$ are also determined and explained in Section 8.2. The techniques are then endowed with their respective performance-enhancing parametric values. Then, they are applied by the general method to solve problems from $\mathcal{Q}^2 \subset \mathcal{M}$ where the generated solutions are used to build composite models, a building process explained in Section 8.3. These models are accurate to predict characteristics of the techniques at more problems than in each of the sets $\mathcal{L}^2$ and $\mathcal{L}^3$. The results of this model building are elaborated in Section 8.4. Making use of the models, the general method legitimises some algorithmic components of McBAR in Section 8.5. Further, it investigates in Section 8.6 the dynamic performance of the some techniques from $\mathcal{T}$. Thus, the first and fourth goals of the thesis are achieved in this chapter through the general method. Note that these same goals were also achieved in Chapter 6 through the limited method. A conclusion on the results given in this chapter is presented in Section 8.7.

## 8.1 Technique Performance Optimisation

This section provides information on the method referred to as *Technique Performance Optimisation* (TPO). This method is applied to determine the EA parametric values of each technique from $\mathcal{E}$ = {RI, NDLPOP, GIBAR, CBAM, McBA, McBAR, McBAS, MedianBAR} whereby this technique performs best for solving the extreme problem instance $X^2$ (described in Section 4.3.3). These EA parametric values are referred to as optimising parameters. Further, any performance referred to in this section is measured in hypervolume.

TPO utilises *Method of Steepest Ascent* (MSA) (described in Section 2.8.6) which involves RSM models. The *first* step in TPO is to decide which EA parameters of the techniques from $\mathcal{E}$ will be included as factors in the RSM models. As explained in Section 5.9, after a number of trials, the performances of these techniques were observed to stabilise before the $300^{th}$ generation of their evolutionary processes. Further, the performances were found to be best with the evolutionary processes each having a selection rate of 0.5. These two parameters are excluded as factors in the RSM models; however, their values are utilised by the techniques.

In [62], various selection, crossover, and mutation methods were considered in the efficiency optimisation of a GA method. In this thesis, however, only the crossover and the mutation methods explained in Section 5.1.3 are considered for the performance optimisation of the techniques from $\mathcal{E}$. The factors included in the RSM models are: population size $P$, crossover rate $\zeta$ and mutation rate $\mu$. The values that can be taken by these factors are restricted to,

$$\begin{aligned} 40 &\leq P \leq 112, \\ 0.1 &\leq \zeta \leq 1, \\ 0.03 &\leq \mu \leq 1. \end{aligned} \tag{8.1}$$

The population size takes values from the set,

$$P_{sz} = \{40, 52, 64, 76, 88, 100, 112\} \equiv \{P_i\}\,. \tag{8.2}$$

Suppose that the parameters $P$, $\zeta$ and $\mu$ change by a similar percentage of their respective ranges in Expression 8.1. After a number of trials, the performance of any technique from $\mathcal{E}$ was observed to change more when its population size and mutation rate independently vary than when its crossover rate varies by the same percentage. Thus, in the context of modified MSA (described in Section 2.8.6), the population size and mutation rate are regarded as important factors [31].

Using the terminologies in Sections 2.5.1, 4.3.3 and 4.4, let $H(\tau)$ denote a hypervolume of the NDS of solutions to an extreme sub-problem $X_i^2$ set in a simulation of the $i^{th}$ snapshot of a MOE, where the solutions are determined by a technique $\tau \in \mathcal{E}$; and $0 \leq i \leq 6$. The average $\mathbf{E}_s\left[H(\tau)\right]$ of the hypervolume $H(\tau)$ is taken over 30 simulations of the $i^{th}$ snapshot. Thus, there is an average hypervolume $\mathbf{E}_s\left[H(\tau)\right]$ that corresponds to each extreme sub-problems $X_3^2$ to $X_6^2$. The average of $\mathbf{E}_s\left[H(\tau)\right]$ over these extreme sub-problems is denoted by $\mathbf{E}_s^{3:6}\left[H(\tau)\right]$. A technique $\tau$ using the above-defined optimising parameters is expected to yield a high performance measured in terms of $\mathbf{E}_s^{3:6}\left[H(\tau)\right]$. The averaged hypervolume $\mathbf{E}_s^{3:6}\left[H(\tau)\right]$ could vary with different sets of values of the population size, crossover rate and mutation rate used by the technique $\tau$. Thus, it is also expressed as $\mathbf{E}_s^{3:6}\left[H(\tau, P, \zeta, \mu)\right]$.

Using the modified MSA described in Section 2.8.6 as a guide, the *second* step in TPO is to determine the Greek letter coefficients in the model,

$$\mathbf{E}_s^{3:6}\left[H(\tau, P_i, \zeta, \mu)\right] = \hat{\alpha}_{i,\tau} + \hat{\beta}_{i,\tau}\zeta + \hat{\gamma}_{i,\tau}\mu, \tag{8.3}$$

where $P_i \in P_{sz}$ is defined in Equation 8.2. To search for the coefficients, *Full Factorial Design* (FFD) (described in Section 2.8.1) is used to form the treatment,

$$\mathbf{T}_{i,\tau}^{j,k} = \mathbf{B}\mathbf{p}_i + j\mathbf{c} + k\mathbf{m} \equiv \langle P_i, \zeta_{j,\tau}, \mu_{k,\tau} \rangle\,, \tag{8.4}$$

where any vector considered in this section is an ordered set of population size, crossover rate and mutation rate, e.g. $\mathbf{x} = \langle P, \zeta, \mu \rangle$; integers $j$ and $k$ take values from the set

$\{-1, 0, 1\}$; $\mathbf{c} = \langle 0, 0.2, 0 \rangle$; $\mathbf{m} = \langle 0, 0, 0.1 \rangle$; and

$$\mathbf{Bp}_i = \langle P_i, 0.7, 0.2 \rangle . \tag{8.5}$$

Consider a plane in the space of crossover and mutation rates. Let this plane correspond to a given population size $P_i$ and to technique $\tau$. When the treatments $\mathbf{T}_{i,\tau}^{j,k}$, that correspond to various values of $j$ and $k$, are projected onto the plane, they become the corners, edge midpoints, and centre ($\mathbf{Bp}_i$) of a rectangle in this plane. The base point $\mathbf{Bp}_i$ in Equation 8.5 was determined from trial investigations to yield a large value of the averaged hypervolume $\mathbf{E}_s^{3:6}\left[H(\tau, P_i, \zeta_{0,\tau}, \mu_{0,\tau})\right]$ of non-dominated solutions determined by the technique $\tau$ that uses a population size $P_i$. Note that $j$ and $k$ are set to zero in $\zeta_{j,\tau}$ and $\mu_{k,\tau}$ for $\mathbf{E}_s^{3:6}\left[H(\tau, P_i, \zeta_{0,\tau}, \mu_{0,\tau})\right]$.

The extreme sub-problem $X_i^2$ is solved by the technique $\tau$ using the values in the treatment $\langle P_i, \zeta_{j,\tau}, \mu_{k,\tau} \rangle$ defined in Equation 8.4. Next, the average hypervolume $\mathbf{E}_s^{3:6}\left[H(\tau, P_i, \zeta_{j,\tau}, \mu_{k,\tau})\right]$ of the obtained non-dominated solutions are determined. This procedure is repeated for each $P_i \in P_{sz}$, $j, k \in \{-1, 0, 1\}$ and $\tau \in \mathcal{E}$. The various values of $\mathbf{E}_s^{3:6}\left[H(\tau, P_i, \zeta_{j,\tau}, \mu_{k,\tau})\right]$ obtained from the repetitions are used to determine the coefficients in Equation 8.3 following the linear model building method explained in Section 2.8.4.

The *third* step in TPO is to determine the optimising parameters. MSA (described in Section 2.8.6) is applied to form the treatments,

$$\mathbf{T}_{i,\tau}^{l} \equiv \langle P_i, \zeta_{l,\tau}, \mu_{l,\tau} \rangle = \mathbf{Bp}_i + l\mathbf{a}_{i,\tau}, \tag{8.6}$$

where,

$$0 \leq l \leq 0.7; \qquad l \text{ is a multiple of } 0.1; \tag{8.7}$$

$$\mathbf{a}_{i,\tau} = \left\langle 0, \hat{\beta}_{i,\tau}/\hat{\gamma}_{i,\tau}, 1 \right\rangle ; \tag{8.8}$$

and $\hat{\beta}_{i,\tau}$ and $\hat{\gamma}_{i,\tau}$ are the coefficients in Equation 8.3 that are already determined at this stage of the TOP. Based on Equations 8.5 to 8.8, the crossover rate $\zeta_{l,\tau}$ is related to the mutation rate $\mu_{l,\tau}$ as,

$$\zeta_{l,\tau} = (\mu_{l,\tau} - 0.2)\,\frac{\hat{\beta}_{i,\tau}}{\hat{\gamma}_{i,\tau}} + 0.7, \tag{8.9}$$

and

$$\mu_{l,\tau} = 0.2 + l. \tag{8.10}$$

Notice that the crossover rate obtained from this equation could be outside the range of zero to one. The treatment obtained from Equation 8.6 that has a crossover rate outside the range is called an *invalid treatment*; otherwise, it is called a *valid treatment*.

The extreme sub-problem $X_i^2$ is again solved by the technique $\tau$, this time using the values in the treatment $\langle P_i, \zeta_{l,\tau}, \mu_{l,\tau}\rangle$ in Equation 8.6. The average hypervolume $\mathbf{E}_s^{3:6}\left[H(\tau, P_i, \zeta_{l,\tau}, \mu_{l,\tau})\right]$ of the obtained non-dominated solutions is determined. This procedure is repeated for every $l$, $P_i$ and $\tau$. The values of $l$ and $i$ are determined at which, for a given technique $\tau$, the averaged hypervolume $\mathbf{E}_s^{3:6}\left[H(\tau, P_i, \zeta_{l,\tau}, \mu_{l,\tau})\right]$ is maximum,

$$\langle l_{max}^{\tau}, i_{max}^{\tau}\rangle = \max_{l,i} \mathbf{E}_s^{3:6}\left[H(\tau, P_i, \zeta_{l,\tau}, \mu_{l,\tau})\right]. \tag{8.11}$$

Finally, using Equations 8.5 to 8.8, the optimising population size $P_{max}^{\tau}$, crossover rate $\zeta_{max}^{\tau}$ and mutation rate $\mu_{max}^{\tau}$ of the technique $\tau$ are determined from,

$$\langle P_{max}^{\tau}, \zeta_{max}^{\tau}, \mu_{max}^{\tau}\rangle = \left\langle P_{i_{max}^{\tau}}, 0.7 + l_{max}^{\tau}\hat{\beta}_{i_{max}^{\tau},\tau}/\hat{\gamma}_{i_{max}^{\tau},\tau}, 0.2 + l_{max}^{\tau}\right\rangle. \tag{8.12}$$

From the above discussions, the optimising parameters of the technique $\tau$ are determined by applying $\tau$ to solve the extreme problem $X^2$. If the performance of technique $\tau$, endowed with its optimising parameters, is optimal for solving the extreme problem set in a MOE that undergoes an extreme change then, based on the definitions in Section 4.3.3, it is legitimate to assume that $\tau$ would still perform best for solving the slight problem set in a MOE that undergoes a slight change. Thus, the optimising parameters in Equation

8.12 are used by the technique $\tau \in \mathcal{E}$ for solving any problem (extreme or otherwise) from $\mathcal{O}^2 \subset \mathcal{Q}^2 \subset \mathcal{M}$ after they are determined.

## 8.2 Optimising Parameters

The TPO method explained recently will now be applied to obtain the performance-enhancing EA parametric values of techniques from $\mathcal{E}$. Through this method the coefficients of the models expressed in Equation 8.3 are determined. These coefficients are presented in Table 8.1. For each row in this table, the entries under the third to the fifth columns are the coefficients of a model which corresponds to the pair of technique and population size under the first two columns respectively; and the entry under the last column is the ratio $\hat{\beta}_{i,\tau}/\hat{\gamma}_{i,\tau}$ in Equation 8.8. Consider the row in the table that corresponds to the pair of technique RI and population size of 100. This row has entries which when substituted to Equation 8.3 yields the model,

$$\mathbf{E}_s^{3:6}\left[H(\mathrm{RI}, 100, \zeta, \mu)\right] = 952393 + 1650\zeta + 3579\mu. \tag{8.13}$$

Based on Equation 8.2, $i = 6$ for $P_i \in P_{sz}$ to be 100. Thus, based on the considered row and the last column, the ratio $\hat{\beta}_{i,\tau}/\hat{\gamma}_{i,\tau}$ that corresponds to this model is $0.461 = \hat{\beta}_{6,\mathrm{RI}}/\hat{\gamma}_{6,\mathrm{RI}}$.

The last result is used to explain the next step in TPO. The model in Equation 8.13 is utilised in Equations 8.5 to 8.9 to create the set $\left\{\mathbf{T}_{6,\mathrm{RI}}^{l}\right\}$ of treatments,

$$\begin{aligned}
\mathbf{T}_{6,\mathrm{RI}}^{0} &= \langle 100, 0.7, 0.2\rangle\,; & \mathbf{T}_{6,\mathrm{RI}}^{0.1} &= \langle 100, 0.7461, 0.3\rangle\,; \\
\mathbf{T}_{6,\mathrm{RI}}^{0.2} &= \langle 100, 0.7922, 0.4\rangle\,; & \mathbf{T}_{6,\mathrm{RI}}^{0.3} &= \langle 100, 0.8383, 0.5\rangle\,; \\
\mathbf{T}_{6,\mathrm{RI}}^{0.4} &= \langle 100, 0.8844, 0.6\rangle\,; & \mathbf{T}_{6,\mathrm{RI}}^{0.5} &= \langle 100, 0.9305, 0.7\rangle\,; \\
\mathbf{T}_{6,\mathrm{RI}}^{0.6} &= \langle 100, 0.9766, 0.8\rangle\,; & \mathbf{T}_{6,\mathrm{RI}}^{0.7} &= \langle 100, 1.0227, 0.9\rangle\,.
\end{aligned} \tag{8.14}$$

The superscripts of these treatments are the values of $l$ in expression 8.7. The subscript

Table 8.1: Coefficients of models determined through TPO

| $\tau$ | $P_i$ | $\hat{\alpha}_{i,\tau}$ | $\hat{\beta}_{i,\tau}$ | $\hat{\gamma}_{i,\tau}$ | $\hat{\beta}_{i,\tau}/\hat{\gamma}_{i,\tau}$ |
|---|---|---|---|---|---|
| CBAM | 40 | 948,362 | 458.96 | 10224.67 | 0.045 |
| | 52 | 949,172 | 1154.23 | 7721.87 | 0.149 |
| | 64 | 955,435 | -10.79 | -155.45 | 0.069 |
| | 76 | 952,949 | -225.59 | 6621.90 | -0.034 |
| | 88 | 956,450 | 3.45 | 40.63 | 0.085 |
| | 100 | 950,604 | 965.26 | 7645.44 | 0.126 |
| | 112 | 959,491 | -1710.33 | -4387.08 | 0.390 |
| GIBAR | 40 | 946,420 | 348.01 | 1214.73 | 0.286 |
| | 52 | 949,804 | 142.21 | 731.07 | 0.195 |
| | 64 | 948,191 | 204.98 | 1108.31 | 0.185 |
| | 76 | 948,621 | 166.08 | 754.63 | 0.220 |
| | 88 | 945,025 | 364.77 | 1059.74 | 0.344 |
| | 100 | 941,371 | -238.39 | -1748.94 | 0.136 |
| | 112 | 952,866 | -275.25 | -793.73 | 0.347 |
| RI | 40 | 948,913 | 31.55 | 10994.86 | 0.003 |
| | 52 | 952,887 | -719.79 | 6428.90 | -0.112 |
| | 64 | 954,912 | 84.75 | 1242.14 | 0.068 |
| | 76 | 953,783 | 289.33 | 851.29 | 0.340 |
| | 88 | 950,631 | -540.11 | 9071.17 | -0.060 |
| | 100 | 952,393 | 1650.13 | 3579.05 | 0.461 |
| | 112 | 951,268 | -4833.48 | 9400.17 | -0.514 |
| NDLPOP | 40 | 943,123 | 282.16 | 2084.07 | 0.135 |
| | 52 | 952,414 | -800.61 | -2969.69 | 0.270 |
| | 64 | 948,823 | 2092.03 | -2474.00 | -0.846 |
| | 76 | 937,414 | 477.11 | 3882.09 | 0.123 |
| | 88 | 944,527 | 391.48 | 7170.53 | 0.055 |
| | 100 | 939,307 | 1145.28 | 2463.04 | 0.465 |
| | 112 | 939,367 | -4355.96 | 6245.47 | -0.697 |
| McBA | 40 | 983,507 | -1489.10 | 6750.11 | -0.221 |
| | 52 | 983,179 | -582.22 | 7999.91 | -0.073 |
| | 64 | 986,091 | -11.29 | -177.01 | 0.064 |
| | 76 | 983,886 | -954.70 | 5170.02 | -0.185 |
| | 88 | 981,627 | 296.36 | 7692.67 | 0.039 |
| | 100 | 981,225 | 450.46 | 9064.52 | 0.050 |
| | 112 | 976,364 | 1281.64 | 14185.01 | 0.090 |
| McBAR | 40 | 988,216 | 525.93 | 3442.37 | 0.153 |
| | 52 | 988,370 | 664.80 | 5770.80 | 0.115 |
| | 64 | 990,826 | 216.13 | 806.72 | 0.268 |
| | 76 | 990,699 | 403.96 | 3469.25 | 0.116 |
| | 88 | 991,232 | 2006.17 | 4707.68 | 0.426 |
| | 100 | 988,875 | 1761.69 | 4508.63 | 0.391 |
| | 112 | 985,221 | 3620.10 | 7907.97 | 0.458 |
| MedianBAR | 40 | 982,760 | 106.64 | 12428.00 | 0.009 |
| | 52 | 990,287 | -232.62 | 4066.40 | -0.057 |
| | 64 | 981,362 | 4769.27 | 10076.62 | 0.473 |
| | 76 | 995,489 | -925.60 | -2882.70 | 0.321 |
| | 88 | 989,760 | 336.03 | 4963.11 | 0.068 |
| | 100 | 991,014 | 754.78 | 3912.23 | 0.193 |
| | 112 | 986,468 | 3345.16 | 7122.86 | 0.470 |
| McBAS | 40 | 978,547 | 2833.02 | 6648.57 | 0.426 |
| | 52 | 977,807 | 347.52 | 11153.03 | 0.031 |
| | 64 | 981,529 | 111.43 | 5157.92 | 0.022 |
| | 76 | 979,056 | 137.44 | 11660.12 | 0.012 |
| | 88 | 981,014 | -309.82 | 8006.33 | -0.039 |
| | 100 | 978,264 | 739.29 | 13037.04 | 0.057 |
| | 112 | 985,351 | -506.60 | 1850.71 | -0.274 |

6 of the treatments is the index in $P_6 = 100$. Notice that the last treatment $\mathbf{T}^{0.7}_{6,\mathrm{RI}}$ expresses that RI is supposed to use a population size of 100, a crossover rate of 1.0227 and a mutation rate of 0.9. Thus, as defined in Section 8.1, this treatment is an invalid one. Consequently, RI that has these parametric settings was not applied to solve the extreme problem $X^2$, an application intended to determine the average hypervolume $\mathbf{E}^{3:6}_s\left[H(\mathrm{RI}, 100, 1.0227, .9)\right]$ (notation based on Equation 8.13).

The set of treatments in Equation 8.14 corresponds to only one model (Equation 8.13) whose coefficients are found in the row of Table 8.1 that was noted above. For each row in this table, a set of treatments is determined by the same process as in the last paragraph. The components (e.g. in $\langle 100, 0.9305, 0.7\rangle$) of each valid treatment (defined in Section 8.1, e.g. $\mathbf{T}^{0.5}_{6,\mathrm{RI}}$ in Equation 8.14) in this set are used by the technique (e.g. RI) that corresponds to this set. This technique is then applied to solve the extreme problem $X^2$. The solutions obtained are used to compute the average hypervolume $\mathbf{E}^{3:6}_s\left[H(\tau, P_i, \zeta_{l,\tau}, \mu_{l,\tau})\right]$ that corresponds to a valid treatment $\mathbf{T}^{l}_{i,\tau}$ in the set. Note that the elements of this set correspond to $l$ defined in expression 8.7.

Each of the Figures 8.2 to 8.8 plots the average hypervolume $\mathbf{E}^{3:6}_s\left[H(\tau, P_i, \zeta_{l,\tau}, \mu_{l,\tau})\right]$ at each value of population size $P_i \in P_{sz}$ and parameter $l$ in expression 8.7, where its vertical and horizontal axes are the population size and mutation rate respectively. These figures correspond to RI, CBAM, McBA, McBAR, MedianBAR, McBAS, NDLPOP and GIBAR respectively. For example, Figure 8.1 corresponds to RI. Now, the row labelled 100 in this figure corresponds to the set $\left\{\mathbf{T}^{l}_{6,\mathrm{RI}}\right\}$ of treatments expressed in Equation 8.14 and its furthest left to furthest right rectangles correspond to parameter $l$ which has values of zero to 0.7, respectively, with 0.1 intervals. By using the values of $l$ for Equation 8.10, the rectangles in the row also correspond to mutation rates of 0.2 to 0.9, respectively, with 0.1 increments. By using these mutation rates for Equation 8.9 with $\hat{\beta}_{6,\mathrm{RI}}/\hat{\gamma}_{6,\mathrm{RI}} = 0.461$ (this equality is explained above), the rectangles in the row labelled 100 correspond to crossover rates of 0.7, 0.7461, 0.7922, 0.8383, 0.8844, 0.9305, 0.9766 and 1.0227 respectively. Thus, the first rectangle (corresponds to $l = 0$) from the left of row labelled 100 in Figure 8.1 is

the colour hue-represented average hypervolume $\mathbf{E}_s^{3:6}\left[H(\mathrm{RI}, 100, \zeta_{6,\mathrm{RI}}, \mu_{6,\mathrm{RI}})\right]$ determined through RI that uses a population size of 100, a crossover rate of 0.7 and a mutation rate of 0.2 for solving the extreme problem $X^2$.

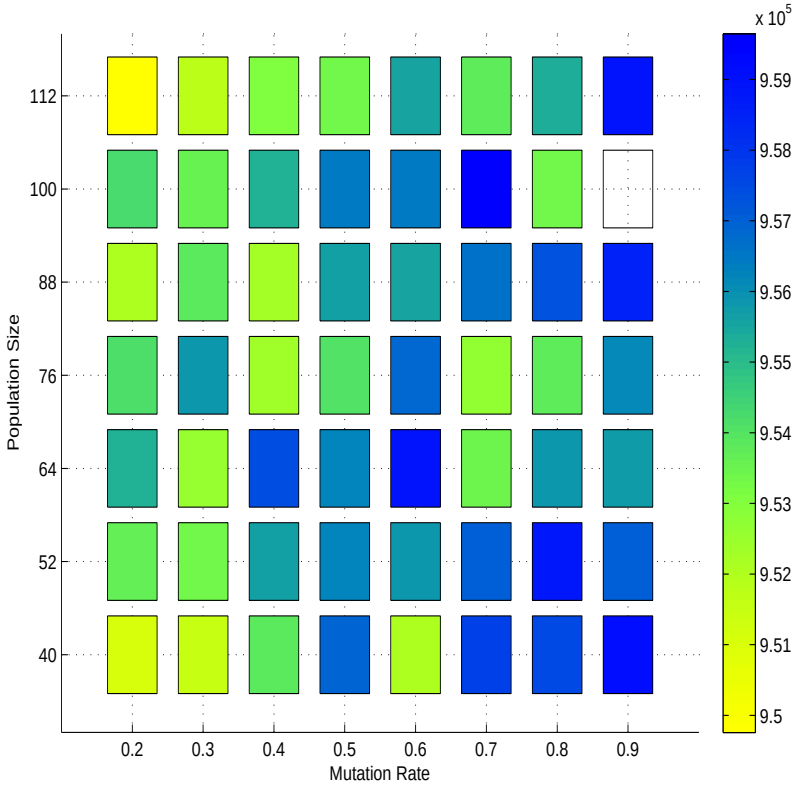



Figure 8.1: Observed hypervolume of solutions determined by RI under TPO

Note that the colour representation in Figures 8.2 to 8.8 depends on the scale expressed by the colour bar to the right and beside each of these figures. Thus, similarly coloured rectangles in different figures could represent different average hypervolumes. As declared above, the average hypervolume that corresponds to the invalid treatment $\mathbf{T}_{6,\mathrm{RI}}^{0.7}$ was not determined. All undetermined average hypervolumes are indicated in all of the figures as unfilled rectangles. This scheme explains the presence of the unfilled rectangle at population size 100 and mutation rate 0.9, a coordinate in Figure 8.1 that corresponds to the invalid treatment $\mathbf{T}_{6,\mathrm{RI}}^{0.7}$ considered above.

Let us now characterise Figures 8.2 to 8.8. Generally, in each row of each of these figures, the average hypervolume first increases with mutation rate then decreases at some point. Consider for example row 88 of Figure 8.3 which corresponds to McBA that

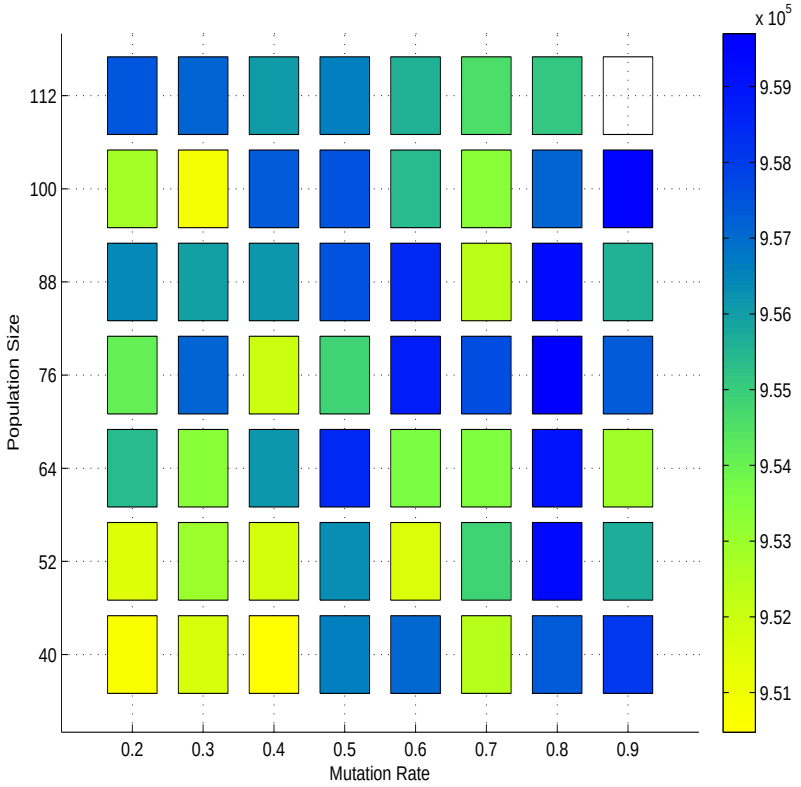



Figure 8.2: Observed hypervolume of solutions determined by CBAM under TPO

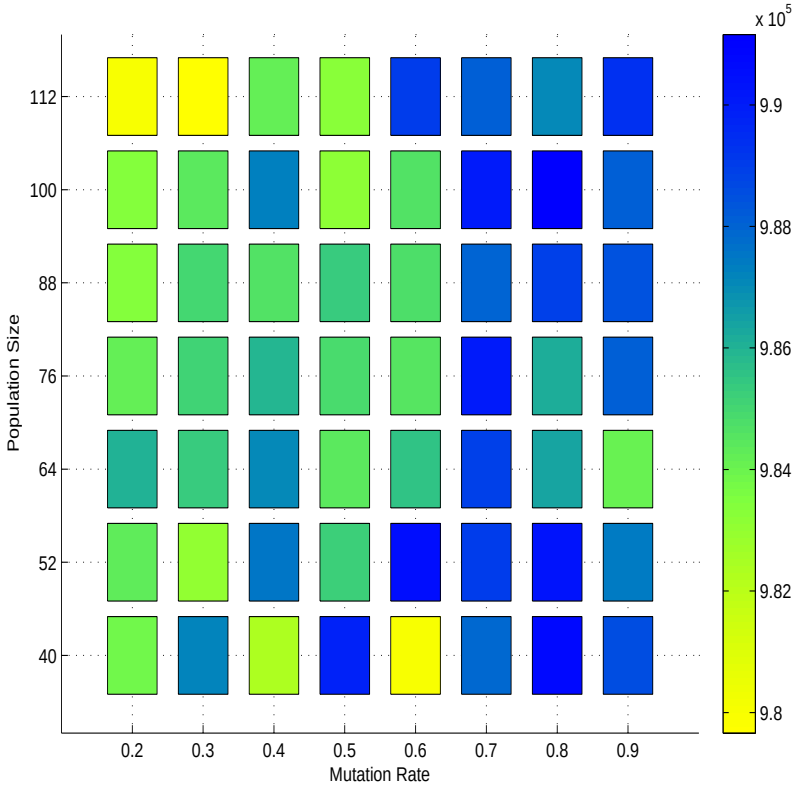



Figure 8.3: Observed hypervolume of solutions determined by McBA under TPO

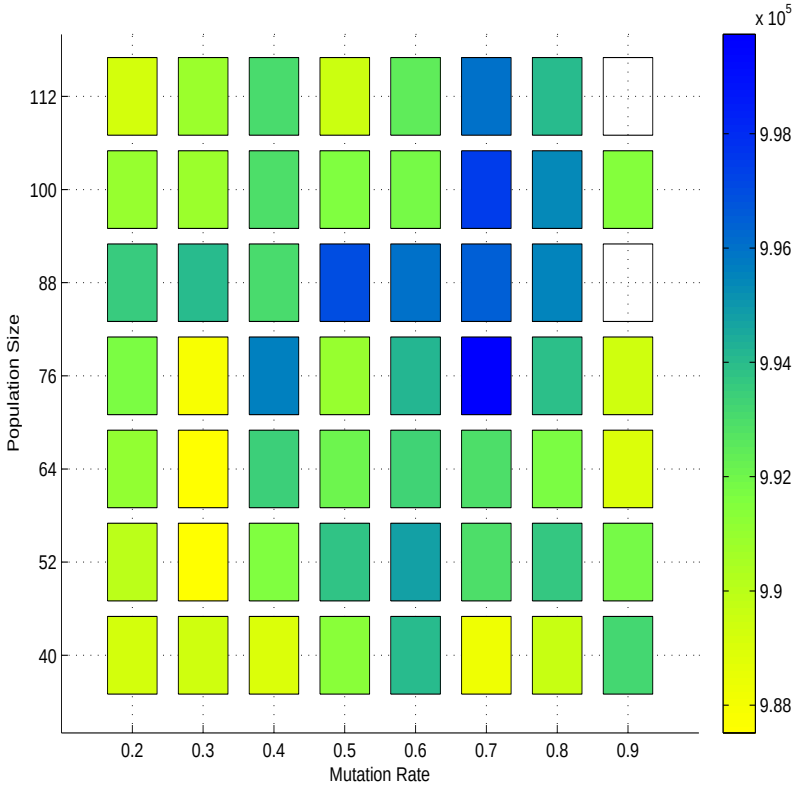



Figure 8.4: Observed hypervolume of solutions determined by McBAR under TPO

in this case uses a population size of 88. In this row, the average hypervolume is higher when the mutation rate used by McBA is within 0.7 and 0.9 inclusive than otherwise. Based on the definitions of hypervolume and average hypervolume $\mathbf{E}_s^{3:6}\left[H(\tau, P_i, \zeta_{l,\tau}, \mu_{l,\tau})\right]$ in Sections 2.5.1 and 8.1, respectively, the quality of solutions can be measured in terms of the average hypervolume, where these solutions are obtained by the technique $\tau$ from $\mathcal{E}$ for solving the extreme problem $X^2$. Thus, row 88 of Figure 8.3 demonstrates that the quality of solutions obtained by McBA for solving $X^2$ is higher when McBA uses a mutation rate of 0.7, 0.8 or 0.9. Note that McBA also uses a population size of 88 and the crossover rate that can be obtained through the process explained above. The average hypervolume dynamics (pattern of changes and not values) across row 88 is basically similar to that of the other rows in Figures 8.2 to 8.8. Therefore, the quality of solutions obtained by any technique from $\mathcal{E}$ for solving $X^2$ is generally higher when this technique uses a mutation rate of 0.7, 0.8 or 0.9.

Consider Figure 8.2 which corresponds to CBAM. The maximum average hypervolume

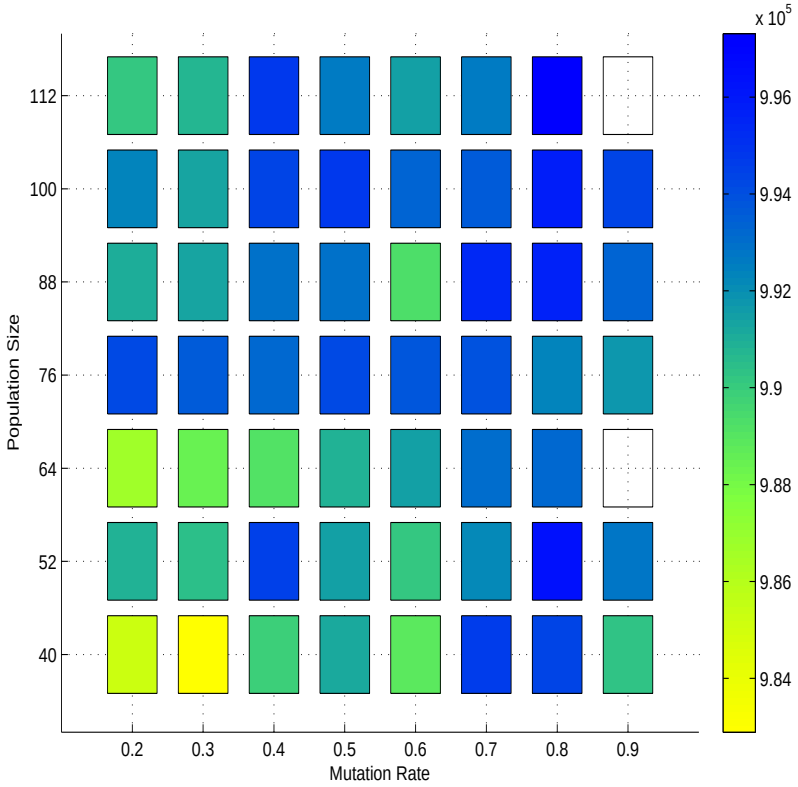



Figure 8.5: Observed hypervolume of solutions determined by MedianBar under TPO

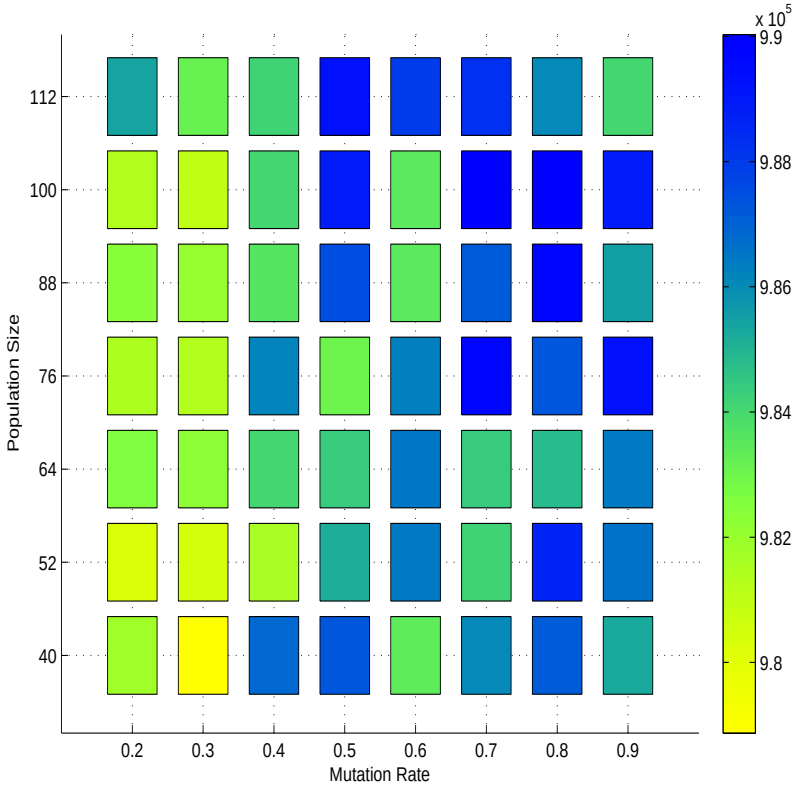



Figure 8.6: Observed hypervolume of solutions determined by McBAS under TPO

depicted in this figure is at a population size of 76 (equal to $P_4$ base on Equation 8.2) and a mutation rate of 0.8. The ratio $\hat{\beta}_{i,\tau}/\hat{\gamma}_{i,\tau} = \hat{\beta}_{4,\text{CBAM}}/\hat{\gamma}_{4,\text{CBAM}}$ = -0.034 is obtained by using the population size and the technique named CBAM for Table 8.1. Then, a crossover rate of 0.6786 is obtained by using the ratio and the mutation rate for Equation 8.9. The maximum average hypervolume implies that CBAM performs best for solving the extreme problem $X^2$ when using a population size of 76, a mutation rate of 0.8 and a crossover rate of 0.6786. Based on Sections 8.1 and 4.3.3, although this implication is valid only for solving $X^2$, it is legitimate to assume that CBAM would also perform optimally for solving any problem from $\mathcal{O}^2$.

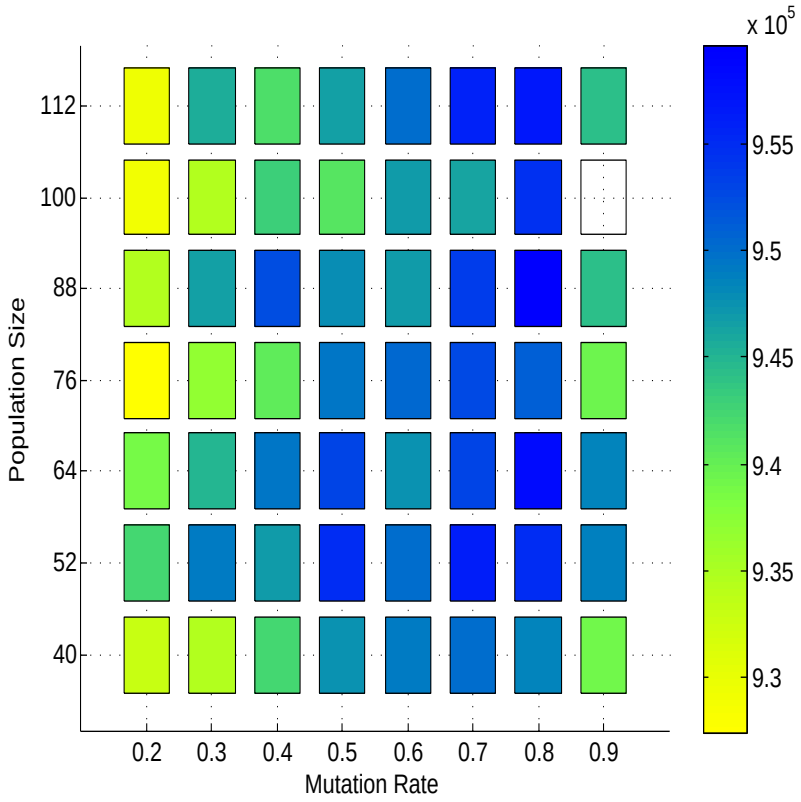



Figure 8.7: Observed hypervolume of solutions determined by NDLPOP under TPO

All the performance-optimising EA-parametric values of each technique in $\mathcal{E}$ are obtained through the process just explained and are presented in the upper part of Table 8.2. Based on this table portion and on Section 4.3.3, all techniques in $\mathcal{E}$ performed optimally for solving any problem from $\mathcal{O}^2$ when using a population size in $P_{sz}$ greater than 75, a crossover rate greater than 0.6 and a mutation rate greater than 0.6. Note that,

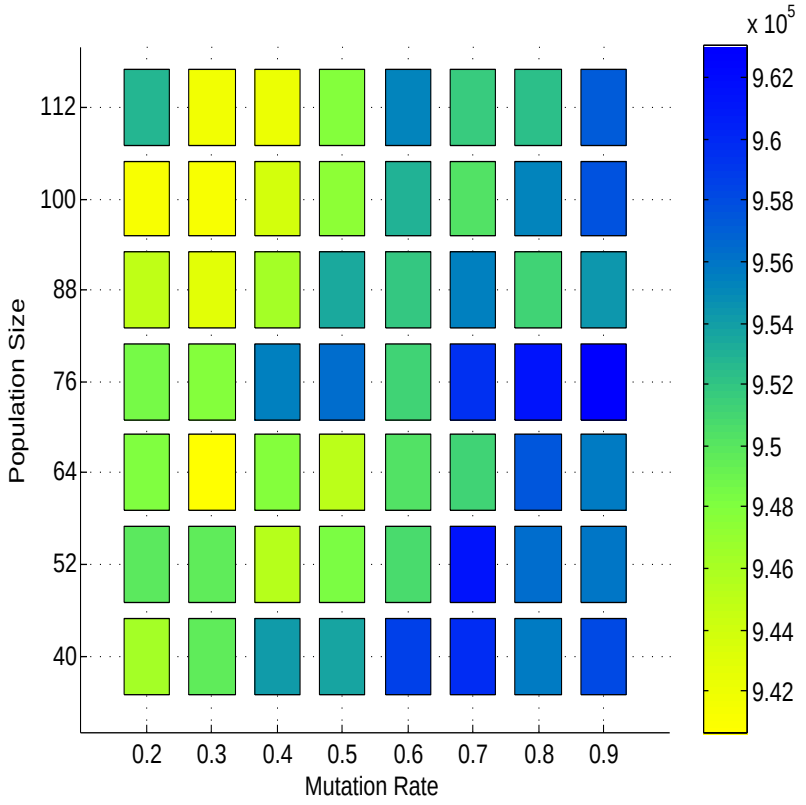



Figure 8.8: Observed hypervolume of solutions determined by GIBAR under TPO

based on Section 8.1, these techniques also use a selection rate of 0.5 and terminate their respective evolutionary processes after 300 generations. The last column of the upper part of Table 8.2 corresponds to the maximum average hypervolume in each of Figures 8.2 to 8.8. The techniques arranged in descending order of maximum average hypervolume are: McBAR, MedianBAR, McBA, McBAS, GIBAR, CBAM, NDLPOP and RI. Hence, the performance of McBAR is the best of all the techniques in $\mathcal{E}$ for solving the extreme problem $X^2$. Based on Sections 8.1 and 4.3.3, this conclusion translates as McBAR possibly having the best performance among all the techniques for solving any problem from $\mathcal{O}^2$. Using the definitions in Section 6.3, Table 8.2 illustrates that the optimal performances of the techniques (from $\mathcal{E}$) which apply the mapping function $\mathcal{F}$ (e.g. McBAR, MedianBAR, McBA and McBAS) are better than those of the techniques that do not (e.g. CBAM and RI). Further, the optimal performance of RI is the worst of all techniques in $\mathcal{E}$. Note that RI does not follow any rule, except SSGS, for creating any initial population.

The set $\mathcal{E}$ of techniques does not include EDA/$\mathcal{P}^2$. To present all the performance-

Table 8.2: Optimal parameters

| Technique | Population Size | Crossover Rate | Mutation Rate | Hypervolume |
|---|---|---|---|---|
| RI | 100 | 0.9295 | 0.7 | 959,650 |
| NDLPOP | 88 | 0.7328 | 0.8 | 959,678 |
| GIBAR | 76 | 0.8541 | 0.9 | 963,025 |
| CBAM | 76 | 0.6786 | 0.8 | 959,700 |
| McBA | 100 | 0.7295 | 0.8 | 991,160 |
| McBAR | 76 | 0.7558 | 0.7 | 999,740 |
| McBAS | 100 | 0.7282 | 0.7 | 990,030 |
| MedianBAR | 112 | 0.9775 | 0.8 | 997,310 |

| Technique | Learning Rate | Percentage Population | | Hypervolume |
|---|---|---|---|---|
| EDA/$\mathcal{P}^2$ | 0.8 | 0.8 | | 927,480 |

optimising parameters of techniques from $\mathcal{T}$, the process of determining the EDA/$\mathcal{P}^2$-performance boosting values of learning rate $\lambda$ and percentage $\rho$ (respectively defined in Items 7 and 5 of the last enumeration in Section 5.8) are presented here despite this process being not RSM-related. The process is as follows. The parameters $\lambda$ and $\rho$ are restricted to take values in the range 0.1 to 0.9 with a 0.1 interval. For each pair of permissible $\lambda$ and $\rho$ values, EDA/$\mathcal{P}^2$ solves the extreme problem $X^2$ (described in Section 4.3.3) from $\mathcal{O}^2$. Then the performance defined in terms of the average $\mathbf{E}_s^{3:6}\left[H(\text{EDA}/\mathcal{P}^2, \lambda, \rho)\right]$ (i.e. the hypervolume is a function of the technique EDA/$\mathcal{P}^2$, $\lambda$ and $\rho$) is determined from the solutions obtained by EDA/$\mathcal{P}^2$. The performances that correspond to all of the pairs are plotted in Figure 8.9. This figure illustrates that the performances are generally high at $\lambda$ greater than 0.6 and $\rho$ greater than 0.4. Among the pairs, EDA/$\mathcal{P}^2$ performs best at $\lambda$ and $\rho$ each being 0.8 which is presented in the lower part of Table 8.2. Note in this lower portion that EDA/$\mathcal{P}^2$ has the worst performance – measured in hypervolume – among all the techniques from $\mathcal{T}$.

## 8.3 Voxel Models

As emphasised in Chapter 1, the general method is applied to build empirical models through RSM. Each of these models describes the dynamics of the relative performances of two different techniques from $\mathcal{W} = \{\text{RI}, \text{CBAM}, \text{McBA}, \text{McBAR}, \text{McBAS}, \text{ MedianBAR}\}$ (defined in Table 5.1). These performances are for solving the problems from $\mathcal{O}^2 \subset \mathcal{Q}^2 \subset$

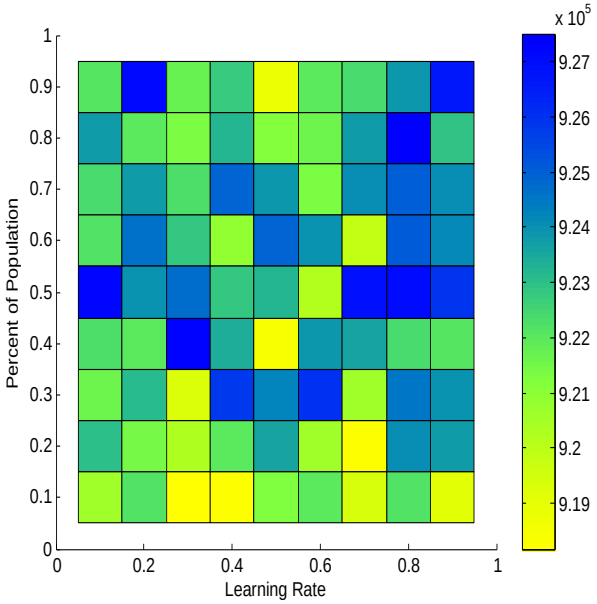



Figure 8.9: Performance of EDA/$\mathcal{P}^2$ at various values of its parameters

$\mathcal{M}$ and are measured in terms of the set coverage explained in Section 2.5.2. From here onwards, each of the models will be referred to as dynamical model. Let us now provide information on how the general method develops the dynamical models.

### 8.3.1 Factors

First, let us look into the factors that constitute the dynamical models. Consider a set of task duration changes sampled from the model in Equation A.9 with $\delta = 3.0$ and another set of task duration changes sampled from the same model with $\delta = 6.0$. The performances of techniques in $\mathcal{W}$ were demonstrated in Section 6.2 to be generally independent of the task duration changes in both sets, where these changes occur in a MOE in which a problem from $\mathcal{L}^2$ is set. Based on this result, all task duration changes in the MOE are screened-out (as defined in Section 2.8.3) as factors in the dynamical models.

The factors included in the dynamical models are the number of new tasks and the numbers of newly broken resources of types $R_1$ and $R_2$ at every SOSA $c$ in the *significant range of orders* (SRO). As noted in Section 4.3.1, SRO range from the third to the sixth

SOSA of any MOE. These factors are denoted as,

$$\begin{aligned} x_{3c-8} &\Rightarrow \text{number of new tasks,} \\ x_{3c-7} &\Rightarrow \text{number of newly broken resources of type } dR_1, \\ x_{3c-6} &\Rightarrow \text{number of newly broken resources of type } dR_2, \end{aligned} \tag{8.15}$$

where the symbol $\Rightarrow$ denote correspondence. Based on the SRO, there are 12 factors in each of the dynamical models; three factors correspond to each SOSA in the SRO. For example, factors $x_7$, $x_8$ and $x_9$ respectively denote the number of new tasks and the numbers of newly broken resources of types $R_1$ and $R_2$ at the fifth SOSA of a MOE.

There are various combinations of the values of the 12 factors. Using the terminology in Section 2.8, each of the combinations is considered as a treatment to a computing experiment that simulates a MOE whose changes are codified in the combination. The general expression of treatment for the dynamical models is,

$$\mathbf{x} = \langle x_1, x_2, \ldots, x_{12} \rangle \,. \tag{8.16}$$

A particular treatment could be,

$$\mathbf{x} = \langle 1, 2, 0, 3, 4, 1, 4, 3, 0, 0, 1, 4 \rangle \,. \tag{8.17}$$

This treatment expresses that at the third SOSA of the MOE there are: one new task, two newly broken resources of type $R_1$, and no newly broken resources of type $R_2$; at the fourth SOSA there are: three new tasks, four newly broken resources of type $R_1$, and one newly broken resource of type $R_2$; and so on.

As noted in Section 4.3.2, the definition of any problem from $\mathcal{O}^2$ is incomplete; at the third to the sixth SOSA of a MOE in which any problem from $\mathcal{O}^2$ is set, the number of new tasks, the number of newly broken resources of type $R_1$ and the number of newly broken resources of type $R_2$ are undefined. These numbers are all defined in the treatment expressed in Equation 8.16. Thus, the partial definition in Sections 4.3.1 and 4.3.2

of any problem from $\mathcal{O}^2$ including all the information encoded in the treatment, which corresponds to this problem, completes the definition of this problem.

### 8.3.2 Resolutions, Levels and Ranges

At each SOSA in the SRO, the number of new tasks and the numbers of newly broken resources of both $R_1$ and $R_2$ types are integers, such that, the resolutions (defined in Section 2.8) of their corresponding factors is unity. Factors that represent the number $x_{3c-8}$ of new tasks, the number $x_{3c-7}$ of newly broken resources of type $R_1$ and the number $x_{3c-6}$ of newly broken resources of type $R_2$ at the $c^{th}$ SOSA have ranges expressed in,

$$\begin{aligned} 0 \leq x_{3c-8} \leq 5, \\ 0 \leq x_{3c-7} \leq 4, \\ 0 \leq x_{3c-6} \leq 4, \end{aligned} \tag{8.18}$$

where $3 \leq c \leq 6$; and these factors have six, five and five levels respectively.

Based on expression 8.18, the number $x_{3c-8}$ of new tasks can be zero at each SOSA $c$ in the SRO, i.e. $3 \leq c \leq 6$. Further, based on Section 4.3.1, there is no new task at the first ($c = 1$) and at the second SOSA ($c = 2$) of a MOE in which any problem from $\mathcal{O}^2$ is set. Thus, it is possible that no new task can appear in the entire dynamics of the MOE, i.e. $\forall\ c$ in the range $0 \leq c \leq 6$. Let an *irrelevant problem* be a problem from $\mathcal{M}$ set in a MOE whereby no new task appears. Further, let an *irrelevant treatment* be a treatment that corresponds to an irrelevant problem. For example,

$$\mathbf{i} = \langle 0, 2, 1, 0, 4, 1, 0, 3, 2, 0, 1, 4 \rangle . \tag{8.19}$$

is an irrelevant treatment. Now, implicit in Section 5.9, the performance of McBAR for solving a problem from $\mathcal{M}$ is expected to be similar to that of CBAM when this problem is set in a dynamic MOE with no new task. Thus, to possibly differentiate the performances of McBAR and CBAM, the problem from $\mathcal{O}^2 \subset \mathcal{Q}^2 \subset \mathcal{M}$ that has at least one new task

is the type of problem considered in the thesis. Let the set $\{R^2\}$, referred to as *relevant problems*, be composed of all problems from $\mathcal{O}^2$, each set in a dynamic MOE that has at least one new task. Using the notations in Equations 8.16 and 8.15, all problems from $\{R^2\}$ correspond to treatments with each having at least one component/factor, which represents a number of new tasks, greater than zero. Let the set $\{\mathbf{t}\}$ of all of these treatments be referred to as *relevant treatments*. It is assumed from here onwards that every problem from $\mathcal{O}^2$ and every treatment are elements of $\{R^2\}$ and $\{\mathbf{t}\}$ respectively.

### 8.3.3 Multiple Models

Let us now explain why a single polynomial model is not practically suitable to model the dynamic performance of any technique from $\mathcal{W} = \{$RI, CBAM, McBA, McBAR, McBAS, MedianBAR $\}$ (also defined in Table 5.1) for solving problems from $\mathcal{O}^2$. After several tests, the polynomial models of order less than five were observed as statistically inadequate to model the dynamic performance.

Let us now consider the case of building a fifth or higher order polynomial model of the dynamic performance. As elaborated in Section 4.4, each of the seven (zeroth to sixth) sub-problems of a considered problem from $\mathcal{O}^2$ is sequentially simulated and then solved for 30 times. The results of these simulations are used to determine a value that corresponds to the problem. The values that correspond to various problems from $\mathcal{O}^2$, as will be explained below, are used to determine the coefficients of a considered model. The number $Nc$ of coefficients of a fifth or higher order polynomial model with 12 factors is equal to or greater than 6,188 based on,

$$Nc = 1 + \sum_{n=1}^{Np} \frac{1}{n!} Nf(Nf+1)\cdots(Nf+n-1). \tag{8.20}$$

where $Np$ is the polynomial order; and $Nf$ is the number of factors in this polynomial. Considering that the number of unknown coefficients in a polynomial model must be lesser than the number of the experimentally-derived values necessary to determine these coef-

ficients [141], then the number of simulated sub-problems from $\mathcal{O}^2$ necessary to produce the required set of values to build a polynomial model of order higher than four must be more than 1,299,480 = 6,188 experimentally-derived values × 30 simulations × seven sub-problems. Thus as it takes a few minutes for our computing system to solve each of the simulated sub-problem, the polynomial model is impractical to build.

To summarise, polynomial models of order less than five with the 12 factors are statistically inadequate as models of the dynamic performance of the techniques in $\mathcal{W}$, and polynomial models of order greater than four with the 12 factors are impractical to build. Consequently, the model building process of the general method cannot practically be realised by using a single polynomial model. A model composed of several polynomials is therefore used to characterise the dynamic performance. It should be noted that, in the space with 12 factors as dimensions, the regions of applicability of these constituting polynomials are designed not to overlap. From here onwards, the composite model will be referred to as the *composite dynamical model* and will be described in Section 8.3.7.

### 8.3.4 Voxels

Let us now investigate the regions of applicability of the polynomials that comprise the composite dynamical model. Let $X = X_1 \otimes X_2 \otimes \cdots \otimes X_{12}$ be a discrete space where $X_i$ is a set (e.g. $\{0, 1, \ldots, 5\}$) of unique values that factor $x_i$ is allowed to take based on the ranges in expression 8.18, where $1 \leq i \leq 12$. Note that all treatments/points in $X$ correspond to all problems from $\mathcal{O}^2$. Let the set $X_{3c-8}$, which corresponds to the number $x_{3c-8}$ of new tasks, be broken into two parts,

$$\begin{aligned} X^w_{3c-8} &= \{0, 1, 2\}, \\ X^h_{3c-8} &= \{3, 4, 5\}, \end{aligned} \tag{8.21}$$

referred to as low and high ranges respectively. Further, let the factors $x^w_{3c-8}$ and $x^h_{3c-8}$ take values from the sets $X^w_{3c-8}$ and $X^h_{3c-8}$ respectively. Note that both of these factors

have three levels. Now, let the space $X$ be subdivided as,

$$X = \bigcup_{i,j,k,l=\{w,h\}} X_{i,j}^{k,l}, \tag{8.22}$$

where each of $i$, $j$, $k$ and $l$ is either $w$ or $h$; and,

$$X_{i,j}^{k,l} = X_1^i \otimes X_2 \otimes X_3 \otimes X_4^j \otimes X_5 \otimes X_6 \otimes X_7^k \otimes X_8 \otimes X_9 \otimes X_{10}^l \otimes X_{11} \otimes X_{12} \tag{8.23}$$

is a subset of $X$ referred to as *voxel*. Note that the subdivision of space $X$ is made along the factors $x_{3c-8}$ which represent the numbers of new tasks. Based on this subdivision, there are 16 voxels in $X$. Their labels are presented in Table 8.3 where "Low" and "High" (ranges) correspond to $w$ and $h$ in Equation 8.22 respectively; and the column headings – third to the sixth SOSA of a MOE – correspond to $i$, $j$, $k$ and $l$ in Equation 8.23 respectively. For example, voxel 12 is a set of points/treatments,

$$\begin{aligned} X_{h,w}^{h,h} \quad = \quad & X_{3\times3-8}^{h} \otimes X_2 \otimes X_3 \otimes X_{3\times4-8}^{w} \otimes X_5 \otimes X_6 \otimes \\ & X_{3\times5-8}^{h} \otimes X_8 \otimes X_9 \otimes X_{3\times6-8}^{h} \otimes X_{11} \otimes X_{12}. \end{aligned} \tag{8.24}$$

For each of the 16 voxels, a second order polynomial model that accurately predicts the responses (defined in Section 2.8) to all the treatments inside this voxel is built and is considered to correspond to this voxel. The composite dynamical model considered in Section 8.3.3 is constituted of models which correspond to all the 16 voxels. From here onwards, the constituting model is referred to as *dynamical sub-model*.

Let us now substantiate the division of the range of the number $x_{3c-8}$ of new tasks into the two sub-ranges, "High" and "Low", in Equation 8.21. As explained in Section 8.3.3, our preliminary tests showed that the polynomial models of order less than five were statistically inadequate to model the dynamic performance of any technique from $\mathcal{W}$ (defined above and in Table 5.1) for solving some problems from $\mathcal{O}^2$. This inadequacy may be due to the highly non-linear performances of the techniques from $\mathcal{W}$. This type of performances was observed in Section 6.5 when the techniques solved the problems set

Table 8.3: Range types for various voxels and SOSAs

| Voxel \ SOSA | 3rd | 4th | 5th | 6th |
|---|---|---|---|---|
| 1 | Low | Low | Low | Low |
| 2 | Low | Low | Low | High |
| 3 | Low | Low | High | Low |
| 4 | Low | Low | High | High |
| 5 | Low | High | Low | Low |
| 6 | Low | High | Low | High |
| 7 | Low | High | High | Low |
| 8 | Low | High | High | High |
| 9 | High | Low | Low | Low |
| 10 | High | Low | Low | High |
| 11 | High | Low | High | Low |
| 12 | High | Low | High | High |
| 13 | High | High | Low | Low |
| 14 | High | High | Low | High |
| 15 | High | High | High | Low |
| 16 | High | High | High | High |

in a MOE that undergoes increase in the number of tasks. The degree of non-linearity may be reduced if the range of the number of new tasks is reduced as was done in the last paragraph through Equation 8.21. With the reduction in non-linearity, a second order model could possibly adequately model the performance of any technique from $\mathcal{W}$ over the reduced range. The range reduction approach is akin to cutting a highly non-linear tenth order polynomial curve to obtain less non-linear sub-curves which could be approximated by second order polynomials. The domains of these sub-curves are analogous to the voxels.

### 8.3.5 Problems from $\mathcal{X}^2$

Let us now explore the design of the treatments (defined in Equation 8.16) whose corresponding responses are used to build the second order polynomial dynamical sub-models described in the last sub-section. Based on Section 2.8.1, the design must be rotatable in order for the dynamical sub-model obtained through it has even prediction accuracy across the voxel that corresponds to this sub-model. As the factors described in Section 8.3.2 take integral values, there is a choice between centre-faced CCD and BBD to design the treatments. Based on Section 2.8.1, centre-faced CCD is not rotatable. Hence, the nearly rotatable [141] BBD was chosen as the treatment design method for building the second order polynomial dynamical sub-models.

BBD produces treatments with factor coded values in the set,

$$X_{BBD} = \{-1, 0, 1\}. \tag{8.25}$$

The actual/natural values of the factors in expression 8.15 for the treatments described in Equation 8.16 are determined through,

$$x_{3c-7} = 2(x_{BBD} + 1) \tag{8.26}$$

and

$$x_{3c-6} = 2(x_{BBD} + 1), \tag{8.27}$$

where $x_{BBD} \in X_{BBD}$ and $3 \leq c \leq 6$. The factor values at the "Low" and "High" ranges are determined through,

$$x^{w}_{3c-8} = x_{BBD} + 1 \tag{8.28}$$

and

$$x^{h}_{3c-8} = x_{BBD} + 4 \tag{8.29}$$

respectively.

For each voxel in Table 8.3, and considering the 12 factors in Equation 8.16, the BBD design produces 204 treatments which are all elements of the voxel; none of these treatments is an irrelevant treatment (defined in Section 8.3.2); and these treatments are used to build a dynamical sub-model for the voxel following the procedure described in Section 2.8.4. At most 30 treatments are formed randomly; are different from the 204 treatments; are elements of the voxel; are not irrelevant treatments; and are used to verify the prediction accuracy (explored in Section 2.8.5) of the dynamical sub-model. Thus, with 16 dynamical sub-models needed to build one dynamical composite model, there are at most 3,744 ( = {204 BBDed treatments + 30 prediction accuracy testing treatments} $\times$ 16 voxels) treatments. These treatments have one-to-one correspondence with the problems that constitute $\mathcal{X}^2 \subset \mathcal{O}^2 \subset \mathcal{M}$. Thus, $\mathcal{X}^2$ has at most 3,744 problems, to be solved

by each technique from $\mathcal{W}$. Note that each problem from $\mathcal{X}^2$ is a relevant problem and since $\mathcal{X}^2 \subset \mathcal{O}^2$ then $\mathcal{X}^2 \subset \{R^2\}$. Each of the treatments is simulated 30 times (explained in Section 4.4). Thus, the maximum total number of problem simulations is 112,320 = 3,744 problems $\times$ 30 simulations; a practically computable size, thereby supporting the relevance of the voxelation (space subdivision) approach in Section 8.3.4.

### 8.3.6 Form of the Dynamical Sub-Model

Before describing the form of the dynamical sub-model, let us define a relative average performance of a pair of techniques $T_a, T_b \in \mathcal{W}$ (defined in Table 5.1). Suppose technique $T_a$ determines a set $A$ of solutions to a sub-problem $p_3^2$ (see Section 4.2.1 for notation) set in a simulation of the third snapshot of a dynamic MOE. Further, technique $T_b$ determines a set $B$ of solutions to the same sub-problem. From the sets $A$ and $B$, the differential set coverage $dSC\left(T_a, T_b\right)$ can be obtained using Equations 2.6 to 2.8. The differential set coverages, that correspond to sub-problems $p_4^2$ to $p_6^2$, can be determined in the same manner. The average of differential set coverages over $p_3^2$ to $p_6^2$ sub-problems is denoted as $\mathbf{E}^{3:6}\left[dSC\left(T_a, T_b\right)\right]$ and referred to as the *Averaged Relative Performance over Sub-problems* ($\mathrm{ARP^oS^b}$). Note that this average corresponds to a single simulation of a dynamical MOE in which the sub-problem is set. The average of $\mathbf{E}^{3:6}\left[dSC\left(T_a, T_b\right)\right]$ over 30 simulations of the MOE is denoted as $\mathbf{E}_s^{3:6}\left[dSC\left(T_a, T_b\right)\right]$ and referred to as the *Averaged Relative Performance over Sub-problems and Simulations* ($\mathrm{ARP^oS^bS^i}$). To indicate the influence of treatment $\mathbf{x}_m$ on the $\mathrm{ARP^oS^bS^i}$ and the voxel – labeled $m$ – to which this treatment belongs, $\mathrm{ARP^oS^bS^i}$ is relabelled as $\mathbf{E}_s^{3:6}\left[dSC_m\left(T_a, T_b, \mathbf{x}_m\right)\right]$.

The dynamical sub-model $\mathbf{M}_m\left(T_a, T_b\right)$ that models $\mathrm{ARP^oS^bS^i}$ is expressed as a second order polynomial,

$$\begin{aligned}\mathbf{E}_s^{3:6}\left[dSC_m\left(T_a, T_b, \mathbf{x}_m\right)\right] &= \hat{\alpha}^{a,b,m} + \textstyle\sum_{i=1}^{N_f} \hat{\beta}_i^{a,b,m} y_{i,m} \\ &+ \textstyle\sum_{i=1}^{N_f-1} y_{i,m} \sum_{j>i}^{N_f} \hat{\gamma}_{i,j}^{a,b,m} y_{j,m} + \sum_{i=1}^{N_f} \hat{\omega}_i^{a,b,m} y_{i,m}^2,\end{aligned} \tag{8.30}$$

where,

$$\mathbf{x}_m = \langle y_{1,m}, y_{2,m}, \ldots, y_{12,m} \rangle ; \tag{8.31}$$

for $3 \leq c \leq 6$ (i.e. the significant range of orders),

$$\begin{aligned} y_{3c-8,m} &= x_{3c-8}^{r(m)}, \\ y_{3c-7,m} &= x_{3c-7}, \\ y_{3c-6,m} &= x_{3c-6}, \end{aligned} \tag{8.32}$$

$r(m)$ is either $w$ or $h$ depending on which voxel, labelled $m$ in Table 8.3, the treatment $\mathbf{x}_m$ belongs; $N_f = 12$ is the number of factors; the pair of techniques $T_a$ and $T_b$ are both from $\mathcal{W}$; and the Greek letters are the constant coefficients of the model. Note that this dynamical sub-model is only valid at and corresponds to voxel $m$ defined in Equation 8.22. Given a treatment $\mathbf{x}_m$, the prediction of this sub-model is denoted as $\mathbf{M}_m(T_a, T_b, \mathbf{x}_m)$.

### 8.3.7 Composite Model

A composite dynamical model ( described in Section 8.3.3) $\mathbf{C}(T_a, T_b)$ is created for each pair of different techniques $T_a$ and $T_b$ in $\mathcal{W}$. It is composed of 16 dynamical sub-models; each corresponds to a unique voxel whose label is found in Table 8.3. Given a treatment $\mathbf{x}_m$, the prediction of the composite model is denoted as $\mathbf{C}(T_a, T_b, \mathbf{x}_m)$.

For any given treatment $\mathbf{x}_m \in X$ (defined in Equation 8.22), a voxel $m$ is determined to which this treatment belongs. Then, using Equation 8.30, $\mathbf{E}_s^{3:6}[dSC_m(T_a, T_b, \mathbf{x}_m)]$ can be obtained. This process could be viewed as though the composite dynamical model is the one used to predict $\mathbf{E}_s^{3:6}[dSC_m(T_a, T_b, \mathbf{x}_m)]$ given the treatment. To emphasise this concept, the index $m$ is sometimes dropped from $\mathbf{E}_s^{3:6}[dSC_m(T_a, T_b, \mathbf{x}_m)]$ in the foregoing to obtain $\mathbf{E}_s^{3:6}[dSC(T_a, T_b, \mathbf{x})]$. As $\mathbf{x}$ can be any element of $X$ then the composite dynamical model can be used to predict $\mathbf{E}_s^{3:6}[dSC(T_a, T_b, \mathbf{x})]$ at any treatment in $X$.

Based on the factor ranges in expression 8.18, the factors representing the number $x_{3c-8}$ of new tasks, the number $x_{3c-7}$ of newly broken resources of type $R_1$ and the

number $x_{3c-6}$ of newly broken resources of type $R_2$ have six, five and five levels, respectively, for each $c$ in the range $3 \leq c \leq 6$ (i.e. four SOSAs). Thus, the domain $X$ (defined in Equation 8.22) of the composite model $\mathbf{C}(\text{McBAR}, T_b)$ has $506,250,000 = (6 \text{ levels } \times 5 \text{ levels } \times 5 \text{ levels})^{4 \text{ SOSAs}}$ points/treatments. Because of the one-to-one correspondence between treatments and problems from $\mathcal{O}^2$, this number of points is the same number of problems whose solution attributes (e.g. $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$) can accurately be predicted by the composite model (built through the general method). Thus, the conclusions derived from this model could be accurate at number of problems much higher than the number (only 30) of problems considered in the limited method.

## 8.4 Dynamical Model Development

Let us now explore the results obtained by the general method on the development of dynamical sub-models that have the form expressed in Equation 8.30. This development follows the procedures discussed in Section 2.8.4. Each of the dynamical sub-models expresses $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$ (defined in Section 8.3.6) of a pair of techniques from $\mathcal{W}$ (defined in Table 5.1). Table 8.4 lists the selected pairs of techniques whose $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s are being modelled. In the foregoing, it is assumed that all of the listed techniques in Table 8.4 utilise their respective performance-optimising EA-parametric values in Table 8.2. Thus, based on the explanations in Section 8.1, each of the two paired techniques could perform at its best for solving any problem from $\mathcal{O}^2$.

Table 8.4: Pairs of techniques from $\mathcal{T}$ whose $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s are being modelled

| $T_a$ | $T_b$ |
|---|---|
| CBAM | RI |
| CBAM | McBAR |
| RI | McBA |
| RI | McBAR |
| RI | McBAS |
| RI | MedianBAR |
| McBA | McBAR |
| McBAR | McBAS |
| McBAR | MedianBAR |

### 8.4.1 Model Building

For a given pair of techniques $T_a$ and $T_b$ in Table 8.4 and a given voxel labelled $m$, a linear preliminary dynamical sub-model (defined in Equation 8.30) is fitted, following the procedure outlined in Section 2.8.4, to $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s determined from the solutions obtained by each of the techniques for independently solving some relevant problems from $\mathcal{X}^2 \subset \{R^2\}$. As explained in Section 8.3.5, these relevant problems correspond to the 204 BBDed treatments contained in voxel $m$ (voxel labels are listed in Table 8.3). The insignificant terms according to the t-test considered in Section 2.8.3 are screened-out from the preliminary dynamical sub-model. Following the same fitting and screening processes, a second preliminary dynamical sub-model, which has linear and interaction terms, is fitted to similar $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s and then screened. Then, a third preliminary dynamical sub-model, which has linear, interaction and quadratic terms, is fitted and then screened in a similar manner as in the last preliminary dynamical sub-model. The Box-Behnken design yields unaliased models of up to quadratic orders [165]. Thus, the three preliminary dynamical sub-models are unaliased. Among these preliminary dynamical sub-models, the F-test statistically significant and highest ordered dynamical sub-model is selected as the final dynamical sub-model used to predict $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s at treatments in and made to correspond with voxel $m$. Based on Section 8.3.6, this final dynamical sub-model is denoted as $\mathbf{M}_m(T_a, T_b)$. The accuracy of $\mathbf{M}_m(T_a, T_b)$ is verified by comparing its prediction to the observed $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s ($\mathbf{E}_s^{3:6}[dSC_m(T_a, T_b, \mathbf{x}_m)]$) determined from the solutions obtained by the pair of techniques for independently solving some relevant problems from $\mathcal{X}^2 \subset \{R^2\}$. These relevant problems correspond to the 30 prediction-accuracy testing treatments (e.g. $\mathbf{x}_m$) described in Section 8.3.5 and contained in voxel $m$.

The model building process just presented is applied to obtain the 16 final dynamical sub-models that correspond to all voxels described in Section 8.3.2. Thereby, one composite dynamical model $\mathbf{C}(T_a, T_b)$ (see Section 8.3.7 for notation) is obtained that corresponds to the pair of techniques $T_a$ and $T_b$. The process of obtaining a composite model is applied to build the nine composite dynamical models that correspond to

the nine pairs of techniques in Table 8.4. Thus 144 (= 16 sub-models × 9 pairs) final dynamical sub-models are built.

The *Analysis of Variance* (ANOVA) results of building the 144 final dynamical sub-models are found in `http://cs.adelaide.edu.au/~optlog/research/RSM/voxelsANOVA.pdf`. Due to space limitation, only the ANOVA results of building the final dynamical sub-model $\mathbf{M}_7$ (McBAR MedianBAR) are presented here and are found in Table 8.5. In this table, the model types are linear, linear plus interaction, and quadratic polynomials. The screened linear, linear plus interaction, and quadratic preliminary forms of $\mathbf{M}_7$ (McBAR, MedianBAR) have F-test p-values of $0.216263 > 0.05$, $0.009535 < 0.05$ and $0.009587 < 0.05$ respectively. Thus, based on the model selection guideline explained in Section 2.8.2, the quadratic preliminary dynamical sub-model is selected as the final $\mathbf{M}_7$ (McBAR, MedianBAR) dynamical sub-model. Shaded model types in Table 8.5 and in the tables in `http://cs.adelaide.edu.au/~optlog/research/RSM/voxelsANOVA.pdf` are the model types of the 144 final dynamical sub-models elaborated in the last paragraph. All of the final dynamical sub-models have F-test p-values less than 0.05 and hence are all significant.

Table 8.5: ANOVA of final $\mathbf{M}_7$ (McBAR, MedianBAR) sub-model

| $T_a$ | $T_b$ | Model Type | dfr | SSr | MSSr | dfe | SSe | MSSe | F-value | p-value |
|---|---|---|---|---|---|---|---|---|---|---|
| McBAR | MedianBAR | Linear | 12 | 0.144408 | 0.012034 | 181 | 1.663385 | 0.00919 | 1.309466 | 0.216263 |
| | | Interaction | 55 | 0.719574 | 0.013083 | 138 | 1.088219 | 0.007886 | 1.659111 | 0.009535 |
| | | Quadratic | 78 | 0.945001 | 0.012115 | 115 | 0.862792 | 0.007503 | 1.614839 | 0.009587 |

### 8.4.2 Data Diagnostics

Each of the final dynamical sub-models described in the last sub-section is diagnosed for adequacy (defined in Section 2.8.2). Due to space limitation, all of the adequacy test results on each of the final dynamical sub-models are found in `http://cs.adelaide.edu.au/~optlog/research/RSM/DOEfigures.pdf`. Further, only the adequacy test results on the final dynamical sub-model $\mathbf{M}_7$ (McBAR, MedianBAR) (described in the last sub-section) are shown here and are illustrated in Figures 8.10 to 8.15. The leverages

of the members of a data set illustrated in Figure 8.10 are below twice its average. The members of this data set are $ARP^{o}S^{b}S^{i}$s used to build the final dynamical sub-model $\mathbf{M}_7\,(\text{McBAR}, \text{MedianBAR})$. Figure 8.11 shows the Cook's distances of the members as being mostly below 0.026 (derived from Equation 2.22). Three outliers (defined in Section 2.8.2), or 1.47% of the members, whose Cook's distances are above 0.06, are removed from the data set. Almost all of the members have Difference in Fits (DFFITS) between $\pm 2$, as depicted in Figure 8.12. Using DFFITS as a basis, no outlier in the data set is removed. Figure 8.13 exhibits a nearly linear normal plot of the studentised residuals derived from the data set and $\mathbf{M}_7\,(\text{McBAR}, \text{MedianBAR})$. Each member of the data set which corresponds to a treatment is plotted in Figure 8.14 against the value predicted by $\mathbf{M}_7\,(\text{McBAR}, \text{MedianBAR})$ at the same treatment. The plotted points are around the $45^o$ line in this figure. The plot of the predicted values and their corresponding previously-considered studentised residuals is exhibited as being random in Figure 8.15. Thus, Figures 8.10 to 8.15 prove that the final dynamical sub-model $\mathbf{M}_7\,(\text{McBAR}, \text{MedianBAR})$ passes all the diagnostic tests enumerated in Section 2.8.2. From each of the data sets used to build the 144 final dynamical sub-models deliberated in Section 8.4.1, 1.5% or less of its members are removed based only on Cook's distance. Each of the 144 final dynamical sub-models passed all of the diagnostic tests.

### 8.4.3 Model Predictions

After providing information in the last two sub-sections on the general adequacy and significance of the final dynamical sub-models, let us now explore the general predictive accuracy of these dynamical sub-models. Due to space limitation, only the predictive accuracy of the final dynamical sub-model $\mathbf{M}_7\,(\text{McBAR}, \text{MedianBAR})$ is presented here. As declared in Section 8.3.5, at most 30 relevant treatments from $\mathcal{X}^2$ (defined in Section 8.3.5) are randomly generated – all contained in voxel 7 at which the dynamical sub-model is valid – and are used to verify the prediction accuracy of the dynamical sub-model. Let the set of these treatments be denoted as $T = \left\{\mathbf{t}_7^j\right\}$ where, the subscript 7 indicates to

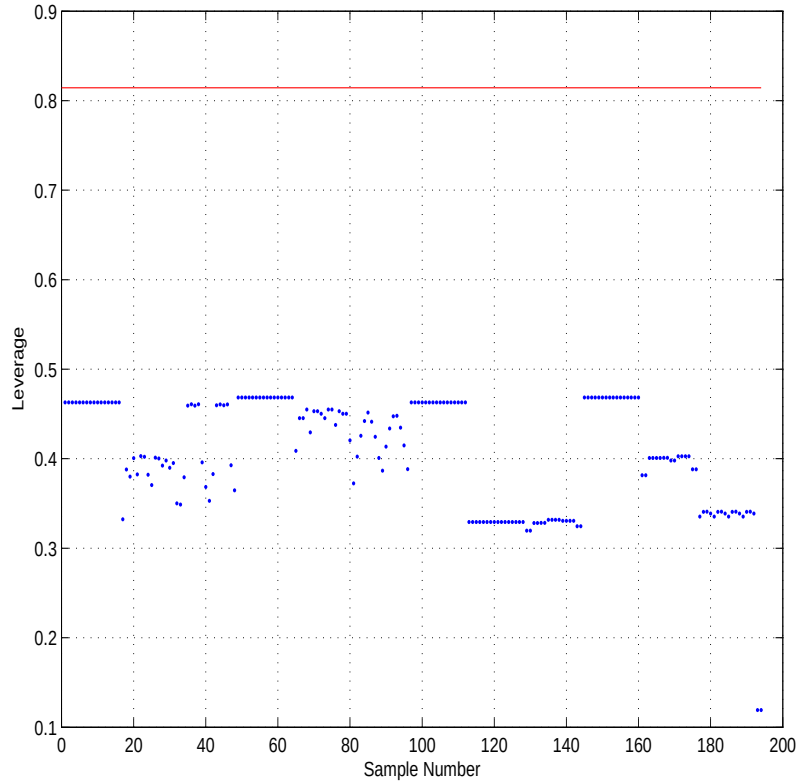



Figure 8.10: Leverage for the final $\mathbf{M}_7$ (McBAR, MedianBAR) sub-model

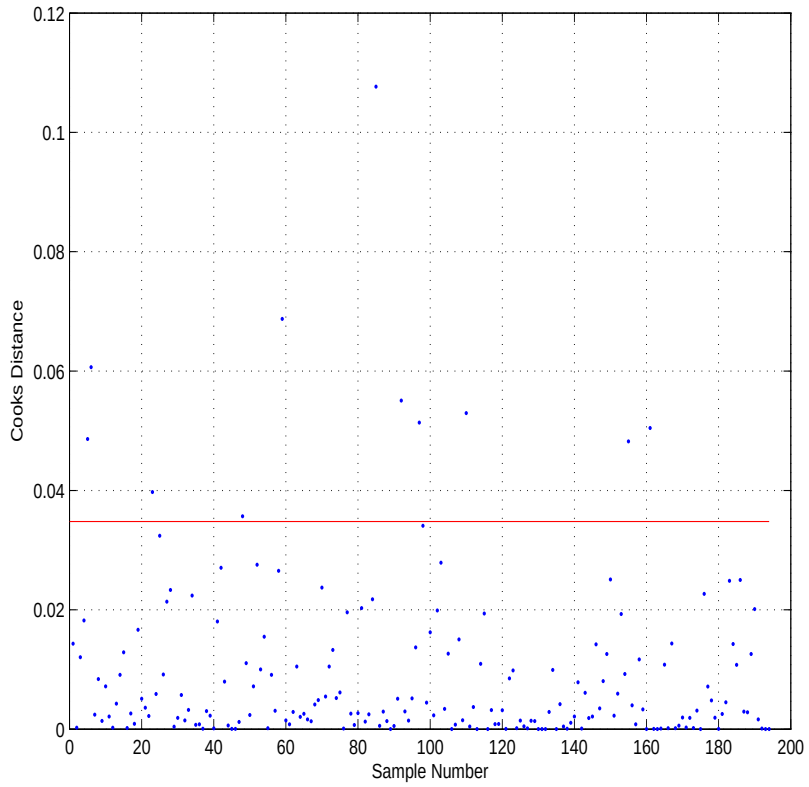



Figure 8.11: Cooks Distance for the final $\mathbf{M}_7$ (McBAR, MedianBAR) sub-model

which voxel these treatments belong; and the superscript $j$ has the range $1 \leq j \leq |T|$. The final dynamical sub-model $\mathbf{M}_7$ (McBAR, MedianBAR) (has a form expressed in Equation 8.30) is used to predict $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$ at each treatment $\mathbf{t}_7^j \in T$. The predicted $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s are plotted as the middle curve in Figure 8.16. The upper and lower curves in this figure bound the 95% prediction interval (defined in Section 2.8.5) of the predicted $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s. The horizontal axis is the index $j$ of a relevant problem from $\mathcal{X}^2$ (defined in Section 8.3.5) denoted as $N^2(j,7)$ that corresponds to the relevant treatment $\mathbf{t}_7^j$.

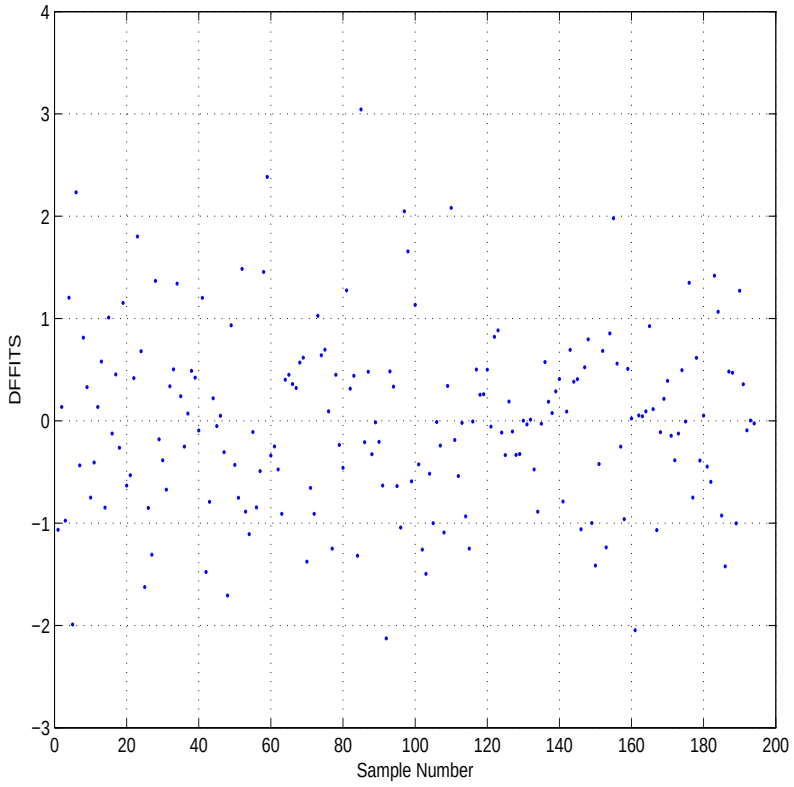



Figure 8.12: DFFITS for the final $\mathbf{M}_7$ (McBAR, MedianBAR) sub-model

Based on Section 4.4, every relevant problem $N^2(j,7)$, $1 \leq j \leq |T|$, is simulated 30 times. Let $C_7^j$ denotes the group of $\text{ARP}^{\text{o}}\text{S}^{\text{b}}$s (defined in Section 8.3.6) derived from the solutions obtained by McBAR and MedianBAR for independently solving all simulations of the relevant problem $N^2(j,7)$. The standard deviation of all $\text{ARP}^{\text{o}}\text{S}^{\text{b}}$s in $C_7^j$ is indicated by the length of the vertical strip straight above problem/instance index $j$ in Figure 8.16. The average of all $\text{ARP}^{\text{o}}\text{S}^{\text{b}}$s in $C_7^j$ is the vertical coordinate of the centre of the vertical strip. Note that this average is equivalent to $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$ based on the definitions in Section

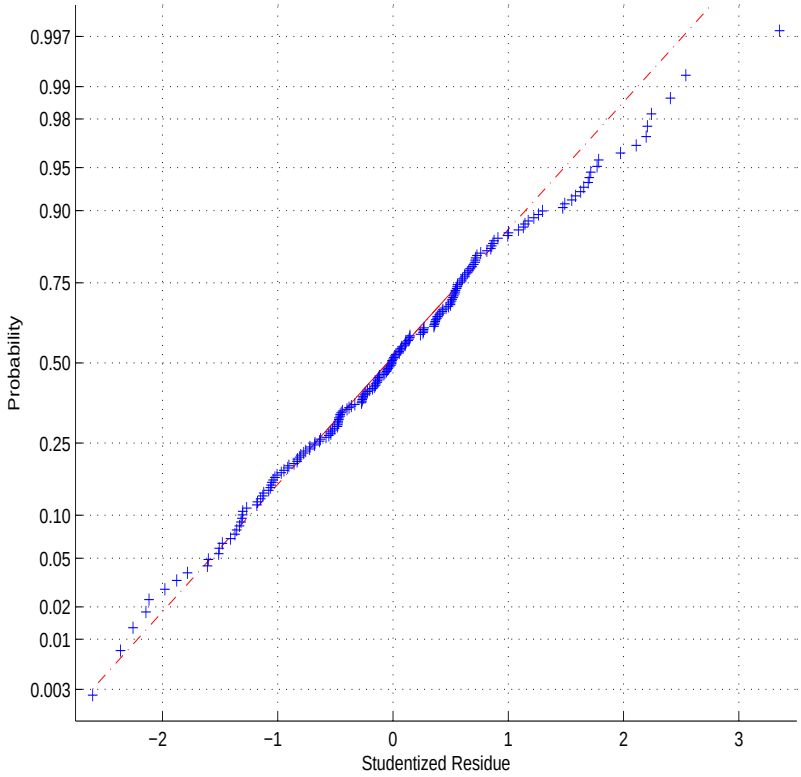



Figure 8.13: Normal plot of Studentized Residue for the final $\mathbf{M}_7$ (McBAR, MedianBAR) sub-model

8.3.6. Further, both $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$ and the standard deviation of $\text{ARP}^{\text{o}}\text{S}^{\text{b}}$s are derived from differential set coverage as explained in Section 8.3.6. Thus, the vertical axis is labelled as differential set coverage.

As evident in Figure 8.16, 90% ($\approx 26\times100\%/29$) of the simulation-obtained $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s (represented by the centres of the vertical strips) declared in the last paragraph are within the 95% prediction interval. Therefore, this result supports the accuracy of the final dynamical sub-model $\mathbf{M}_7$ (McBAR, MedianBAR) for predicting $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s. Plots of the same format as Figure 8.16 that correspond to all the final dynamical sub-models described in Section 8.3.5 are found in `http://cs.adelaide.edu.au/~optlog/research/RSM/DOEfigures.pdf`. Each of these plots has characteristics that also support the prediction accuracy of its corresponding final dynamical sub-model for predicting $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s at the prediction-accuracy testing treatments located in the voxel that corresponds to this final dynamical sub-model.

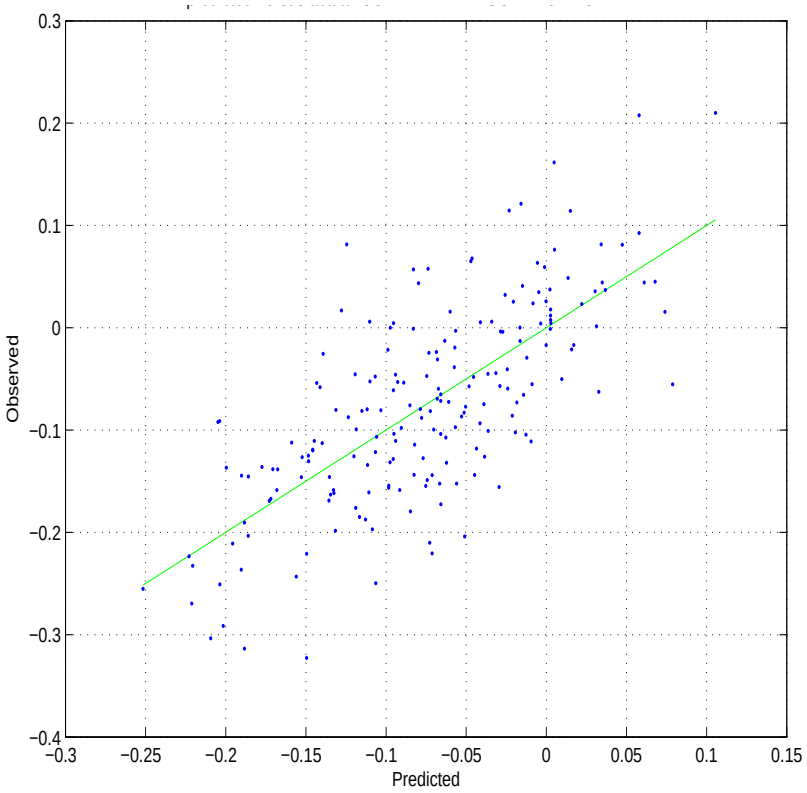



Figure 8.14: Predicted versus actual for the final $\mathbf{M}_7$ (McBAR, MedianBAR) sub-model

All the final dynamical sub-models described in Section 8.3.5 have now been demonstrated as being adequate, significant and prediction-accurate. Thus they can now be used for the other processes of the general method that are discussed in the succeeding sections. Before using them, let us discuss the following.

Consider the relevant problem $N^2(1,7)$ whose corresponding vertical strip is straight above index 1 in Figure 8.16. As implied above, this problem corresponds to treatment $\mathbf{t}_7^1 \in T$ and is simulated 30 times, thereby obtaining the set $C_7^1$ of ARP$^{\text{o}}$S$^{\text{b}}$s. These ARP$^{\text{o}}$S$^{\text{b}}$s are plotted in Figure 8.17 against simulation indices. The distance between the standard deviation lines (lowest and highest solid horizontal lines) in this figure is the standard deviation of the elements of $C_7^1$. The levels of the lowest and highest solid horizontal lines are equal, respectively, to those of the top and bottom of the vertical strip indexed 1 in Figure 8.16. The level of the middle horizontal solid line in Figure 8.17 is the average of elements in $C_7^1$ and is equal to the level of the centre of the strip. Figure 8.17 depicts the elements of $C_7^1$ as having a large standard deviation relative to their range

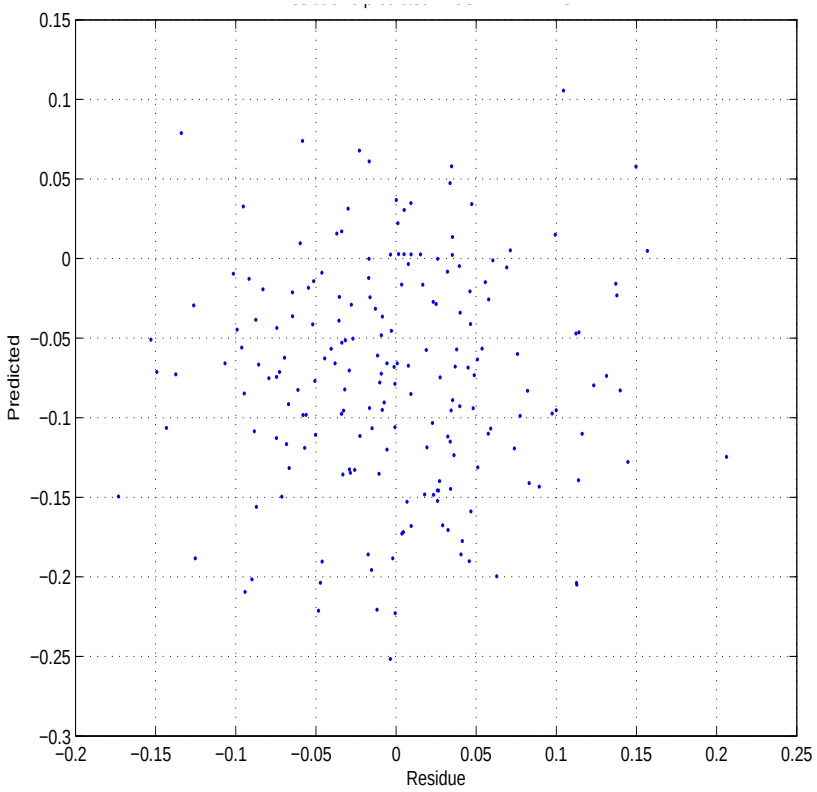



Figure 8.15: Residue versus predicted for the final $\mathbf{M}_7$ (McBAR, MedianBAR) sub-model

of values, -1 to 1 based on Equation 2.9. This magnitude is also demonstrated by the length of the vertical strip indexed 1. The vertical strips not indexed by 1 in Figure 8.16 generally depict large standard deviation of their respective sets of ARP$^{\text{o}}$S$^{\text{b}}$s. All plots in `http://cs.adelaide.edu.au/~optlog/research/RSM/DOEfigures.pdf` of the same format as Figure 8.16 also manifest profiles of the standard deviation of ARP$^{\text{o}}$S$^{\text{b}}$s to be generally similar to that of Figure 8.16.

Let us now investigate this last result. As noted in Section 4.4, the duration of any unfinished task in one simulation of a relevant problem from $\{R^2\}$ could differ from that of a similarly-IDed unfinished task in other simulations of the same relevant problem. Thus, different simulations of this problem could involve different dynamic scheduling problems. The solutions to these different scheduling problems could yield largely different ARP$^{\text{o}}$S$^{\text{b}}$s across the simulations, thereby obtaining a large standard deviation of these ARP$^{\text{o}}$S$^{\text{b}}$s. Note, however, that the standard deviation of the quantities, derived from the solutions obtained by an EA-based algorithm solving a similar problem at different runs

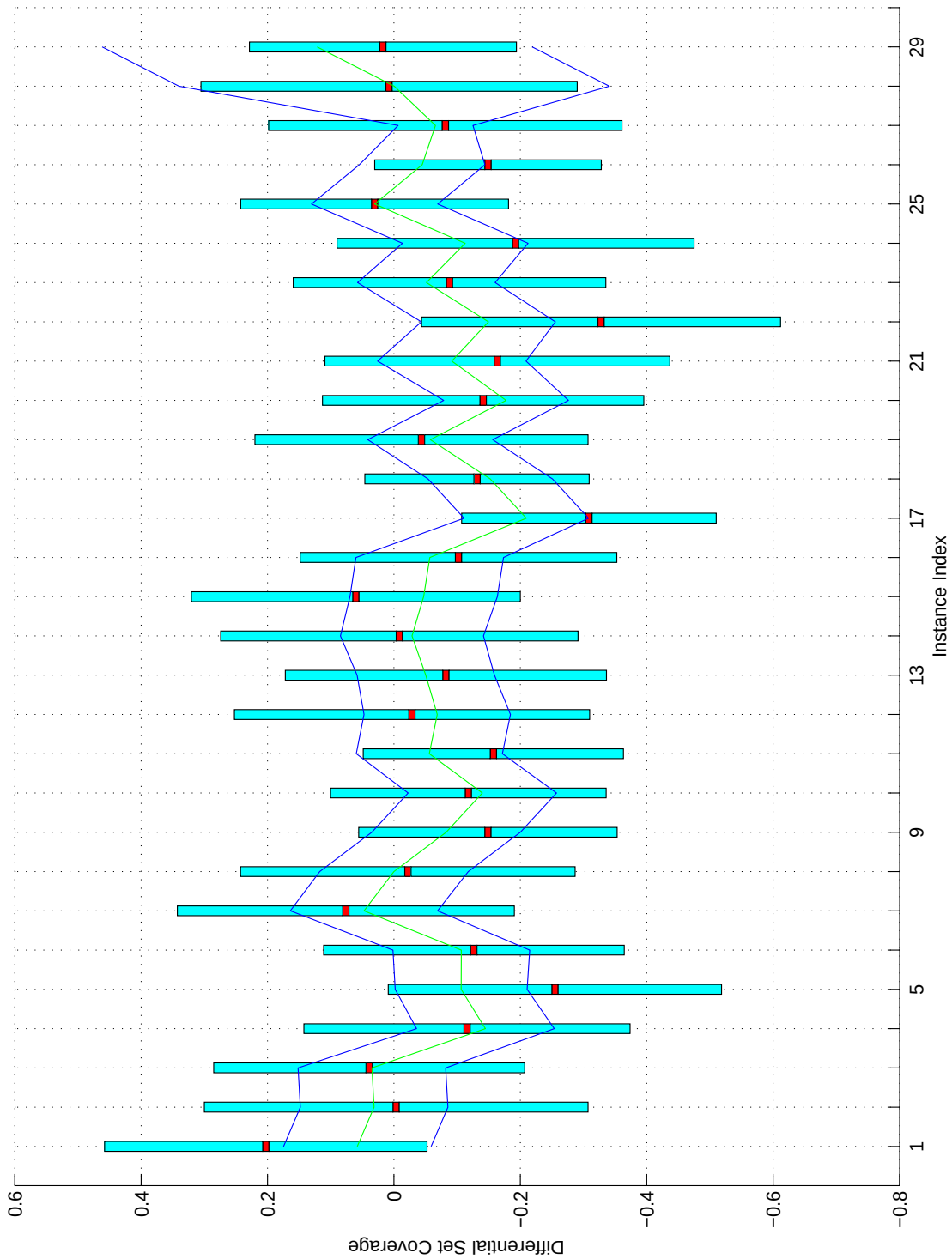



Figure 8.16: Predictions of the final $\mathbf{M}_7$ (McBAR, MedianBAR) sub-model

(simulations), could still be large as that reported in [165].

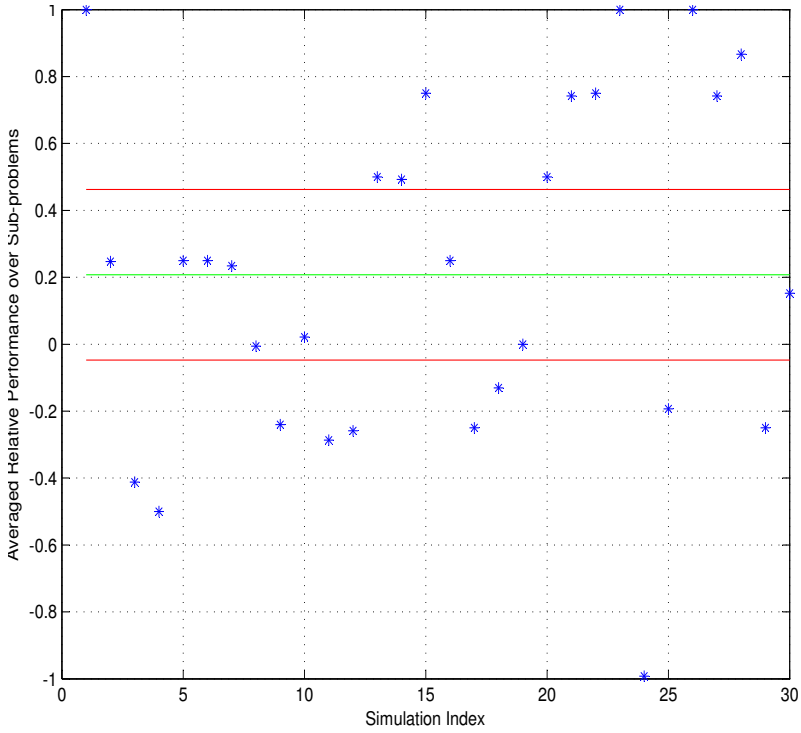



Figure 8.17: ARP$^{o}$S$^{b}$s from the simulations of the problem instance indexed 1

## 8.5 Legitimising the Components of McBAR

Let us now deliberate on the procedure undertaken by the general method (i.e. using RSM-built models) to legitimise the algorithmic components of McBAR. In Section 6.3, the legitimisation of the components involves the comparison of the performance of McBAR, for solving the relevant problems from $\{R^2\}$, to those of techniques from $\mathcal{K} \subset \mathcal{T}$. In this section, the other subset $\mathcal{W} = \{$RI, CBAM, McBA, McBAR, McBAS, MedianBAR$\}$ (also defined in Table 5.1) of $\mathcal{T}$ is utilised for the legitimisation instead of $\mathcal{K}$. As the dynamical composite model can be utilised to compare the performance of McBAR and to that of any of the other techniques, it is utilised for the legitimisation by the general method.

The first step in the legitimisation of the components of McBAR is the application of MCS, where 10,000 relevant treatments are randomly selected from $\{\mathbf{t}\}$ (defined in Section

8.3.2). The composite model $\mathbf{C}\left(\text{McBAR}, T_b\right)$ is then used to predict ARP$^{\text{o}}$S$^{\text{b}}$S$^{\text{i}}$s at these selected treatments where, $T_b \in \mathcal{W}$. These ARP$^{\text{o}}$S$^{\text{b}}$S$^{\text{i}}$s are then averaged where the result is denoted as $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b\right)\right]$ and referred to as *Average Relative Performance over Treatments* (ARP$^{\text{o}}$T).

Referring to Section 8.3.7, the domain of any composite model has 506,2500,000 treatments. Thus, using the composite model to predict ARP$^{\text{o}}$S$^{\text{b}}$S$^{\text{i}}$s at all of these treatments to determine $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b\right)\right]$ is computationally expensive. Thus, MCS is applied instead. The ARP$^{\text{o}}$Ts that correspond to all pairs of McBAR and the other techniques in $\mathcal{W}$ are presented in Table 8.6. Before discussing this table, however, let us investigate several aspects.

Based on Section 2.5.2, the performance of McBAR is better than that of another technique $T_b$ for solving a given problem if $dSC(\text{McBAR}, T_b) > 0$. Applying the average $\mathbf{E}_{s,t}^{3:6}\left[\bullet\right]$ operator on the expression $dSC(\text{McBAR}, T_b) > 0$ yields $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b\right)\right] > 0$. Thus, we define the performance of McBAR as better than that of $T_b$ for solving the relevant problems from $\{R^2\}$ if $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b\right)\right] > 0$. Based on the same section, $dSC(\text{McBAR}, T_b) = -dSC(T_b, \text{McBAR})$. Applying the average $\mathbf{E}_{s,t}^{3:6}\left[\bullet\right]$ operator to the last equation yields, $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b\right)\right] = -\mathbf{E}_{s,t}^{3:6}\left[dSC\left(T_b, \text{McBAR}\right)\right]$.

Table 8.6: ARP$^{\text{o}}$Ts between McBAR and other techniques in $\mathcal{T}$

| $T_a$ | $T_b$ | $E_{s,t}^{3:6}[dSC(T_a,T_b)]$ | ± | $\sigma_{s,t}^{3:6}[dSC(T_a,T_b)]$ |
|---|---|---|---|---|
| CBAM | McBAR | -0.8216 | ± | 0.1140 |
| RI | McBAR | -0.8275 | ± | 0.1016 |
| McBA | McBAR | -0.0853 | ± | 0.1541 |
| McBAR | MedianBAR | 0.0341 | ± | 0.1180 |
| McBAR | McBAS | 0.1907 | ± | 0.1740 |

The first and second columns of Table 8.6 comprise the pair of techniques from $\mathcal{W}$ used in the $\mathbf{E}_{s,t}^{3:6}\left[\bullet\right]$ operator. In the third column, the value to the left of the ± sign is $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(T_a, T_b\right)\right]$ which relates to the techniques in the first and the second columns that are techniques $T_a$ and $T_b$ respectively. Further, the value to the right of the ± sign is the standard deviation $\sigma_{s,t}^{3:6} dSC\left(T_a, T_b\right)$ of ARP$^{\text{o}}$S$^{\text{b}}$S$^{\text{i}}$s predicted by the composite model $\mathbf{C}\left(T_a, T_b\right)$ as part of the MCS noted in this sub-section.

Let us now discuss the results presented in Table 8.6. From this table, $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\right.\right.$ CBAM, McBAR)] = -.8216 which implies that $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{CBAM}\right)\right] = 0.8216 > 0$ based on the above explanations. Thus, based on the definition in the second to the last paragraph, the performance of McBAR is better than that of CBAM for solving the problems from $\{R^2\}$. In addition, $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{McBAS}\right)\right] = .1907 > 0$ which implies that the performance of McBAR is better than that of McBAS for solving similar problems. Note that $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{MedianBAR}\right)\right]$ is just above zero. Further, the standard deviation $\sigma_{s,t}^{3:6} dSC\left(T_a, T_b\right) = 0.118$ is large relative to the average $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{MedainBAR}\right)\right] = 0.0341$. This demonstrates that the performance of McBAR is not significantly better than that of MedianBAR for solving the problems. Based on the legitimisation concept presented in Section 4.4 and based on Table 8.6, the superior performance of McBAR over that of CBAM, RI, McBA, MedianBar and McBAS legitimises the following components of McBAR respectively: the mapping operation $\mathcal{F}$ explained in Section 5.5; the memory-based approach; the random component of the initial population defined in Section 5.6.6; the use of mean to compute for centroid in Equation 5.1; and the minimal centroid repair method $\mathcal{N}$ investigated in Section 5.6.5.

## 8.6 Dynamics of Average Performance

Let us now explore how the general method manifests the dynamical performance of McBAR, relative to those of the other techniques in $\mathcal{W}$ (defined in Table 5.1) for solving the relevant problems from $\{R^2\}$. This performance is measured in terms of ARP$^o$T defined in Section 8.5 and is relabelled as $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \beta, n, c\right)\right]$, where $T_b \in \mathcal{W}$ to indicate its dependency on some variables and on a factor name. The factor $\beta$ is the dynamical factor of interest that could be the number of new tasks, the number of newly broken resources of type $R_1$, or the number of newly broken resources of type $R_2$ where $\beta$ is substituted in $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \beta, n, c\right)\right]$ as either $\tau$, $\rho_1$ or $\rho_2$ respectively. Further, the variables are the SOSA $c$ of a MOE in which a problem from $\{R^2\}$ is set; and the value $n$ of factor $\beta$ at a given $c$. Based on Sections 4.3.1 and 4.3.2, the value of $n$ is zero

at the zeroth to the second SOSA of the MOE. The SOSA $c$ considered for the analysis in this sub-section is in the range $3 \leq c \leq 6$, i.e. in the significant range of orders defined in Section 4.3.1.

The process of determining $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \beta, n, c\right)\right]$ is explained next through an example. Let the relevant treatment $\mathbf{x}$ in $\{\mathbf{t}\}$ (defined in Section 8.3.2) be expressed as,

$$\mathbf{x} = \langle x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12} \rangle . \tag{8.33}$$

Further, let the dynamical factor of interest be the number of new tasks (i.e. $\beta = \tau$). To determine $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \tau, n, c\right)\right]$s at all pairs of $n$ and $c$, where $1 \leq n \leq 5$ and $3 \leq c \leq 6$:

1. Set $n = 1$ and $c = 3$.

2. Set the factor $x_1$ in Equation 8.33 to $n$ while set the factors $x_4$, $x_7$ and $x_{10}$ to zero. Notice that, based on expression 8.15, the factors $x_1$, $x_4$, $x_7$ and $x_{10}$ represent the numbers of new tasks appearing, respectively, from the third to the sixth SOSA of the MOE in which the problem from $\{R^2\}$ that corresponds to treatment $\mathbf{x}$ is set.

3. Set all the other factors in treatment $\mathbf{x}$ to random integral values that satisfy the ranges in expression 8.18. This step completes the specification of the treatment.

4. Predict an $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$ given the treatment $\mathbf{x}$ using the composite model $\mathbf{C}\left(\text{McBAR}, T_b, \mathbf{x}\right)$ as explained in Section 8.3.7.

5. Repeat 1,000 times the steps 2 to 4 to obtain 1,000 treatments in $\{\mathbf{t}\}$, then use these treatments to predict 1,000 $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s through the composite model.

6. Take the average of these $\text{ARP}^{\text{o}}\text{S}^{\text{b}}\text{S}^{\text{i}}$s to obtain $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \tau, n, 3\right)\right]$.

7. Repeat the steps 2 to 6 with $c$ set to four, five and then six to obtain $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \tau, n, 4\right)\right]$, $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \tau, n, 5\right)\right]$ and $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \tau, n, 6\right)\right]$ respectively. However, at step 2 of each repetition, the factors $x_1$, $x_4$ until

$x_{3c-8}$ (refer to expression 8.18 for notation) are each set to $n$ while the factors $x_{3(c+1)-8}$, $x_{3(c+2)-8}$ until $x_{10}$ are each set to zero if $c$ is not equal to six.

8. Repeat step 7 with $n = 2$, $n = 3$, $n = 4$ and then $n = 5$.

Let us legitimise steps 5 and 6. As implied in step 3, the values of the factors $x_2$, $x_3$, $x_5$, $x_6$, $x_8$, $x_9$, $x_{11}$ and $x_{12}$ in Equation 8.33 are randomly chosen, effectively generating pseudo-random treatments. Based on expression 8.18, each of these factors has five levels. As the values of the factors $x_1$, $x_4$, $x_7$ and $x_{10}$ are fixed in the last enumerated procedure, there is a pool of 390,625 $(= (5 \times 5)^4)$ treatments from which the pseudo-random generation is applied. Enormous computational cost could be spent if, in step 4 of the enumerated procedure, ARP$^{\text{o}}$S$^{\text{b}}$S$^{\text{i}}$s are predicted through the composite model $\mathbf{C}\left(\text{McBAR}, T_b\right)$ at all treatments in the pool. To reduce this cost, MCS is applied and is implemented as steps 5 and 6 of the enumerated procedure.

Before discussing the dynamical performance of McBAR, let us consider the following points. As defined in Section 8.5, the performance of McBAR is better than that of $T_b$ for solving the relevant problems from $\{R^2\}$ if $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b\right)\right] > 0$. Further, $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b\right)\right] = -\mathbf{E}_{s,t}^{3:6}\left[dSC\left(T_b, \text{McBAR}\right)\right]$. It could be shown that these points hold when $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b\right)\right]$ is relabelled as $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \beta, \quad n, c\right)\right]$. Further, applying the averaging operator $\mathbf{E}_{s,t}^{3:6}\left[\bullet\right]$ to the range $-1 \leq dSC(\text{McBAR}, T_b) \leq 1$ yields $-1 \leq \mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b\right)\right] \leq 1$.

The dynamics of the average ARP$^{\text{o}}$T relative performance over treatments of McBAR to the other techniques in $\mathcal{W}$ (defined in Table 5.1) are illustrated in Figures 8.18 to 8.27. In each of these figures, the colour of each rectangle denotes a value of ARP$^{\text{o}}$T that is based on the colour bar located at the right and beside its corresponding figure. Further, the horizontal axis is the value $n$ of the dynamical factor $\beta$ and the vertical axis is the SOSA $c$. Figure 8.18 illustrates the dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{CBAM}, \tau, n, c\right)\right]$, where the dynamical factor $\beta$ of interest is $\tau$, i.e. $n$ denotes the number of new tasks. It illustrates $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{CBAM}, \tau, n, c\right)\right]$ as being positive at all number of new tasks and SOSAs. Based on the last paragraph, this implies that the performance of McBAR

is better than that of CBAM for solving problems from $\mathcal{O}^2$, that are taken into account in determining the plotted ARP$^o$Ts. However, as will be demonstrated, this superiority varies with SOSA and the number of new tasks.

The ARP$^o$Ts determined at step 7 of the enumerated procedure correspond to rectangles at the leftmost column of Figure 8.18 which corresponds to $n = 1$. This column illustrates that the performance of McBAR becomes increasingly superior to that of CBAM as SOSA $c$ increases from the third to the sixth, where one new task appears from the third to the $c^{th}$ SOSA and none after the $c^{th}$ SOSA based on the last enumeration in this section. This dynamical performance is different from the case where the number of new tasks is greater than one, as illustrated in the other columns. The bottom row of rectangles (corresponds to the third SOSA) demonstrates the performance of McBAR as being increasingly superior to that of CBAM as the number $n$ of new tasks increases from one to five at the third SOSA. This dynamical performance differs from those depicted by the other rows which correspond to SOSA $c > 3$.

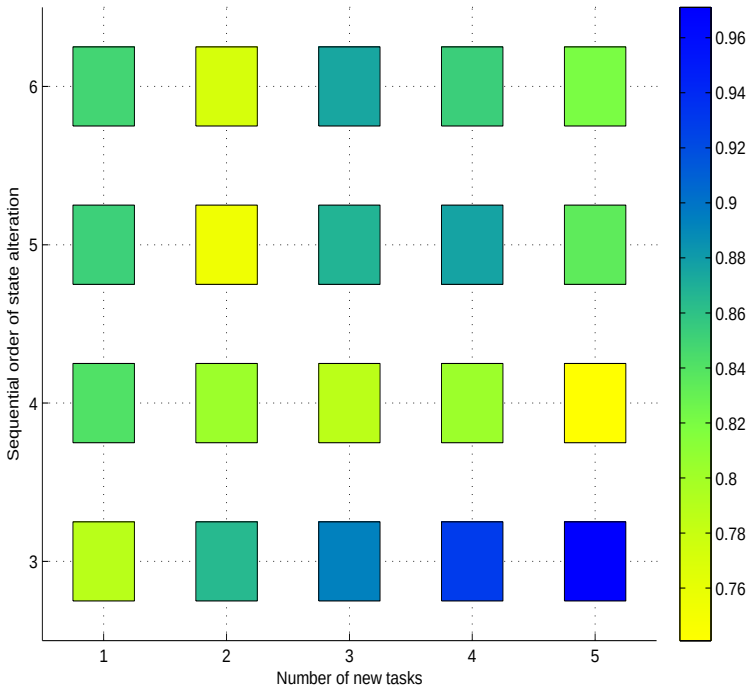



Figure 8.18: Dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{CBAM}, \tau, n, c\right)\right]$ under the changes in the number of tasks

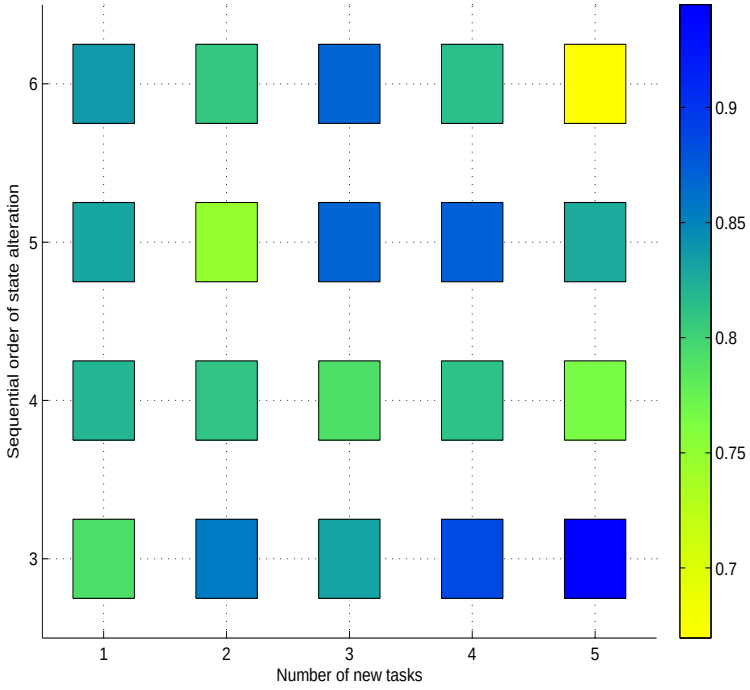



Figure 8.19: Dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{RI}, \tau, n, c\right)\right]$ under the changes in the number of tasks

The generic performance dynamics illustrated in Figure 8.18 is also observed in: Figure 8.19 which depicts the performance dynamics of McBAR relative to that of RI; Figure 8.20 which shows the performance dynamics of McBAR relative to that of McBA; and Figure 8.21 which illustrates the performance dynamics of McBAR relative to that of McBAS. However, Figure 8.20 illustrates ARP$^{o}$Ts as being near zero while Figures 8.18, 8.19 and 8.21 generally show ARP$^{o}$Ts much greater than zero. This difference implies that, although the performance of McBAR is generally superior to that of McBA, this superiority is not as great as to that of CBAM, RI and McBAS.

The dynamics of ARP$^{o}$T between McBAR and MedianBAR is illustrated in Figure 8.22. Note that in order to clearly distinguish the polarities (positive or negative) of ARP$^{o}$Ts, the colour bar for this figure is different to those of the previously described ARP$^{o}$T-related figures. In Figure 8.22, the performance of McBAR becomes generally superior to that of MedianBAR as the values of $n$ and $c$ increase. However, there are pairs of $n$ and $c$ at which the performance of McBAR is inferior to that of MedianBAR.

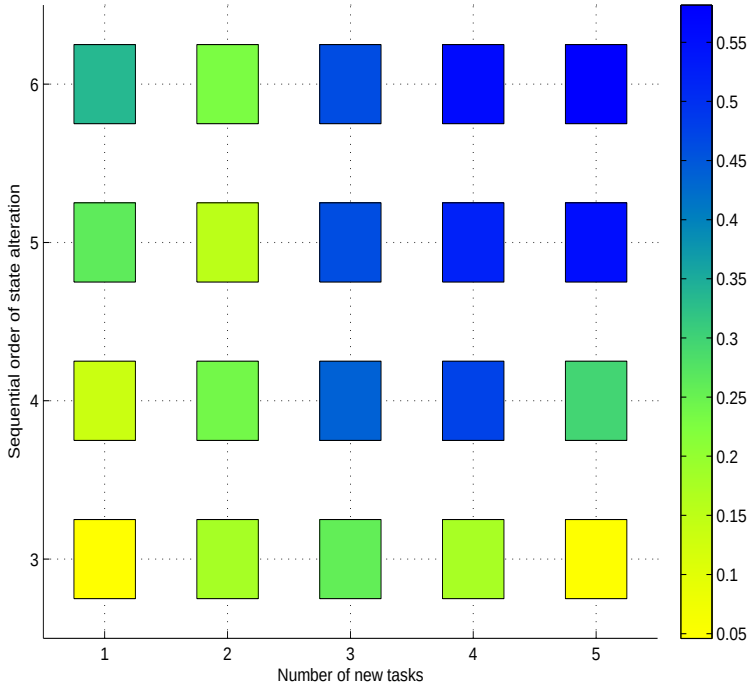



Figure 8.20: Dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{McBA}, \tau, n, c\right)\right]$ under the changes in the number of tasks

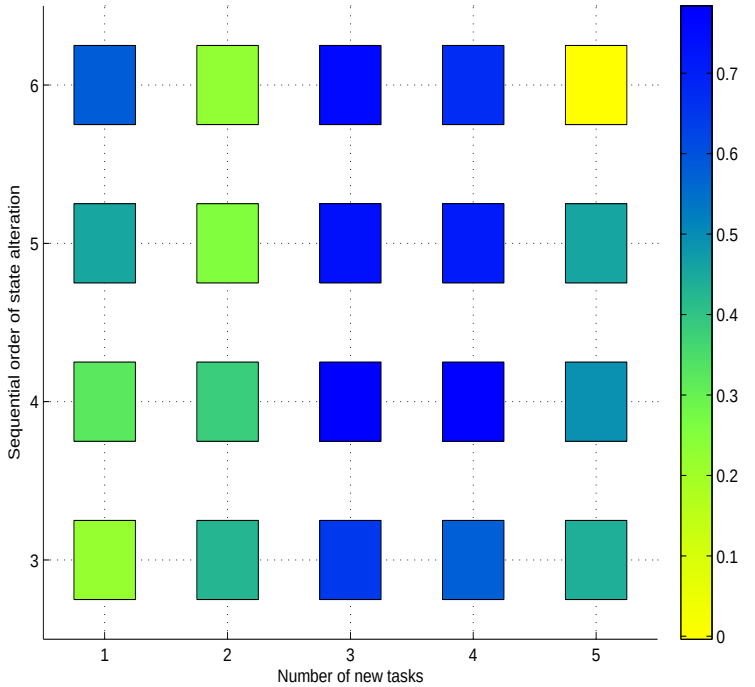



Figure 8.21: Dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{McBAS}, \tau, n, c\right)\right]$ under the changes in the number of tasks

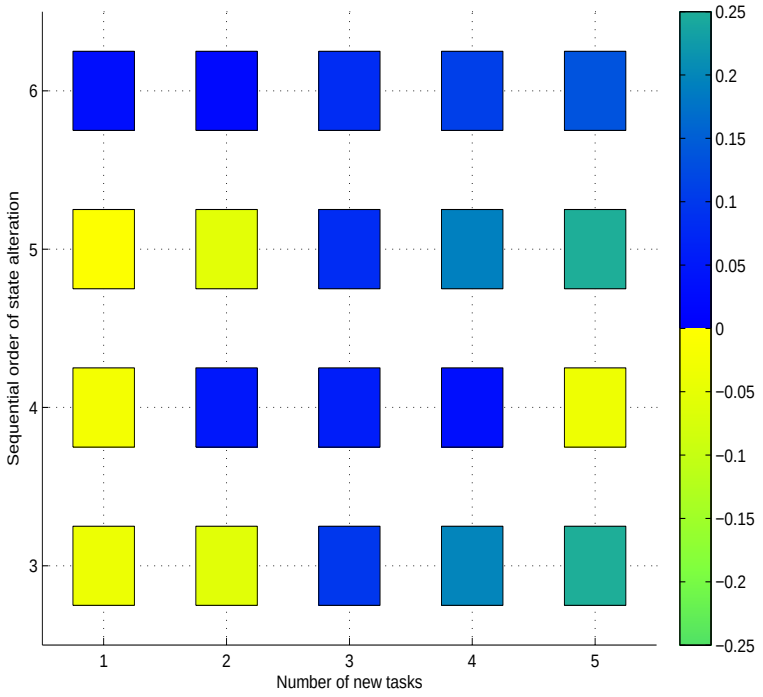



Figure 8.22: Dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{MedianBAR}, \tau, n, c\right)\right]$ under the changes in the number of tasks

Let us now explore, through an example, the procedure of how to determine $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\right.\right.$ $\left.\left.\text{McBAR}, T_b, \rho_1, n, c\right)\right]$, where $T_b \in \mathcal{W}$ (defined in Table 5.1). Note that this ARP$^o$T is based on the number of newly broken resources of type $R_1$ as the dynamical factor of interest, i.e. $\beta = \rho_1$. This procedure differs from the last enumerated procedure which is for determining $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \tau, n, c\right)\right]$, where the number of new tasks is the dynamical factor of interest. It differs from the last enumerated procedure on the following step numbers:

1) Set $n = 0$ and $c = 3$.

2) Set the factor $x_2$ in Equation 8.33 to $n$ while the factors $x_5$, $x_8$ and $x_{11}$ are set to zero. Notice that, based on expression 8.15, the factors $x_2$, $x_5$, $x_8$ and $x_{11}$ represent the numbers of newly broken resources of type $R_1$ at the third to the sixth SOSA respectively.

7) Repeat step 2 and the steps 3 to 6 of the last enumerated procedure with $c$ set to four,

five and then six to obtain $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \rho_1, n, 4\right)\right]$, $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \rho_1, n, 5\right)\right]$ and $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \rho_1, n, 6\right)\right]$ respectively. However, at step 2 of each repetition, the factors $x_2$, $x_5$ until $x_{3c-7}$ (refer to expression 8.18 for notation) are each set to $n$ while the factors $x_{3(c+1)-7}$, $x_{3(c+2)-7}$ until $x_{11}$ are each set to zero if $c$ is not equal to six.

8) Repeat step 7 with $n = 1$, $n = 2$, $n = 3$ and then $n = 4$.

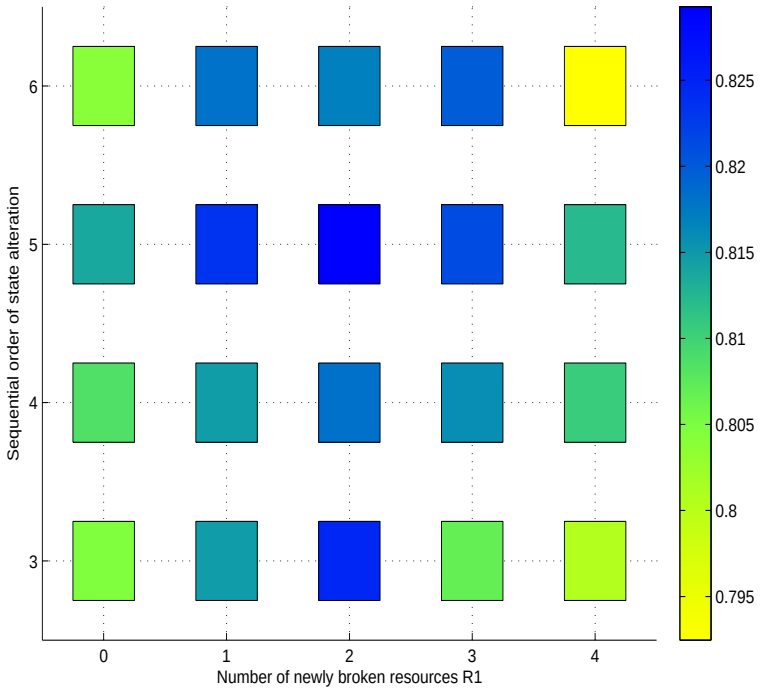



Figure 8.23: Dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{CBAM}, \rho_1, n, c\right)\right]$ under the changes in resource type $R_1$

Let us now investigate the dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \rho_1, n, c\right)\right]$ depicted in Figure 8.23. All ARP$^o$Ts (represented by the colour of the rectangles) in this figure are positive, which implies that the performance of McBAR is superior to that of CBAM at the third to the sixth SOSA and at $n = 0$ to four, when the dynamical factor of interest is the number of newly broken resources of type $R_1$. Similar observations and implications can be drawn from Figures 8.24, 8.25 and 8.26 that correspond to $T_b$ equal to RI, McBA and McBAS respectively. However, Figure 8.25 presents ARP$^o$Ts as being near zero while

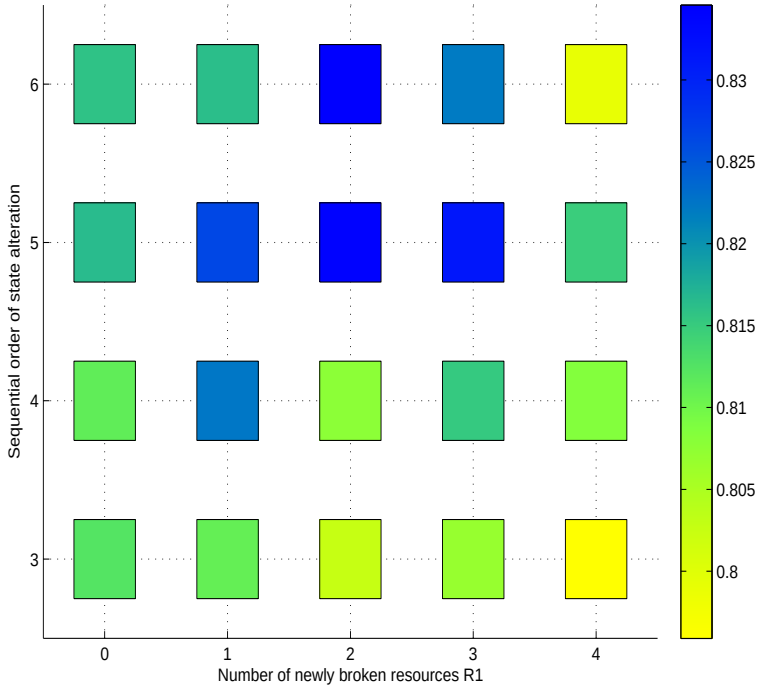



Figure 8.24: Dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{RI}, \rho_1, n, c\right)\right]$ under the changes in resource type $R_1$

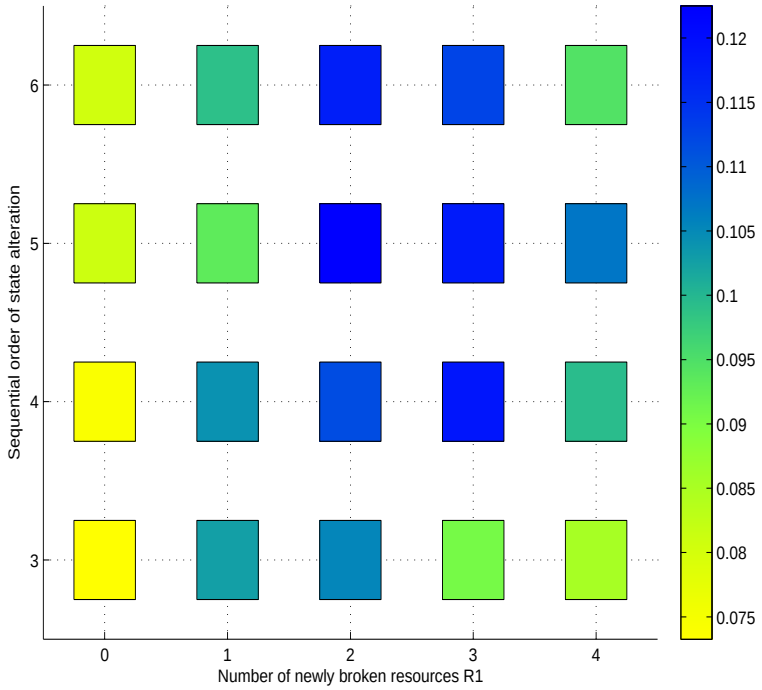



Figure 8.25: Dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{McBA}, \rho_1, n, c\right)\right]$ under the changes in resource type $R_1$

Figures 8.23, 8.24 and 8.26 depict ARP$^o$Ts much greater than zero. This difference implies that, although the performance of McBAR is superior to that of McBA, this superiority is not as great as to that of CBAM, RI and McBAS, when the dynamical factor of interest is the number of newly broken resources of type $R_1$, i.e. $\beta = \rho_1$.

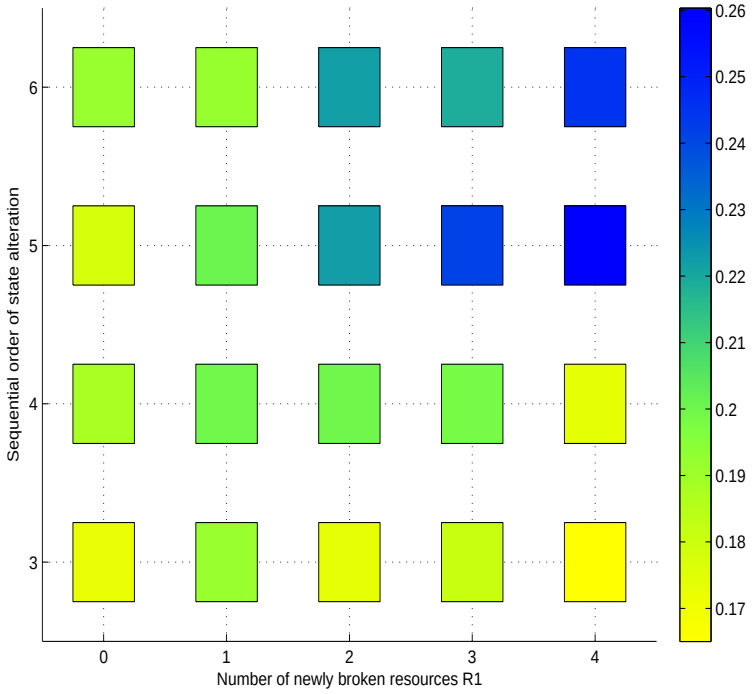



Figure 8.26: Dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{McBAS}, \rho_1, n, c\right)\right]$ under the changes in resource type $R_1$

In Figure 8.23, $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{CBAM}, \rho_1, n, c\right)\right]$ peaks around fifth SOSA and at $n = 2$. Similar dynamics can be observed in Figure 8.24 on $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{RI}, \rho_1, n, c\right)\right]$. Figure 8.25 illustrates the dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{McBA}, \rho_1, n, c\right)\right]$ as being generally increasing with SOSA $c$ and $n$. Figure 8.26 depicts the dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{McBAS}, \rho_1, n, c\right)\right]$ as being nearly similar to that of the last dynamics. $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{MedianBAR}, \rho_1, n, c\right)\right]$ depicted in Figure 8.27 varies in a narrow near-zero range. More points in this figure have positive values than negative. The dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{MedianBAR}, \rho_1, n, c\right)\right]$ is generally similar to that of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{MedianBAR}, \tau, n, c\right)\right]$ depicted in Figure 8.22. Notice that the former ARP$^o$T is a function of the number of newly broken resources of type $R_1$ while the

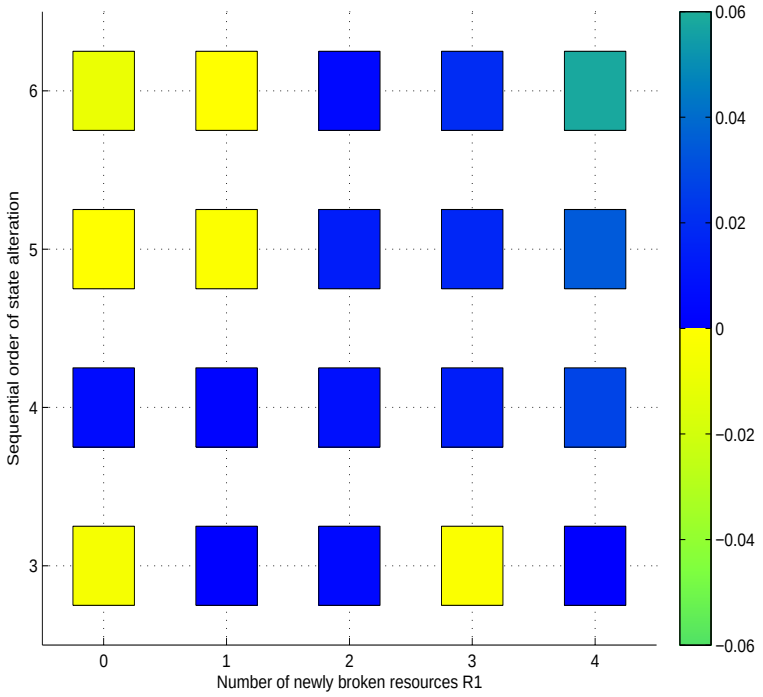



Figure 8.27: Dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, \text{MedianBAR}, \rho_1, n, c\right)\right]$ under the changes in resource type $R_1$

latter ARP$^{\text{o}}$T is a function of the number of new tasks.

The dynamics of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \rho_2, n, c\right)\right]$s are generally similar to those of $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \rho_1, n, c\right)\right]$s, but are not shown for brevity. This similarity is true for any $T_b$ from $\mathcal{W}$ (defined in Table 5.1) different to McBAR. Note that $\mathbf{E}_{s,t}^{3:6}\left[dSC\left(\text{McBAR}, T_b, \rho_2, n, c\right)\right]$ is ARP$^{\text{o}}$T with the number of newly broken resources of type $R_2$ (i.e. $\beta = \rho_2$) as the factor of interest.

## 8.7 Conclusions and Future Work

The research undertaken for this chapter showed that the composite models created through RSM and characterising some considered techniques from $\mathcal{T}$ including McBAR, are adequate, statistically significant, and have accurate prediction of the characteristics. Further, through the general method which utilised the composite models, McBAR was

proven to have the best performance among the considered techniques for solving problems from $\mathcal{O}^2$. Note however, that the performance of the unconsidered techniques from $\mathcal{T}$ is already proven in Chapter 6 to be inferior to that of McBAR through the limited method. Thus, although the first thesis' goal is partially fulfilled, it confidently be assumed to be fully fulfilled through the general method.

The research also demonstrated that the components of McBAR are legitimate through the general method, thereby achieving the fourth goal of the thesis. The performances of some considered techniques from $\mathcal{T}$, including McBAR, were also shown to vary with changes in the factors of the composite models considered above (factors such as the number of new tasks) that correspond to environmental-type parameters of the problems from $\mathcal{O}^2$.

The considered composite models are accurate only at small values of the declared environmental-type parameters. The future directions of the research in this chapter are to investigate models accurate at large values of the parameters and for predicting the characteristics of the techniques from $\mathcal{T}$ for solving problems with more than two objectives.

# Other Facets of the Techniques

# Chapter 9

This chapter investigates the limitation of McBAR measured in terms of resiliency (defined in Section 1.4.3). Section 9.1 presents the investigative approach which utilises the RSM-built models prepared in Chapter 8 to determine the resiliencies of techniques from $\mathcal{U} \subset \mathcal{T}$. The results of the application of this approach are presented and analysed in Section 9.2. Conclusion of and future work related to the results are discussed in Section 9.3.

This chapter also investigates the intelligence of each technique from $\mathcal{T}$, an intelligence that is an indirect measure of the efficiency of the technique for solving problems from $\mathcal{L}^2$. The definition of intelligence in Section 2.10.2 is revised in Section 9.4 to suit the system $\mathcal{S}$ (described in Section 5.6) used in the thesis. The revised definition is utilised in Section 9.5 to manifest the individual influences of a parameter and a category on the intelligence of each technique from $\mathcal{T}$. Conclusions and future work related to the results analysed in Section 9.5 are presented in Section 9.6.

## 9.1 Lagragian Optimisation

The performance of any technique $T_x \in \mathcal{U}$ (defined through Table 5.1 as the set {CBAM, McBA, McBAR, McBAS, MedianBAR}) for solving the problems from $\mathcal{O}^2$ could be affected by the changes of the MOE in which these problems are set. This section explores the degree related to these changes at which the performance of $T_x$ deteriorates to a level above that of a benchmark EA technique solving the same problem. As stated in Section 1.4.3, this degree is referred to as resiliency. It is assumed in the definition of resiliency

that some EA parameters of $T_x$ are fine tuned for $T_x$ to perform optimally under the environmental dynamics. Further, the benchmark EA technique follows no rules in creating its initial population.

The *Random Immigrants* (RI) technique defined in Section 5.9 is chosen as the benchmark EA technique. Thus resiliency is determined for each technique from $\mathcal{U}$ which does not contain RI. It is measured as,

$$R\left(\mathbf{x}_m\right) = \mathbf{x}_m \cdot \mathbf{x}_m, \tag{9.1}$$

where, referring to Equation 8.31 and to the definition of the factors in expression 8.15, the treatment/vector $\mathbf{x}_m$ contains factors which denote changes in the dynamic MOE described in Section 4.2; and $m$ is the label in Table 8.3 of the voxel (described in Section 8.3.4) in which $\mathbf{x}_m$ belongs.

The definition of resiliency entails the comparison of performances between $T_a \in \mathcal{U}$ and RI. The chosen comparator is $\mathrm{ARP^oS^bS^i}$ expressed as $\mathbf{E}_s^{3:6}\left[dSC_m\left(T_a, \mathrm{RI}, \mathbf{x}_m\right)\right]$ and defined in Section 8.3.6 with $dSC(T_a, \mathrm{RI})$ as the comparator between the performances of $T_a$ and RI. The definition of resiliency is mathematically expressed as,

$$\mathbf{E}_s^{3:6}\left[dSC_m\left(T_a, \mathrm{RI}, \mathbf{x}_m\right)\right] = \zeta, \tag{9.2}$$

where $\zeta > 0$ is a constant value; and $m$ is the label of the voxel over which this equation is satisfied. Based on the definition of performance in Section 2.5.2, this equation expresses the finding that, under the environmental changes encoded in $\mathbf{x}_m$, the performance of $T_a$ is better than that of RI by the degree $\zeta$ for solving the problem from $\mathcal{O}^2$ that corresponds to the treatment $\mathbf{x}_m$, where this performance is measured as the $\mathrm{ARP^oS^bS^i}$.

The resiliency of $T_a$ can be determined by finding the root $\mathbf{x}_m$ of Equation 9.2. Let us

begin the search for this root by applying Equation 9.2 to Equation 8.30 to obtain,

$$\zeta = \hat{\alpha}^{a,b,m} + \sum_{i=1}^{N_f} \hat{\beta}_i^{a,b,m} y_{i,m} + \sum_{i=1}^{N_f-1} y_{i,m} \sum_{j>i}^{N_f} \hat{\gamma}_{i,j}^{a,b,m} y_{j,m} + \sum_{i=1}^{N_f} \hat{\omega}_i^{a,b,m} y_{i,m}^2, \tag{9.3}$$

where, based on Equation 8.30, index $b$ corresponds to the fixed technique $T_b = \text{RI}$. By the fixed correspondence of index $b$, it will be dropped from the equation from here onwards. Equation 9.3 may have several roots (i.e. several $\mathbf{x}_m$s) in several voxels. The root with the least magnitude squared is the one selected as input to Equation 9.1 to determine the resiliency. As the components ($y_{j,m}$s in Equation 8.31) of this root are factors that denote environmental changes then the magnitude squared of this root corresponds to the least amount of environmental change at which the performance of $T_a$ is better than that of RI by the degree $\zeta$. Thus, the condition for the selection of the root is implemented as the minimisation of $R(\mathbf{x}_m)$,

$$\min_{\mathbf{x}_m} R(\mathbf{x}_m). \tag{9.4}$$

where $R(\mathbf{x}_m)$ is the magnitude square of $\mathbf{x}_m$, based on Equation 9.1.

The popular method called *Lagragian Optimization* [91] is applied to obtain a definite solution to the optimisation problem where $R(\mathbf{x}_m)$ is the function to be optimised/minimised; and,

$$g(\mathbf{x}_m) = \zeta - \mathbf{E}_s^{3:6}\left[dSC_m(T_a, \text{RI}, \mathbf{x}_m)\right] = 0 \tag{9.5}$$

is the constraint. In Lagragian Optimisation, a constant called *Lagragian Multiplier* is determined. It is denoted as $\lambda^{a,m}$ and satisfies,

$$\nabla\left(R(\mathbf{x}_m) + \lambda^{a,m} g(\mathbf{x}_m)\right) = \mathbf{0}, \tag{9.6}$$

where $\mathbf{0}$ is a zero vector and,

$$\nabla = \left[\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \ldots, \frac{\partial}{\partial x_{N_f}}\right]. \tag{9.7}$$

Using Equations 8.30, 9.1 and 9.5 we obtain the root,

$$\mathbf{x}^* \left(\lambda^{a,m}\right) = \lambda^{a,m} \left[2I - \lambda^{a,m}\Gamma^{a,m}\right]^{-1} \beta^{a,m}, \tag{9.8}$$

where $I$ is the identity matrix,

$$\Gamma^{a,m} = \begin{bmatrix} 2\hat{\omega}_1^{a,m} & \hat{\gamma}_{1,2}^{a,m} & & \cdots & \hat{\gamma}_{1,k}^{a,m} \\ \hat{\gamma}_{1,2}^{a,m} & 2\hat{\omega}_2^{a,m} & \hat{\gamma}_{2,3}^{a,m} & \cdots & \hat{\gamma}_{2,k}^{a,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \hat{\gamma}_{1,k}^{a,m} & \cdots & & \hat{\gamma}_{k-1,k}^{a,m} & 2\hat{\omega}_k^{a,m} \end{bmatrix} \tag{9.9}$$

and the column vector,

$$\beta^{a,m} = \left[\hat{\beta}_1^{a,m}, \hat{\beta}_2^{a,m}, \ldots, \hat{\beta}_k^{a,m}\right], \tag{9.10}$$

where $\hat{\omega}_i^{a,m}$, $\hat{\gamma}_{i,j}^{a,m}$, and $\hat{\beta}_i^{a,m}$ are equal to $\hat{\omega}_i^{a,b,m}$, $\hat{\gamma}_{i,j}^{a,b,m}$, and $\hat{\beta}_i^{a,b,m}$ found in Equation 8.30 respectively; and $1 \leq i, j \leq Nf$. Note that the root $\mathbf{x}^*$ in Equation 9.8 is a function of Lagragian Multiplier $\lambda^{a,m}$. Substituting Equation 9.8 to Equation 9.5 yields,

$$g\left(\lambda^{a,m}\right) = \zeta - \mathbf{E}_s^{3:6}\left[dSC_m(T_a, RI, \mathbf{x}^*\left(\lambda^{a,m}\right))\right] = 0. \tag{9.11}$$

Equation 9.11 may have several roots. Let,

$$R^{a,m} = \left\{\lambda_1^{a,m}, \lambda_2^{a,m}, \ldots, \lambda_{N_{a,m}(m)}^{a,m}\right\} \tag{9.12}$$

be a set comprised of the roots of Equation 9.11 numbering $N_{a,m}(m)$ which is a function of voxel label $m$. For a given $\lambda_k^{a,m} \in R^{a,m}$, the necessary conditions for its corresponding root $\mathbf{x}^*\left(\lambda_k^{a,m}\right)$ (in Equation 9.8) to be a possible solution to the optimisation problem are:

1. $\lambda_k^{a,m} > 0$ [91]
2. $\mathbf{x}^*\left(\lambda_k^{a,m}\right)$ must satisfy Equation 9.6 [91]
3. $\mathbf{x}^*\left(\lambda_k^{a,m}\right)$ must be found inside the voxel $m$.

4. Using the notations in Equation 8.16 and expression 8.15, the $x_{3\times3-8}$, $x_{3\times4-8}$, $x_{3\times5-8}$ and $x_{3\times6-8}$ components of the solution $\mathbf{x}^*\left(\lambda_k^{a,m}\right)$ must not simultaneously be zero; components that denote the number of new tasks at the third to the sixth SOSA. This prohibition conforms to the restriction emphasised in Section 8.3.2 not to consider irrelevant treatments.

Among all the solutions (that correspond to various $\lambda_k^{a,m}$s) which satisfy the conditions, the solution $\mathbf{x}^*\left(\lambda_{\kappa(m)}^{a,m}\right)$ (dependent on voxel label $m$) which has the least magnitude is the only solution considered for further analysis. Thus, each voxel can have at most one considered solution. And among the considered solutions in all voxels, the solution $\mathbf{x}^*\left(\lambda_{\kappa(m)}^{a,\mu}\right)$ which has the least magnitude is considered as the solution to the optimisation problem. Note that Equation 9.6 can have no root that satisfies the conditions for a given $\zeta$ in Equation 9.5.

## 9.2 Resiliency

This section explores the results of applying the procedures laid out in the last section to determine the resiliency of each technique from $\mathcal{U}$ (defined through Table 5.1). These results are presented in Table 9.1. The first column of this table contains the techniques from $\mathcal{U}$. The threshold $\zeta$ in Equation 9.2 is at the second column. The components (factor values) of the solutions/treatments (e.g. $\langle 1,1,0,0,0,0,0,0,0,0,0,0\rangle$) to the optimisation problems defined by the relationship 9.4 and Equation 9.5 are found under the column heading $x_i$, where $1 \leq i \leq 12$. These $x_i$-labelled columns are grouped according to the SOSA – indicated at the topmost row – to which they correspond, based on expression 8.15. Using the solution to the optimisation problem, the resiliency (e.g. 2.0) of a given technique (e.g. CBAM) is determined through Equation 9.1 and found under the column heading $R$. Note that a given technique at the first column has several pairs of thresholds at the second column and at the same row as this technique. The resiliency of this technique is determined for each of these thresholds. The average of the determined

resiliencies (e.g. 13.429) is found under the column heading $\mathbf{E}[R]$. The set of information in the same row as that of a technique name is referred to as data group. In each data group, the value at the row labelled $\mathbf{E}[x_i]$ (e.g. $\mathbf{E}[x_1] = 1.6$) and under the column $x_i$ is the average of the $i^{th}$ component of the solutions in the data group. The value under the column heading $\mathbf{E}[dT]$ is the average (e.g. 0.964) of $\mathbf{E}[x_i]$s over the factors $x_1$, $x_4$, $x_7$ and $x_{10}$, i.e. factors that represent the number of new tasks based on expression 8.15. The value under the column heading $\mathbf{E}[dR_1]$ ($\mathbf{E}[dR_2]$) is the average of $\mathbf{E}[x_i]$s over factors that represent the number of newly broken resources of type $R_1$ ($R_2$).

Table 9.1: Resiliency of techniques at various thresholds

| SOSA | | 3 | | | 4 | | | 5 | | | 6 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Factors | | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ | $x_{12}$ | | | | | |
| Technique | threshold $\zeta$ | | | | | | | | | | | | | $R$ | $\mathbf{E}[R]$ | $\mathbf{E}[dT]$ | $\mathbf{E}[dR_1]$ | $\mathbf{E}[dR_2]$ |
| CBAM | 0.175 | 3 | 1 | 2 | 1 | 3 | 1 | 1 | 1 | 1 | 2 | 3 | 2 | 45 | 13.429 | 0.964 | 0.4 | 0.4 |
| | 0.25 | 1 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 3 | 1 | 1 | 24 | | | | |
| | 0.3 | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 12 | | | | |
| | 0.325 | 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 6 | | | | |
| | 0.375 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 3 | | | | |
| | 0.4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | | | | |
| | 0.425 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | | | | |
| | $\mathbf{E}[x_i]$ | 1.6 | 0.6 | 0.4 | 0.9 | 0.4 | 0.1 | 0.6 | 0.1 | 0.6 | 0.9 | 0.6 | 0.4 | | | | | |
| McBA | 0.775 | 3 | 1 | 2 | 0 | 2 | 1 | 3 | 2 | 0 | 1 | 3 | 3 | 51 | 29.5 | 0.96 | 1.25 | 1.125 |
| | 0.825 | 0 | 1 | 0 | 1 | 0 | 2 | 0 | 1 | 1 | 0 | 0 | 0 | 8 | | | | |
| | $\mathbf{E}[x_i]$ | 1.5 | 1 | 1 | 0.5 | 1 | 1.5 | 1.5 | 1.5 | 0.5 | 0.5 | 1.5 | 1.5 | | | | | |
| McBAR | 0.775 | 2 | 1 | 3 | 0 | 4 | 3 | 2 | 2 | 4 | 2 | 0 | 3 | 76 | 55.5 | 1.25 | 1.25 | 2.6 |
| | 0.825 | 3 | 1 | 1 | 0 | 1 | 4 | 0 | 0 | 1 | 1 | 1 | 2 | 35 | | | | |
| | $\mathbf{E}[x_i]$ | 2.5 | 1.0 | 2.0 | 0.0 | 2.5 | 3.5 | 1.0 | 1.0 | 2.5 | 1.5 | 0.5 | 2.5 | | | | | |
| MedianBAR | 0.775 | 2 | 0 | 1 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 14 | 13.5 | 1.25 | 0.25 | 0.4 |
| | 0.825 | 3 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 13 | | | | |
| | $\mathbf{E}[x_i]$ | 2.5 | 0.0 | 0.5 | 1.0 | 0.0 | 0.5 | 1.0 | 0.0 | 0.0 | 0.5 | 1.0 | 0.5 | | | | | |
| McBAS | 0.775 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 11 | 10.5 | 1.00 | 0.13 | 0.5 |
| | 0.825 | 1 | 0 | 1 | 2 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 10 | | | | |
| | $\mathbf{E}[x_i]$ | 2.0 | 0.0 | 0.5 | 1.0 | 0.0 | 0.0 | 0.0 | 0.5 | 1.0 | 1.0 | 0.0 | 0.5 | | | | | |

As pointed out in Section 9.1, for a given technique and a given threshold, the solution to the optimisation problem defined by the relationships 9.4 and Equation 9.5 may not exist. With CBAM as the given technique, solutions to the optimisation problem exist at several thresholds. However, only the solutions at selected thresholds $\zeta$ in the range $0.175 \leq \zeta \leq 0.425$ are presented in Table 9.1. In any map-based technique (defined in

Section 6.3 as McBA, McBAR, McBAS or MedianBAR), the solutions to the optimisation problem are found to exist at two selected thresholds in a range that differs from that of other map-based techniques. These two selected thresholds are 0.775 and 0.825 and are common across the different ranges that correspond to all of the map-based techniques. This commonality is devised to facilitate the comparison of resiliencies of the map-based techniques. For each technique from $\mathcal{U}$, no solution to the optimisation problem is found to exist at several selected thresholds outside the range that corresponds to this technique. Thus, the authors believe that solutions at the thresholds outside this range are unlikely to exist.

Consider the data group in Table 9.1 at the same row as CBAM. The second to the bottom row of this group presents a resiliency of 2.0 and a threshold of 0.425. Based on the definition of resiliency, this data implies that changes in the MOE of at least 2.0 units will cause a deterioration of the performance of CBAM to 0.425 above that of RI. This MOE is the one in which the problem from $\mathcal{O}^2$ is set, the problem that corresponds to the treatment (at the second to the bottom row of this data group) which encodes the changes. Note that in the other rows in the data group, except for the bottom row, if the degree of changes is greater than 2.0 units the performance of CBAM is still superior to that of RI but this superiority is less than 0.425. The average of the resiliencies in the data group is 13.429. Not tabulated, the average of the thresholds in the data group is 0.3214. The data group also demonstrates that a 0.964 average $\mathbf{E}[dT]$ number of new tasks, a 0.4 average $\mathbf{E}[R_1]$ number of newly broken resources of type $R_1$ and a 0.4 average $\mathbf{E}[R_2]$ number of newly broken resources of type $R_2$ in the MOE cause the deterioration of the average performance of CBAM to 0.3214 above that of RI; the average is over the thresholds in the data group.

Based on Table 9.1 and among the map-based techniques, McBAR has the highest average resiliency over the domain of selected thresholds, 0.775 and 0.825, which averages to 0.8. In other words, on average over this domain, McBAR can withstand the largest degree of changes in the MOE before its performance will deteriorate to 0.8 above that

of RI. Further, McBAR can withstand the highest average $\mathbf{E}[dT]$ increase in number of tasks, the highest average $\mathbf{E}[R_1]$ number of newly broken resources of type $R_1$, and the highest average $\mathbf{E}[R_2]$ number of newly broken resources of type $R_2$ before its average performance will deteriorate to 0.8 above that of RI. The averages $\mathbf{E}[dT]$, $\mathbf{E}[R_1]$ and $\mathbf{E}[R_2]$ are over the domain of selected thresholds and are 1.25, 1.25 and 2.6 respectively.

MedianBAR can also withstand the highest average $\mathbf{E}[dT]$ increase in the number of tasks; and McBA can also withstand the highest average $\mathbf{E}[R_1]$ number of newly broken resources of type $R_1$ before their performances will deteriorate in the manner just described. The averages $\mathbf{E}[dT]$ and $\mathbf{E}[R_1]$ are both 1.25. The techniques arranged in descending order of resiliency are: McBAR, McBA, MedianBAR and McBAS.

Each data group in Table 9.1 expresses the finding that the lesser the threshold in this group, the larger the resiliency that corresponds to this threshold. In the data group in the same row as McBAR, the threshold of 0.775 corresponds to the resiliency of 76. Based on the expression, if the resiliency of McBAR is determined at each of the thresholds (all less than 0.775) in the data group in the same row as CBAM it is expected to be greater than 76. Thus the average of these expected resiliencies could be greater than the average (13.429) of the resiliencies in the data group in the same row as CBAM. Therefore, on the average, McBAR is expected to be more resilient than CBAM. This conclusion is appropriate since the expected resiliencies correspond to the thresholds that are identical to those in the data group in the same row as CBAM.

The expected resiliencies of McBAR correspond to the non-existent solutions to the optimisation problems (defined by the relationship 9.4 and Equation 9.5) with thresholds in the same row as CBAM. This could be explained as follows: based on the expression considered in the last paragraph, a solution $\mathbf{s}$ to the optimisation problem with given small threshold (e.g. 0.175) could yield a large resiliency (e.g. 731) of McBAR. Further, based on Equation 9.1, $\mathbf{s}$ can have component factors of large values (e.g. $x_1 = 8$). Thus, based on Section 9.1, $\mathbf{s}$ can be determined using a final dynamical sub-model (described in Section 8.3.6) valid in a voxel that contains this solution (treatment based on Section

8.3.6). However, all the voxels listed in Table 8.3 contain treatments with factors of small values (e.g. $x_1 < 8$) due to the restriction in expression 8.18. Thus there is no voxel in the table that contains $\mathbf{s}$. Therefore, the final dynamical sub-model does not exist. Consequently, $\mathbf{s}$ cannot be found through the process presented in Section 9.1.

## 9.3 Conclusions on Resiliency

The research performed above proves McBAR to be the most resilient – among all the techniques from $\mathcal{U} \subset \mathcal{T}$ – under changes in the MOEs in which the problems from $\mathcal{O}^2$ are set. Further, the research showed McBAR and other mapped-based techniques (e.g. McBA, McBAR, McBAS and MedianBAR) from $\mathcal{U}$ to be more resilient than CBAM. Based on the results presented in Section 6.2, the performance of CBAM is manifested through the limited method to be inferior to each of those of the mapped-based techniques and superior to other non-mapped-based techniques from $\mathcal{T}$. Thus we confidently claim that all of the mapped-based techniques are more resilient than all the other techniques from $\mathcal{T}$. Considering that McBAR is more resilient than any other mapped-based techniques, we claim that McBAR is the most resilient among all techniques from $\mathcal{T}$. We therefore achieved the third goal of the thesis in this chapter.

The composite models utilised to determine the resiliencies of the techniques from $\mathcal{U}$ are accurate only – as presented in Section 8.3.2 – at small values of their factors which are environmental-type parameters. A future direction of the above research would be to investigate the resiliencies of the techniques using composite models accurate at large values of their factors.

## 9.4 Intelligence of Techniques

Let us now investigate the intelligence of the techniques from $\mathcal{T}$ as defined in Section Section 2.10.2. For this purpose:

1. The Kolmogorov complexity $K(\rho)$ in Equation 2.30 is redefined as the number of CPU cycles in system $\mathcal{S}$ spent by a technique (viewed as an agent) $\tau$ from $\mathcal{T}$ to solve a sub-problem $\rho$ (e.g. $p_3^2$) from a set $P_x$ of sub-problems. Note that, in relation to Reinforcement Learning, the solution process is considered as a series of actions/operations by $\tau$, based on the terminology in Section 2.10.2.

2. The reward for the search of the obtained solution is measured as the hypervolume of the solutions. The reference point (defined in Section 2.5.1) used to compute the reward/hypervolume is the maximum objective vector of all solutions to all simulations of all sub-problems of all problems from $\mathcal{L}^2$.

3. The summation in Equation 2.30 is made to take into account all sub-problems from $P_x$.

4. The Kolmogorov complexity and the reward are denoted as a function of the simulation index $s$ to account for the simulations (explained in Section 4.4) of these sub-problems.

5. The average $\mathbf{E}_s[\bullet]$ of the only summation term in Equation 2.30 is taken over the simulations.

Consequently, the intelligence of the technique $\tau$ in solving all sub-problems from $P_x$ is defined as,

$$\Psi(\tau, x) = \frac{1}{|P_x|} \sum_{\rho \in P_x} \mathbf{E}_s \left[ 2^{-\alpha K(\tau,\rho,s)} R^{\tau}_{\rho,s} \right] \tag{9.13}$$

where its variation is with respect to the parameters or categories $x$ of the sub-problems from $P_x$; and $\alpha$ is a scaling factor of the Kolmogorov complexity and set equal to $10^{-10}$ to cope with numerical limitations of the computing machine in system $\mathcal{S}$. The additional parameter $\tau$ in the Kolmogorov complexity $K(\tau, \rho, s)$ indicates which technique corresponds to this complexity.

The subscript $x$ of the set $P_x$ is dropped if it is composed of all sub-problems of all problems from $\mathcal{L}^2$. Its subscript is $\delta = d$ when it is composed of all sub-problems $\rho^2_{i,s}$

of some problems from $\mathcal{L}^2$, where $i$ is the SOSA; and each of these problems has task duration changes that abide by Equation A.9 with the given value $d$ of $\delta$. The set $P_{\delta=d}$ is used to examine the intelligence variation of a technique from $\mathcal{T}$ under the different values of $\delta$. The subscript of $P_x$ is $\gamma = c$ when it is composed of some sub-problems $\rho^2_{i,s}$ of all problems from $\mathcal{L}^2$, where each of these sub-problems is set in the $i^{th}$ snapshot taken immediately after a change of type $c$ in Table 4.2. The set $P_{\gamma=c}$ is used to examine the intelligence variation of a technique from $\mathcal{T}$ under the different change types $c$.

## 9.5 Measured Intelligence

The intelligence of each technique from $\mathcal{T}$ in solving all sub-problems from each of the sets $P_{\delta=3.0}$, $P_{\delta=6.0}$ and $P$ is found, respectively, in the second to the fourth row of Table 9.2. The fourth row reveals McBAR as being the most intelligent amongst all techniques from $\mathcal{T}$ in solving any problem from $\mathcal{L}^2$. The second and third rows show that the last result is true, independent of the two values of $\delta$, 3.0 and 6.0. The techniques arranged in descending order of intelligence listed in Table 9.2 are: McBAR, McBAS, MedianBAR, McBA, CBAM, NDLPOP, GIBAR, RI and EDA/$\mathcal{P}^2$. This order is closely related to that of the computing time illustrated in Figure 6.2 and to the performance of the techniques described in Section 6.2. Table 9.2 shows McBAR and McBAS, which apply a statistical mean to determine centroids as explained in Section 5.1, to be more intelligent than MedianBAR which uses the statistical median instead of the mean. This is despite that, based on Table 6.1, the performance of MedianBAR is as good as that of McBAR and better than that of McBAS for solving sub-problems from $\mathcal{L}^2$. However, based on Equation 9.13, the longer time needed to compute the median rather than the mean could cause MedianBAR to be inferiority in intelligence than either McBAR or McBAS. Based on Equation 9.13, the chosen value of the scaling factor could also influence the inferiority of the intelligence of MedianBAR.

The intelligence of each technique from $\mathcal{T}$ in solving sub-problems from $P_{\gamma=0}$, $P_{\gamma=1}$,

Table 9.2: Average Universal Intelligence with respect to $\delta$

| δ | McBAR | McBAS | MedianBAR | McBA |
|---|---|---|---|---|
| 3.0 | 20,210 | 17,904 | 16,649 | 14,791 |
| 6.0 | 19,817 | 17,919 | 16,505 | 14,406 |
| Average | 20,013 | 17,912 | 16,577 | 14,599 |

| δ | CBAM | NDLPOP | GIBAR | RI | EDA/$\mathcal{P}^2$ |
|---|---|---|---|---|---|
| 3.0 | 12,284 | 11,844 | 10,624 | 10,476 | 9,101 |
| 6.0 | 11,979 | 11,439 | 10,205 | 10,076 | 8,760 |
| Average | 12,131 | 11,641 | 10,414 | 10,276 | 8,930 |

..., $P_{\gamma=6}$ is found from the second to the eighth row of Table 9.3 respectively. All of these rows show McBAR to be the most intelligent technique amongst all techniques from $\mathcal{T}$ independent of the type $\gamma$ of change – listed in Table 4.2 – that occurs in the MOE in which the sub-problems are set. The techniques arranged in descending order of intelligence listed in Table 9.3 are similar to those found in the last paragraph.

Table 9.3: Average Universal Intelligence with respect to change type

| Change Type | McBAR | McBAS | MedianBAR | McBA |
|---|---|---|---|---|
| 0 | 30,117 | 27,377 | 25,439 | 22,693 |
| 1 | 12,462 | 10,666 | 9,777 | 8,448 |
| 2 | 15,334 | 13,192 | 12,360 | 9,762 |
| 3 | 4,706 | 3,920 | 3,484 | 2,931 |
| 4 | 12,011 | 10,149 | 9,248 | 8,076 |
| 5 | 5,406 | 4,518 | 3,950 | 3,778 |
| 6 | 9,542 | 8,231 | 7,469 | 6,032 |

| Change Type | CBAM | NDLPOP | GIBAR | RI | EDA/$\mathcal{P}^2$ |
|---|---|---|---|---|---|
| 0 | 19,262 | 18,477 | 16,423 | 16,198 | 14,354 |
| 1 | 6,613 | 5,452 | 5,374 | 6,041 | 4,764 |
| 2 | 7,564 | 7,179 | 6,326 | 7,637 | 5,672 |
| 3 | 2,073 | 2,365 | 1,733 | 2,118 | 1,363 |
| 4 | 6,394 | 6,848 | 5,304 | 5,735 | 4,318 |
| 5 | 2,795 | 2,736 | 2,325 | 2,559 | 1,694 |
| 6 | 4,561 | 4,253 | 3,714 | 3,696 | 3,227 |

All of these results established McBAR to be the most intelligent technique amongst all of the techniques from $\mathcal{T}$ despite utilising a simple mapping function $\mathcal{F}$ to fundamentally alleviate the performance of GIBAR.

## 9.6 Conclusions on Technique's Intelligence

The research conducted for Sections 9.4 and 9.5 proves McBAR to be the most intelligent among the techniques from $\mathfrak{T}$ in solving the problems from $\mathfrak{L}^2$. As explained in Section 1.4.5, this intelligence defines the efficiency of the techniques in solving the problems. Thus, McBAR is the most efficient among the techniques. Thereby, the fifth goal of the thesis is achieved in this chapter.

The last conclusion is true at any of the two values of $\delta$ used above as the parameter of Equation A.9 that models changes in task durations of the problems from $\mathfrak{L}^2$; at any type of simultaneous changes in the MOEs in which the problems are set, types that are listed in Table 4.2; and at any form of PNT that constrained the problems.

The problems from $\mathfrak{L}^2$ are the ones utilised to determine the intelligence of the techniques from $\mathfrak{T}$. A future direction for the research will be to investigate such intelligence for solving problems from $\mathfrak{Q}^2$ whereby this intelligence is being modelled in a manner explained in Chapter 8.

# Conclusions

## Chapter 10

The research exhibited in this thesis has tackled the demand for an effective and efficient algorithm to solve many problems from $\mathcal{M}$ that is a set of multi-objective dynamic resource-constrained project scheduling (RCPS) problems, each with time-varying number of tasks. The thesis concludes by recalling the hypothesis and the thesis goals in Section 10.1; the processes used and the results revealed in attaining these goals in Section 10.2A; and the summary of the deficiencies of the processes and some future objectives of the research in Section 10.3.

## 10.1 Recalled Hypothesis and Goals

As stated in the Chapter 1, it was hypothesised that "the evolutionary and memory-based algorithm referred to as Mapping of Task IDs for Centroid-Based Adaptation with Random Immigrant (McBAR) is an effective and efficient technique to solve several problems from $\mathcal{M}$". This hypothesis was supported by the achievement of the five following thesis' goals:

1. To demonstrate the effectiveness of McBAR for solving problems from $\mathcal{L}^2 \cup \mathcal{O}^2 \subset \mathcal{M}$, with each having two conflicting objectives to minimise – cost and duration of schedules.

2. To legitimise the sub-algorithms of McBAR.

3. To manifest the core algorithm of McBAR as being significant even when applied

to solve problems from $\mathcal{L}^3 \subset \mathcal{M}$, with each having three objectives to minimise – the last two problem objectives and the objective to minimise the probability of a schedule to become infeasible due to the dynamics of the environment in which this schedule is executed.

4. To exhibit the limitation of McBAR in solving some significant problems from $\mathcal{O}^2 \subset \mathcal{M}$.

5. To present the efficiency of McBAR in solving problems from $\mathcal{L}^2 \subset \mathcal{M}$.

## 10.2 Summary of Methodology and Results

A summary of the research procedures and results that led to the achievement of the thesis' goals is now presented. The first goal was achieved initially by undertaking research on the deficiencies of the current techniques in the literature applied to solve scheduling problems, such as the technique referred to as Centroid Based-Adaptation with Random Immigrant (CBAR). CBAR was applied in our previous work [40] to solve multi-objective dynamic RCPS problems with a fixed number of tasks. We found CBAR to be inapplicable to solve any problem from $\mathcal{M}$, i.e. a problem with time-varying number of tasks. This inapplicability was remedied by innovating CBAR. However, despite the remedy, the performance (solution searching ability) of the innovated CBAR for solving problems from $\mathcal{M}$ started to degrade when new tasks appeared, tasks which were absent before any change occurred in the environments in which the problems were set. This degradation was neutralised by the several designed algorithms added to CBAR; thereby CBAR was renamed McBAR.

The second step for achieving the first thesis' goal was the creation of several dynamic environments in which the performance of McBAR – for solving the dynamic problems from $\mathcal{L}^2 \cup \mathcal{L}^3 \cup \mathcal{O}^2 \subset \mathcal{M}$ set in these environments – was manifested. Each of the dynamic environments was regarded as a sequence of static environments, each being a snapshot of the dynamic environment taken right after each change to this dynamic

environment. Each of the dynamic problems was regarded as a sequence of sub-problems set in the similarly ordered snapshots of the dynamic environment in which it was set.

The third step in achieving the first thesis' goal was as follows. The effectiveness of McBAR was demonstrated in two ways – the limited and the general methods. In the limited method, each technique from $\mathcal{T}$ (a set of techniques investigated in the thesis) was applied to solve sub-problems of dynamic problems from $\mathcal{L}^2 \subset \mathcal{M}$, each dynamic problem being set in a dynamic environment that had a small or large degree of change. This application showed that, among all the techniques from $\mathcal{T}$, McBAR has generally the best performance for solving such sub-problems, i.e. it is effective. The performance of McBAR was found to be best for solving the sub-problems set in snapshots taken right after the first occurrence of new tasks in the dynamic environments.

In the general method, McBAR was applied to solve dynamic problems whose parametric values and/or categorical types were designed using Design of Experiment [141]. The obtained solutions were utilised to build an empirical model, following Response Surface Methodology [141]. Such a model predicts the performance of McBAR when McBAR solves the sub-problems of dynamic problems from $\mathcal{O}^2 \subset \mathcal{M}$, where these dynamic problems are set in dynamic environments with a smaller degrees of changes than those in the dynamic environments in which the dynamic problems from $\mathcal{L}^2$ are set. A similar procedure was applied to build an empirical performance model of some techniques from $\mathcal{T}$ other than McBAR. Through all these empirical models, McBAR was manifested to perform best for solving dynamic problems from $\mathcal{O}^2$.

The second thesis' goal was attained by extending the core algorithm of McBAR to solve problems from $\mathcal{L}^3 \subset \mathcal{M}$. The generated solutions/schedules proved, despite the extension, the effectiveness of the core algorithm based on their properties such as financial worth, execution costs and durations.

The third thesis' goal was achieved firstly by the development of an equation that models the resiliency of some techniques from $\mathcal{T}$. Here resiliency is a measure of the limitation of the techniques. It is defined as the degree of change in an environment

whereby the performance of a technique from $\mathcal{T}$ will deteriorate to a given level above the performance of a benchmark technique for solving the same problems. The performance is for solving problems from $\mathcal{O}^2$ set in the environment. Secondly, Lagrange Optimisation [91] was applied to the above-mentioned empirical models to determine the resiliency of some techniques from $\mathcal{T}$, whereby McBAR was found to be the most resilient.

The fourth thesis' goal was attained by the independent application of the limited and the general methods. In the application of the limited method, each technique from $\mathcal{T}$ was applied to solve problems from $\mathcal{L}^2$. The generated solutions were utilised to support the legitimacy of the sub-algorithms of McBAR. Indeed, several sub-algorithms of McBAR were found to be legitimate. In the application of the general method, the above-declared empirical models were successfully used to prove the legitimacy of the sub-algorithms over the problems from $\mathcal{O}^2$.

The fifth thesis' goal was realised firstly by defining the efficiency of each technique from $\mathcal{T}$ through a researched measure of algorithm intelligence. Second, this measure was applied to the solutions obtained by each technique for solving problems from $\mathcal{L}^2$. The research had proven McBAR to be the most intelligent (i.e. most efficient) technique among all the techniques from $\mathcal{T}$.

A conclusion, covered in and related to the thesis but beyond the original objectives above, points to the mapping function $\mathcal{F}$ as being the possible root cause of the superiority of techniques considered in this thesis whose core algorithm includes this function, the superiority for solving several dynamic and complex problems from $\mathcal{M}$. As explained in Section 5.5, this function maps an ID of a given task to a value close to already mapped IDs of tasks with task precedence order (explained in Section 5.5) equal to that of the given task, where task precedence order is the maximum number of task-representing nodes traversed from a considered task to the start of a precedence network of tasks. The function is an identity for the first ID it mapped. Further, the IDs in genes of a genotype are sequentially mapped by the function from the ID in the first to the last gene of the genotype. As explained in Section 5.4, the effect of this function is to transform

the search space associated to a past problem from $\mathcal{M}$ in order to reduce the number of clusters in this search space, clusters of the genotypes of non-dominated solutions to the problem. The representative of the clusters, after being reduced in number, was explained in Section 5.4 as possibly being beneficial in determining solutions to a current problem from $\mathcal{M}$ that is closely related to the past problem. The characteristics of this representative are influenced by the mapping function, such that this function can possibly be the root cause of the referred superiority. Sections 6.3 and 8.5 prove that the function as a legitimate sub-algorithm of McBAR and that the techniques which contain this function perform (performance for solving several problems from $\mathcal{M}$) better than those considered in this thesis that did not.

In conclusion, our hypothesis was successfully supported, in that the effectiveness and efficiency of McBAR for solving many problems from $\mathcal{M}$ was established.

## 10.3 Future Work

The future direction for research arising from this thesis is to extend the above-emphasised superiority of McBAR in effectiveness (for solving problems from $\mathcal{L}^2$ and $\mathcal{O}^2$), efficiency (for solving problems from $\mathcal{L}^2$) and resiliency (for solving problems from $\mathcal{O}^2$) over all problems from $\mathcal{M}$; over various EA approaches, such as Genetic Algorithm, Differential Evolution, and Augmented Simulated Annealing; and over diverse MOEAs such as, SPEA2 [238], PDE [2], and VEGA [179]. McBAR needs to be proven as effective for solving problems from $\mathcal{M}$ with four conflicting objectives – the three above-noted objectives and the objective to minimise the difference between the starting times of the same unfinished task in two schedules which are solutions to sub-problems set in the current and last snapshots. The last objective is important in order to minimise rescheduling tasks due to changes in the environment where the schedules are implemented. The effectiveness (for solving problems from $\mathcal{L}^3$) of McBAR's core algorithm and the legitimacy (manifested in solving problems from $\mathcal{L}^2$ and $\mathcal{O}^2$) of McBAR's component need to

be investigated over four or more conflicting objectives as well. It is hoped that future researches will be directed toward the areas suggested.

# Appendix

## A.1 Mathematical Formulation of the Static Sub-Problems from $\Gamma_i$

The partially described sub-problems of problems from $\mathcal{L}^2 \cup \mathcal{L}^3$ and $\mathcal{O}^2$ in Sections 4.2 and 4.3, respectively, will now formally be defined. Note that the sub-problems are elements of $\Gamma_i$, where index $i$ is the SOSA of MOEs in which the problems are set; and has the range, $0 \leq i \leq L$, with given $L$.

1. **Inputs:**

   (a) A set $T$ of non-pre-emptive tasks:

   $$T = \{T_0, T_1, T_2, ..., T_N, T_{N+1}\}, \tag{A.1}$$

   where $T_0$ and $T_{N+1}$ are not tasks but, respectively, central base and ending locations; and $N$ as the total number of executable tasks. In the foregoing discussion, tasks refer to executable tasks only, except when explicitly stated. Each task $T_i$ will have:

   i. A duration $d_i$.

   ii. A vector $\mathbf{R} = \{R_i\}$ of required resources by $T_i$: $R_i = \{r_{i,j}\}$ with $i = 1, ..., N_{rt}$; $N_{rt}$ is the number of resource types; $j = 1, ..., N_{it}$; $N_{it}$ is the number of items per type; and $r_{i,j}$ is the $j^{th}$ item of resource $R_i$.

   (b) A PNT $N_{tw}$ is expressed as,

   $$N_{tw} = (T, A_{rc}), \tag{A.2}$$

where $A_{rc}$ is a set of directed arcs. $Pred(j)$ defines a set of direct predecessors, while $Succ(j)$ is the set of direct successors of the task $j$. Each task $T_i \in T$ has a vector $S_i = \{s_{i,j}\}$ where,

$$s_{i,j} = \begin{cases} 1 & \text{task } j \text{ succeeds } i \\ 0 & \text{otherwise.} \end{cases} \tag{A.3}$$

(c) A matrix of operational costs $C = \{c_{i,j,k}\}$ , $i = 0, .., N$ ; $j = 0, ..., N$, and $k = 1, ..., N_{rt}$. Here $c_{i,j,k}$ is the cost of moving resource type $k$ from task $i$ to task $j$. $c_{i,0,k} = 0 \; \forall \; i, k$, i.e., no cost is imposed on the return of items to base (the starting point $T_0$)

(d) The current location of $R_i$'s $j^{th}$ items at time $t$ is denoted as $l^c_{i,j,t}$. Their collection is denoted by $\mathbf{l}^c_{i,t} = \left\{l^c_{i,j,t}\right\}$, $j = 0, \ldots, N_{it}$. $j = 0$ means the item is at the central base ($T_0$).

(e) The previous location of $R_i$'s $j^{th}$ items at time $t$ is denoted as $l^p_{i,j,t}$. Their collection is denoted by $\mathbf{l}^p_{i,t} = \left\{l^p_{i,j,t}\right\}$. $j = 0$ means the item was at the central base ($T_0$).

(f) The state of $R_i$'s $j^{th}$ items being moved or not at time $t$ is denoted as $m^v_{i,j,t}$. Their collection is denoted by $\mathbf{m}^v_{i,t} = \left\{m^v_{i,j,t}\right\}$:

$$m^v_{i,j,t} = \begin{cases} 1 & l^c_{i,j,t} \neq l^p_{i,j,t} \\ 0 & \text{otherwise.} \end{cases} \tag{A.4}$$

2. **Constraints:**

(a) Time constraint: If task $i$ is a predecessor of task $j$, then it needs to be completed before starting task $j$,

$$st_i + d_i \leq st_j, \tag{A.5}$$

$\forall \; j$, and $\forall \; i \in Prec(j)$

(b) Resource constraint: the amount of being-used resources cannot exceed the total amount,

$$r_{i,t} \leq R_i, \tag{A.6}$$

$\forall\ i$ and $\forall\ t$, where $r_{i,t}$ is the total number of being-used items of resource type $R_i$ at time $t$.

3. **Objective functions:**

(a) Makespan ($f_{ms}$) is the time needed to execute an entire schedule, i.e., equal to the end time of the last task to be accomplished,

$$f_{ms} = st_l + d_l, \tag{A.7}$$

where $st_l$ and $d_l$ are the starting time and the duration of the last task respectively.

(b) Cost of resource operations ($f_{cs}$): cost of moving resources between locations of tasks,

$$f_{cs} = \sum_{t=1\rightarrow T_{max}} \sum_{j=1\rightarrow N_{rt}} \sum_{k=1\rightarrow N_{it}} m^v_{jkt} \times c_{l^p_{jkt}, l^c_{jkt}, j}. \tag{A.8}$$

(c) Probability of a schedule to fail due to resource constraint violation defined in Equation 2.19. This probability is determined through a MCS that utilises Equation 2.18. Note that this objective is only minimised to determine solutions to sub-problems of problems from $\mathcal{L}^3$.

4. **Outputs:** The products in determining solutions to the sub-problem from $\mathcal{M}$ are:

(a) A vector of start time $\mathbf{st} = \{st_i\}$, with $i = 1, ..., N$ and $st_i$ as the starting time of task $T_i$

(b) The genotype of task IDs is defined in Equation 4.1.

(c) The solution to any of the sub-problems is a schedule defined in Equation 4.2.

## A.2 Mathematical Formulation of the Dynamic Problems from $\mathcal{M}$

Changes in problem factors, due to environmental dynamics, occur at times denoted as $\tau$.

1. **Duration**: The dynamic duration of a task $T_i$ is defined as,

$$d'_i(\tau) = N_m(d_i, \delta) + \delta, \tag{A.9}$$

where $d_i$ is the pre-defined duration of task with ID $i$; $N_m$ is a normal distribution; and $\delta$ is the standard deviation. Note that this equation is also used to generate different task duration across simulations of any sub-problem of any problem from $\mathcal{M}$. Constraint expression A.5 is rewritten as follows,

$$st_i + d'_i(\tau) \leq st_j. \tag{A.10}$$

2. **Availability of resources**: The availability status of a resource is,

$$A_{i,j,\tau} = A_v(\tau) * l^c_{i,j,\tau}, \tag{A.11}$$

where $A_v(\tau)$ takes a value of one, zero, or negative one that correspond to resource being available, broken or unavailable but at the location $l^c_{i,j,\tau}$.

3. **Number of Tasks**: Each problem from $\mathcal{M}$ only has an increase in the total number of tasks as explained in Section 4.1.1. The function representing this increase is,

$$N^c_{tw} = \Gamma_{task}\left(T^p, A^p_{rc}, N^p, N^c\right), \tag{A.12}$$

where $T^{s,p}$, $A^p_{rc}$, and $N^p$ are the sets of tasks and arcs, and the total number of tasks from the previous change respectively. Further, $N^c_{tw}$ and $N^c$ are the new PNT and

the current total number of tasks respectively.

4. **Parameters after a change**: Change is applied to the indices of tasks $I = \langle I_1, I_2, ..., \ I_{N^c} \rangle$ after any type of change. However, change is applied to the PNT $N^c_{tw}$ of Equation A.12 and the structures of $\mathbf{l}^c_{i,t}$, $\mathbf{l}^p_{i,t}$ and $\mathbf{m}^v_{i,t}$ (defined in Items 1d to 1f respectively) only when there is a change in the total number of tasks.

# Bibliography

[1] B. Abbasi, S. Shadrokh, and J. Arkat. Bi-objective resource-constrained project scheduling with robustness and makespan criteria. *Applied Mathematics and Computation*, 180(1):146–152, 2006.

[2] H. A. Abbass, R. Sarker, and C. Newton. PDE: A Pareto frontier differential evolution approach for multiobjective optimization problems. In *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 971–978, Seoul Korea, 2001. IEEE Service Center.

[3] M. B. Abello, L. T. Bui, and Z. Michalewicz. An adaptive approach for solving dynamic scheduling with time-varying number of tasks - part I. In *IEEE Congress on Evolutionary Computation*, pages 1703–1710. IEEE, 2011.

[4] M. B. Abello, L. T. Bui, and Z. Michalewicz. An adaptive approach for solving dynamic scheduling with time-varying number of tasks - part II. In *IEEE Congress on Evolutionary Computation*, pages 1711–1718. IEEE, 2011.

[5] M. B. Abello and Z. Michalewicz. Implicit memory-based technique in solving dynamic scheduling problems through Response Surface Methodology - part I: Model and method. *International Journal of Intelligent Computing and Cybernetics*, 7(2):114–142, 2014.

[6] M. B. Abello and Z. Michalewicz. Implicit memory-based technique in solving dynamic scheduling problems through Response Surface Methodology - part II:

Experiments and analysis. *International Journal of Intelligent Computing and Cybernetics*, 7(2):143–174, 2014.

[7] M. B. Abello and Z. Michalewicz. Multi-Objective Resource-Constrained Project Scheduling with a Time-varying Number of Tasks. *The Scientific World Journal*, DOI: 10.1155/2014/420101, 2014.

[8] M. B. Abello, Z. Michalewicz, and L. T. Bui. A reactive-proactive approach for solving dynamic scheduling with time-varying number of tasks. In *IEEE Congress on Evolutionary Computation*, pages 1–10. IEEE, 2012.

[9] D. Aberdeen, S. Thibaux, and L. Zhang. Decision-theoretic military operations planning. In *Proceedings of the International Conference on Autonomous Planning and Scheduling*, pages 402–411. AAAI, 2004.

[10] J. Alcaraz and C. Maroto. A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research*, 102(4):83–109, 2001.

[11] M. A. Aloulou and M.C. Portmann. An efficient proactive-reactive scheduling approach to hedge against shop floor disturbance. In *Multidisciplinary Scheduling: Theory and Applications*. 1st International Conference, MISTA '03, 2003.

[12] S. Alvisi and M. Franchini. Multiobjective optimization of rehabilitation and leakage detection scheduling in water distribution systems. *Journal of Water Resources Planning and Management*, 135(6):426–439, 2009.

[13] K. P. Anagnostopoulos and G. Mamanis. A portfolio optimization model with three objectives and discrete variables. *Computers & Operations Research*, 37(7):1285–1297, 2010.

[14] S. Andreev, G. Rzevski, P. Shviekin, P. Skobelev, and I. Yankov. A multi-agent scheduler for rent-a-car companies. In *HoloMAS '09: Proceedings of the 4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, pages 305–314, Berlin, Heidelberg, 2009. Springer-Verlag.

[15] C. Artigues and F. Roubellat. A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *European Journal of Operational Research*, 127(2):297–316, 2000.

[16] P. Artzner, F. Delbaen, J.M. Eber, and D. Heath. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999.

[17] R. Axelrod. *The Complexity of Cooperation*. Princeton University Press, Princeton, NJ, 1997.

[18] A. Azaron, K.N. Brown, S.A. Tarim, and M. Modarres. A multi-objective stochastic programming approach for supply chain design considering risk. *International Journal of Production Economics*, 116(1):129–138, November 2008.

[19] R. Baati, A. Kamoun, M. Chaabouni, M. Sergent, and R. Phan-Tan-Luu. Screening and optimization of the factors of a detergent admixture preparation. *Chemometrics and Intelligent Laboratory Systems*, 80(2):198–208, 2006.

[20] M. B. Bader-El-Den, R. Poli, and S. Fatima. Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework. *Memetic Computing*, 1(3):205–219, 2009.

[21] A. D. Baker. A survey of factory control algorithms which can be implemented in a multi-agent heterarchy. *Journal of Manufacturing Systems*, 17(4):279–320, 1998.

[22] G. Barlow and S. Smith. A memory enhanced evolutionary algorithm for dynamic scheduling problems. In Mario Giacobini et al., editor, *Applications of Evolutionary Computing: EvoWorkshops 2008*, volume 4974/2008, pages 606–615, Berlin / Heidelberg, April 2008. Springer.

[23] T. Barrows. The impact of multi-agent systems on supply chain management - a review of the literature. Technical Report MKX 5150, Department of Marketing Monash University Faculty of Business & Economics, 2004.

[24] R. Battiti, M. Brunato, and P. Campigotto. Learning while optimizing an unknown fitness surface. In *2nd International Conference on Learning and Intelligent Optimization*, volume 5313, pages 25–40, 2008.

[25] L. Belfares, W. Klibi, N. Lo, and A. Guitouni. Multi-objective tabu search based algorithm for progressive resource allocation. *European Journal of Operational Research*, 177(3):1779–1799, 2007.

[26] D. J. Berndt and A. Watkins. Investigating the performance of genetic algorithm-based software test case generation. In *High-Assurance Systems Engineering*, pages 261–262. IEEE Computer Society, 2004.

[27] C. Bierwirth and H. Kopfer. Dynamic task scheduling with genetic algorithms in manufacturing systems, 1994.

[28] C. Bierwirth and D. C. Mattfeld. Production scheduling and rescheduling with genetic algorithms. *Evolutionary Computation*, 7(1):1–18, 1999.

[29] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: from Natural to Artificial Systems*. Oxford University Press, Oxford, 1999.

[30] A. Bonfill, A. E. Camarasa, and L. Puigjaner. Proactive approach to address the uncertainty in short-term scheduling. *Computers & Chemical Engineering*, 32(8):1689–1706, 2008.

[31] C. M. Borror and D. C. Montgomery. A modified path of steepest ascent for split-plot experiments. *Journal Of Quality Technology*, 37(1):75–83, 2005.

[32] A. Boukhtouta, A. Bedrouni, J. Berger, F. Bouak, and A. Guitouni. A survey of military planning systems. In *The 9th ICCRTS International Command and Control Research and Technology Symposium*, Copenhagen, Denmark, 2004.

[33] G. Box and D. Behnken. Some new three level designs for the study of quantitative variables. *Technometrics*, 2(4):455–475, 1960.

[34] K. Brandisky, D. Sankowski, and R. Banasiak. ECT sensor optimization based on RSM and GA. *International Journal of Intelligent Computing and Cybernetics*, 31(3):858–869, 2012.

[35] I. Branke, T. Kauler, C. Schmidt, and H. Schmeck. *A multipopulation approach to dynamic optimization problems*. Springer-Verlag, Berlin, Germany, 2000.

[36] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 3, pages 1875–1882, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.

[37] J. Branke. *Evolutionary optimization in dynamic environments*. Kluwer Academic Publishers, Massachusetts USA, 2002.

[38] R. Brennan, P. Vrba, P. Tichy, A. Zoitl, C. Snder, T. Strasser, and V. Mak. Developments in dynamic and intelligent reconfiguration of industrial automation. *Computers in Industry*, 59(6):533–547, 2008.

[39] M.E. Bruni, P. Beraldi, F. Guerriero, and E. Pinto. A heuristic approach for resource constrained project scheduling with uncertain activity durations. *Computers & Operation Research*, 38(9):1305–1318, 2011.

[40] L. T. Bui, Z. Michalewicz, E. Parkinson, and M. B. Abello. Adaptation in dynamic environments: A case study in mission planning. *IEEE Transactions on Evolutionary Computation*, 16(2):190–209, 2012.

[41] L.T. Bui, M. Barlow, and H.A. Abbass. A multiobjective risk-based framework for mission capability planning. *New Mathematics and Natural Computation*, 5(2):459 – 485, 2009.

[42] L.T. Bui, J. Branke, and H.A. Abbass. Multiobjective optimization for dynamic environments. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2349–2356, 2005.

[43] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward. Exploring hyper-heuristic methodologies with genetic programming. In C. Mumford and L. Jain, editors, *Computational Intelligence*, volume 1 of *Intelligent Systems Reference Library*, pages 177–201. Berlin, Heidelberg: Springer, 2009.

[44] W. Cai, A. Pacheco-Vega, M. Sen, and K. T. Yang. Heat transfer correlations by symbolic regression. *International Journal of Heat and Mass Transfer*, 49(23-24):4352–4359, November 2006.

[45] A. B. Carlson. *Communication systems : an introduction to signals and noise in electrical communication, 3rd ed.* New York : McGraw-Hill, 1986.

[46] J. A. Carruthers and A. Battersby. Advances in critical path methods. *Operations research quarterly*, 17(4):359–380, 1966.

[47] F. Castillo, A. Kordon, and G. Smits. Robust pareto front genetic programming parameter selection based on design of experiments and industrial data. In Rick L. Riolo, Terence Soule, and Bill Worzel, editors, *Genetic Programming Theory and Practice IV*, volume 5 of *Genetic and Evolutionary Computation*, chapter 2, pages –. Springer, 11-13 May 2006.

[48] S. H. Chen, M. C. Chen, P. C. Chang, Q. Zhang, and Y. M. Chen. Guidelines for developing effective estimation of distribution algorithms in solving single machine scheduling problems. *Expert System Applications*, 37(9):6441–6451, 2010.

[49] S. H. Chen, J. Duffy, and C. H. Yeh. Equilibrium selection via adaptation: Using genetic programming to model learning in a coordination game. *The Electronic Journal of Evolutionary Modeling and Economic Dynamics*, 15 January 2002.

[50] C. K. Chow and S. Y. Yuen. An evolutionary algorithm that makes decision based on the entire previous search history. *IEEE Transactions Evolutionary Computation*, 15(6):741–769, 2011.

[51] H. G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent non-stationary environments. Technical Report AIC-90-001, Naval Research Laboratory, Washington, DC, 1990.

[52] D. Dasgupta and D. R. McGregor. Non-stationary function optimization using the structured genetic algorithm. In *Parallel Problem Solving from Nature, R.Mnner and B. Manderick, Eds.*, volume 1498, pages 145–154, Amsterdam, The Netherlands, 1992. Elsevier.

[53] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, New York, 2001.

[54] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the third international conference on Genetic algorithms*, pages 42–50, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[55] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions On Evolutionary Computation*, 6(2):182–197, 2002.

[56] E. Demeulemeester and W. Herroelen. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38(12):1803 – 1818, 1992.

[57] M. M. Deza and E. Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2009.

[58] O. Djamila and S. Petrovic. A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, 12(4):417–431, 2009.

[59] A. Drexl and A. Kimms. Optimization guided lower and upper bounds for the resource investment problem. *Journal of Operations Reseach Society*, 52:340–351, 2001.

[60] M. P. Dubuisson and A. K. Jain. A modified hausdorff distance for object matching. In *Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on Pattern Recognition*, volume 1, pages 566 – 568, 1994.

[61] W. Duch, R.J. Oentaryo, and M. Pasquier. Cognitive architectures: Where do we go from here? In *Proceedings of the First AGI Conference, eds. Wang, P., Goertzel, B., and Franklin, S.* IOS Press, Amsterdam, 2008.

[62] O. Engin, G. Ceran, and M. K. Yilmaz. An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. *Applied Soft Computing*, 11(3):3056–3065, 2011.

[63] A. I. Esparcia-Alcazar and K. C. Sharman. Genetic programming techniques that evolve recurrent neural networks architectures for signal processing. In *IEEE Workshop on Neural Networks for Signal Processing*, Seiko, Kyoto, Japan, September 1996.

[64] R. Everson, J. Fieldsend, and S. Singh. Full elite sets for multi-objective optimization. In *Proceedings of the fifth international conference on adaptive computing in design and manufacture*, page 87100, 2002.

[65] C. Fang and L. Wang. An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem. *Computers & Operation Research*, 39(5):890–901, 2012.

[66] L. Fei and L. Juan. Optimal genes selection with a new multi-objective evolutional algorithm hybriding NSGA-II with EDA. In *BioMedical Engineering and Informatics (1)*, pages 327–331. IEEE Computer Society, 2008.

[67] S.L.C. Ferreira, R.E. Bruns, H.S. Ferreira, G.D. Matos, J.M. David, G.C. Brand, E.G.P. da Silva, L.A. Portugal, P.S. dos Reis, A.S. Souza, and W.N.L. dos Santos. Box-Behnken design: an alternative for the optimization of analytical methods. *Analytica Chimica Acta*, 597(2):179–186, 2007.

[68] A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In *3rd Conference on Learning and Intelligent Optimization*, volume 5851, pages 176–190, 2009.

[69] M. Fleischer. The measure of pareto optima: Applications to multi-objective metaheuristics. In C. M. et al. Fonseca, editor, *Evolutionary Multiobjective Optimization*, volume 2632 of *Lecture Notes in Computer Science*, page 519533. Berlin, Germany: Springer-Verlag, 2003.

[70] M. Fletcher and J. Brusey. The Story of the Holonic Packing Cell. In *Proceedings of 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems*. ACM Press, Melbourne, 2003.

[71] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Procedings of the fifth International Conference on Genetic Algorithms, S. Forrest, Ed.*, pages 416–423, 1993.

[72] T. Frankola, M. Golub, and D. Jakobovic. Evolutionary algorithms for the resource constrained scheduling problem. In *Proceedings of the ITI 2008 30th International Conference on Information Technology Interfaces, Cavtat, Croatia*, pages 715–722, 2008.

[73] H. Fu, B. Sendhoff, K. Tang, and X. Yao. Characterizing environmental changes in robust optimization over time. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.

[74] A. L. Garcia-Almanza and E. P. K. Tsang. Forecasting stock prices using genetic programming and chance discovery. In *12th International Conference On Computing In Economics And Finance*, page number 489, July 2006.

[75] F. Ghezail, H. Pierreval, and S. Hajri-Gabouj. Analysis of robustness in proactive scheduling: A graphical approach. *Computers & Industrial Engineering*, 58(2):193–198, 2010.

[76] A. Giret and V. Botti. Holons and agents. *Journal of Intelligent Manufacturing*, 15(5):645659, 2004.

[77] C. Gogu and J. C. Passieux. Efficient surrogate construction by combining response surface methodology and reduced order modeling. *Structural and Multidisciplinary Optimization*, 47(6):821–837, 2013.

[78] A. R. Goncalves and F. J. Von Zuben. Hybrid evolutionary algorithm guided by a fast adaptive gaussian mixture model applied to dynamic optimization problems. In *Proceedings of III Workshop on Computational Intelligence (WCI 2010)- Joint Conference, Sao Bernardo do Campo, Brazil*, pages 553–558, 2010.

[79] A. R. Goncalves and F. J. Von Zuben. Online learning in estimation of distribution algorithms for dynamic environments. In *IEEE Congress on Evolutionary Computation 2011*, 2011.

[80] R. Goodacre. Explanatory analysis of spectroscopic data using machine learning of simple, interpretable rules. *Vibrational Spectroscopy*, 32(1):33–45, 5 August 2003.

[81] L. Gou, P. B. Luh, and Y. Kyoya. Holonic manufacturing scheduling: Architecture, cooperation mechanism, and implementation. *Comput. Ind.*, 37(3):213–231, 1998.

[82] J.J. Grefenstette. Genetic algorithms for dynamic environments. In *Parallel problem solving from nature*, volume 2, pages 137–144, 1992.

[83] T. D. Gwiazda. *Genetic Algorithms Reference; Volume I Crossover for single objective numerical optimization problems.* Tomasz Gwiazda, 2006.

[84] T. D. Gwiazda. *Genetic Algorithms Reference; Volume II Mutation operator numerical optimization problems.* Tomasz Gwiazda, 2007.

[85] B. S. Hadad and C. F. Eick. Supporting polyploidy in genetic algorithms using dominance vectors. In Peter J. Angeline, Robert G. Reynolds, John R. McDonnell, and Russell C. Eberhart, editors, *Evolutionary Programming*, volume 1213 of *Lecture Notes in Computer Science*, pages 223–234. Springer, 1997.

[86] P. Hajela and C. Y. Lin. Genetic search strategies in multicriterion optimal design. *Structural and Multidisciplinary Optimization*, 4(2):99–107, 1992.

[87] H. Handa. Solving multi-objective reinforcement learning problems by EDA-RL acquisition of various strategies. In *2009 Ninth International Conference on Intelligent Systems Design and Applications*, pages 426–431, 2009.

[88] S. Hartmann and R. Kolisch. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127(2):394–407, 2000.

[89] S. S. Heragu, R. J. Graves, B. Kim, and A. S. Onge. Intelligent agent based framework for manufacturing systems control. *IEEE Transactions on Systems, Man, and Cyberneticspart A: Systems and Humans*, 32(5):560–573, 2002.

[90] W. Herroelen and R. Leus. Project scheduling under uncertainty: Survey and research potentials. *European journal of operational research*, 165(2):289–306, 2005.

[91] F. S. Hillier and G. J. Liebermann. *Introduction to operations research.* McGraw-Hill, 2005.

[92] K. S. Hindi, H. Yang, and K. Fleszar. An evolutionary algorithm for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 6(5):512–518, 2002.

[93] R. Hochreiter. Evolutionary multi-stage financial scenario tree generation. In Cecilia Di Chio, Anthony Brabazon, Gianni A. Di Caro, Marc Ebner, Muddassar Farooq, Andreas Fink, Jrn Grahl, Gary Greenfield, Penousal Machado, Michael O'Neill, Ernesto Tarantino, and Neil Urquhart, editors, *EvoApplications (2)*, volume 6025 of *Lecture Notes in Computer Science*, pages 182–191. Springer, 2010.

[94] J. Hodík, J. Vokřínek, and M. Jakob. Contract monitoring in agent-based systems: Case study. In *HoloMAS '09: Proceedings of the 4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, pages 295–304, Berlin, Heidelberg, 2009. Springer-Verlag.

[95] J. H. Holland. *Adaptation in natural and artificial systems.* MIT Press, 1992.

[96] J. Horn and N. Nafpliotis. Multiobjective optimization using the niched pareto genetic algorithm. Illinois Genetic Algorithms Lab., Univ. Illinois, Urbana, IL, IlliGAL Tech. Rep. 93005, 1993.

[97] D. Howard, S. C. Roberts, and R. Brankin. Target detection in imagery by genetic programming. *Advances in Engineering Software*, 30(5):303–311, 1999.

[98] D. Howard, S. C. Roberts, and C. Ryan. Pragmatic genetic programming strategy for the problem of vehicle detection in airborne reconnaissance. *Pattern Recognition Letters*, 27(11):1275–1288, 2006.

[99] M. W. Iruthayarajan and S. Baskar. Evolutionary algorithms based design of multivariable PID controller. *Expert System Applications*, 36(5):9159–9167, 2009.

[100] K. Iwamura, N. Mayumi, Y. Tanimizu, and N. Sugimura. A study on real-time scheduling for holonic manufacturing systems — determination of utility values based on multi-agent reinforcement learning. In *HoloMAS '09: Proceedings of the*

*4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, pages 135–144, Berlin, Heidelberg, 2009. Springer-Verlag.

[101] D. Jafari, S.M.M. Husseini, M.H. Fazel Zarandi, and R. Zanjirani Farahani. Coordination of order and production policy in buyer-vendor chain using PROSA Holonic architecture. *International Journal of Advanced Manufacturing Technology*, 45(9–10):1033–1050, 2009.

[102] D. Jakobović and L. Budin. Dynamic scheduling with genetic programming. In Pierre Collet, Marco Tomassini, Marc Ebner, Steven Gustafson, and Anikó Ekárt, editors, *Proceedings of the 9th European Conference on Genetic Programming*, volume 3905 of *Lecture Notes in Computer Science*, pages 73–84, Budapest, Hungary, 10 - 12 April 2006. Springer.

[103] M.P. Jenarthanan, R. Jeyapaul, and N. Naresh. Modelling and analysis of factors influencing surface roughness and delamination of milling of GFRP laminates using rsm. *International Journal of Intelligent Computing and Cybernetics*, 8(4):489–504, 2012.

[104] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

[105] Y. Jin, M. Olhofer, and B. Sendhoff. Managing approximate models in evolutionary aerodynamic design optimization. In *IEEE Congress on Evolutionary Computation, Seoul, South Korea*, pages 592–599. Inst. of Elec. and Elec. Eng. Computer Society, 2001.

[106] Y. Jin and B. Sendhoff. Trade-off between performance and robustness: an evolutionary multiobjective approach. In *EMO'03: Proceedings of the 2nd international conference on Evolutionary multi-criterion optimization*, pages 237–251, Berlin, Heidelberg, 2003. Springer-Verlag.

[107] M. A. Kaliakatsos-Papakostas, M. G. Epitropakis, and M. N. Vrahatis. Weighted markov chain model for musical composer identification. In *EvoApplications (2)*, pages 334–343, 2011.

[108] N. Karimi, M. Zandieh, and H. R. Karamooz. Bi-objective group scheduling in hybrid flexible flowshop: A multi-phase approach. *Expert System Application*, 37(6):4024–4032, 2010.

[109] P. Keskinocak, F. Wu, R. Goodwin, S. Murthy, R. Akkiraju, S. Kumaran, and A. Derebail. Scheduling solutions for the paper industry. *Operation Research*, 50(2):249–259, March 2002.

[110] R.H. Kewley and M.J. Embrechts. Computational military tactical planning system. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 32(2):161–171, 2002.

[111] C.Y. Khor and M.Z. Abdullah. Optimization of IC encapsulation considering fluid/structure interaction using response surface methodology. *Simulation Modelling Practice and Theory*, 29:109–122, 2012.

[112] J. D. Knowles and D. W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[113] A. Koestler. *The Ghost in the Machine*. Arkana Books, London, 1969.

[114] R. Kolishch and S. Hartmann. Experimental investigation of heuristic for resource constrained project scheduling: An update. *European Journal of Operational Research*, 174:23–37, 2006.

[115] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[116] J. R. Koza. *Genetic programming - on the programming of computers by means of natural selection.* Complex adaptive systems. MIT Press, 1993.

[117] O. Lambrechts, E. Demeulemeester, and W. Herroelen. Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *Journal of Scheduling*, 11(2):121–136, 2007.

[118] W. B. Langdon, R. Poli, N. F. McPhee, and J. R. Koza. Genetic programming: An introduction and tutorial, with a survey of techniques and applications. In John Fulcher and Lakhmi C. Jain, editors, *Computational Intelligence: A Compendium*, volume 115 of *Studies in Computational Intelligence*, pages 927–1028. Springer, 2008.

[119] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer, Boston, MA, 2002.

[120] J. Lee. *Riemannian Manifolds: An Introduction to Curvature*. Springer New York, New York, NY, 1997.

[121] S. Legg and M. Hutter. Universal intelligence: A definition of machine intelligence. *Computing Research Repository*, abs/0712.3329, 2007.

[122] P. Leitao. Holonic rationale and self-organization on design of complex evolvable systems. In *HoloMAS '09: Proceedings of the 4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, pages 1–12, Berlin, Heidelberg, 2009. Springer-Verlag.

[123] J. Leung. *Handbook of scheduling: algorithms, models, and performance analysis*. Chapman & Hall/CRC, 2004.

[124] T. L. Lew, A. B. Spencer, F. Scarpa, K. Worden, A. Rutherford, and F. Hemez. Identification of response surface models using genetic programming. *Mechanical Systems and Signal Processing*, 20(8):1819–1831, November 2006.

[125] J. Lewis, E. Hart, and G. Ritchie. A comparison of dominance mechanisms and simple mutation on non-stationary problems. In *Parallel Problem Solving from*

*Nature. ser. LNCS, A. E. Eiben, T. Bck, M. Schoenauer, and H.-P. Schwefel, Eds*, volume 1498, pages 139–148, Berlin, Germany, 1998. Springer-Verlag.

[126] M. Li and P. Vitanyi. *An introduction to Kolmogorov complexity and its applications*. Springer Verlag, London, 1997.

[127] S.C. Lin, E. D. Goodman, and W. F. Punch. A genetic algorithm approach to dynamic job shop scheduling problems. In *Proceedings of an International Conference on Genetic Algorithms, T. Bck, Ed.*, pages 481–488, 1997.

[128] F. Liu and G. Zeng. Study of genetic algorithm with reinforcement learning to solve the TSP. *Expert Systems with Applications*, 36:6995–7001, 2009.

[129] H. Liu, L. Gao, and Q. Pan. A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem. *Expert Systems with Applications*, 38(4):4348–4360, 2011.

[130] X. Liu, Y. Wu, and J. Ye. An improved estimation of distribution algorithm in dynamic environments. In *Fourth International Conference on Natural Computation*, volume 6, pages 269–272, 2008.

[131] S.J. Louis and J. McDonnell. Learning with case-injected genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(4):316–328, 2004.

[132] S. Luke. Two fast tree-creation algorithms for genetic programming. *IEEE Trans. Evolutionary Computation*, 4(3):274–283, 2000.

[133] A. K. Maiti and M. Maiti. Discounted multi-item inventory model via genetic algorithm with roulette wheel selection, arithmetic crossover and uniform mutation in constraints bounded domains. *International Journal of Computer Mathematics*, 85(9):1341–1353, 2008.

[134] L. Martins, C. Francisco, J. Redol, J. Craveirinha, J. Clímaco, and P. Monteiro. Evaluation of a multiobjective alternative routing method in carrier IP/MPLS net-

works. In *NETWORKING '09: Proceedings of the 8th International IFIP-TC 6 Networking Conference*, pages 195–206, Berlin, Heidelberg, 2009. Springer-Verlag.

[135] D. C. McFarlane and S. Bussmann. Developments in holonic production planning and control. *International Journal on Production Planning and Control*, 11(6):522–536, 2000.

[136] D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 6(4):333–346, 2002.

[137] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, London, UK, 3rd edition, 1996.

[138] P. Miller. The genius of swarms. *National Geographic*, 2007.

[139] T. Miyazaki. An evaluation pattern generation scheme for electric components in hybrid electric vehicles. In *2010 IEEE International Conference on Control Applications*, pages 530–535, 2010.

[140] B. Mo, A. Gjelsvik, and A. Grundt. Integrated risk management of hydro power scheduling and contract management. *IEEE Transactions on Power Systems*, 16:216–221, 2001.

[141] D. C. Montgomery. *Design and analysis of experiments*. Wiley, 7. ed., international student version edition, 2009.

[142] N. Mori, S. Imanishia, H. Kita, and Y. Nishikawa. Adaptation to changing environments by means of the memory based thermodynamical genetic algorithms. In T. Bäck, editor, *Seventh International Conference on Genetic Algorithms*, pages 299–306. Morgan Kaufmann, 1997.

[143] N. Muhammad, Y. H. P. Manurung, R. Jaafar, S. K. Abas, G. Tham, and E. Haruman. Model development for quality features of resistance spot welding using multi-

objective taguchi method and response surface methodology. *Journal Intelligent Manufacturing*, 24(6):1175–1183, 2013.

[144] R. H. Myers and D. C. Montgomery. *Response Surface Methodology. Process and Product Optimization Using Designed Experiments*. John Wiley and Sons Inc., 1995.

[145] A. Nagar, J. Haddock, and S. Heragu. Multiple and bi-criteria scheduling: a literature survey. *European Journal of Operational Research*, 81(1):88104, 1995.

[146] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan. A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Transactions on Evolutionary Computation*, 17(5):621–639, 2013.

[147] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan. Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. *IEEE Transactions on Evolutionary Computation*, 18(2):193–208, APRIL 2014.

[148] T.B.M.J. Ouarda and J.W. Labadie. Chance-constrained optimal control for multireservoir system optimization and risk analysis. *Stochastic Environmental Research and Risk Assessment*, 15:185–204, 2001.

[149] R. Padman and S. Roehrig. A genetic programming approach for heuristic selection in constrained project scheduling. In *5th INFORMS Computer Science Technical Section Conference, Eds. RS Barr, RV Helgason, JL Kennington*, pages 405–421, 1997.

[150] A. J. Page and T. J. Naughton. Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing. In *Proceedings of the 19th IEEE/ACM International Parallel and Distributed Processing Symposium*, Denver, Colorado, USA, April 2005. IEEE Computer Society.

[151] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz. BOA: The bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, 1999.

[152] X. Peng, X. Gao, and S. Yang. Environment identification-based memory scheme for estimation of distribution algorithms in dynamic environments. *Soft Computing*, 15:311–326, 2011.

[153] M. V. F. Pereira. Optimal stochastic operations scheduling of large hydroelectric systems. *Electrical Power and Energy Systems*, 11(3):161–169, 1989.

[154] C. W. Pickardt, T. Hildebrandt, J. Branke, J. Heger, and B. Scholz-Reiter. Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. *International Journal on Production Economics*, 145:67–77, 2013.

[155] M. Pickett, D. Miner, and T. Oates. Essential phenomena of general intelligence. In *Proceedings of the First AGI Conference, eds. Wang, P., Goertzel, B., and Franklin, S.* IOS Press, Amsterdam, 2008.

[156] M. L. Pinedo. *Scheduling: theory, algorithms, and systems 4th ed.* New York : Springer, 2012.

[157] A. Popov and H. Werner. Efficient design of low-order $H_\infty$ optimal controllers using evolutionary algorithms and a bisection approach. In *Proceedings of the joined IEEE Conference on Computer Aided Control System Design, International Conference on Control Applications and the IEEE International Symposium on Intelligent Control*, pages 760–765, Munich, Germany, 2006.

[158] Wikipedia Post. Watson (computer). `http://en.wikipedia.org/wiki/Watson_(computer)`.

[159] R. C. Purshouse and P. J. Fleming. Evolutionary multi-objective optimisation: An exploratory analysis. In *Proceedings of the 2003 Congress on Evolutionary Compu-*

*tation (CEC2003), Canberra, Australia*, volume 3, pages 2066–2073. IEEE Press, 2003.

[160] J. C. Quinton, J. C. Buisson, and F. Perotto. Anticipative coordinated cognitive processes for interactivist and piagetian theories. In *Proceedings of the First AGI Conference, eds. Wang, P., Goertzel, B., and Franklin, S.* IOS Press, Amsterdam, 2008.

[161] C. L. Ramsey and J. J. Grefenstette. Case-based initialization of genetic algorithms. In *Proceedings of International Conference on Genetic Algorithms, S. Forrest, Ed.*, pages 84–91, 1993.

[162] A. S. Raza and A. Akgunduz. A comparative study of heuristic algorithms on economic lot scheduling problem. *Computers & Industrial Engineering*, 55(1):94–109, 2008.

[163] S. A. Raza, A. Akgunduz, and M. Y. Chen. A tabu search algorithm for solving economic lot scheduling problem. *Journal on Heuristics*, 12(6):413–426, 2006.

[164] C. Reeves and H. Karatza. *Dynamic sequencing of a multiprocessor system: A genetic algorithm approach*. Springer-Verlag, Berlin, Germany, 1993.

[165] E. Ridge and D. Kudenko. Analyzing heuristic performance with response surface models: prediction, optimization and robustness. In Hod Lipson, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2007)*, pages 150–157. ACM, 2007.

[166] E. Ridge and D. Kudenko. Tuning the performance of the MMAS heuristic. In Thomas Sttzle, Mauro Birattari, and Holger H. Hoos, editors, *SLS*, volume 4638 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2007.

[167] B. Roorda. An algorithm for sequential tail value at risk for path-independent payoffs in a binomial tree. *Annals Operation Research*, 181(1):463–483, 2010.

[168] M. R. Rose and G. V. Lauder. *Adaptation*. Academic Press, 1996.

[169] P.J. Ross. *Taguchi techniques for quality engineering.* McGraw-Hill, 1988.

[170] B.C. Routara, A.K. Sahoo, A.K. Parida, and P.C. Padhi. Response surface methodology and genetic algorithm use to optimize the cutting condition for surface roughness parameters in CNC turning. In *International Conference on Modeling, Optimization and Computing (ICMOC, 2012)*, volume 38, pages 1893–1904. Elsevier Ltd., 2012.

[171] C. Rovelli. *Quantum gravity.* Cambridge, UK ; New York : Cambridge University Press, 2004.

[172] C. Russ and A. Walz. MACSIMA: On the effects of adaptive negotiation behavior in agent-based supply networks. In Lars Braubach, Wiebe van der Hoek, Paolo Petta, and Alexander Pokahr, editors, *MATES*, volume 5774 of *Lecture Notes in Computer Science*, pages 128–140. Springer, 2009.

[173] A. Ruszczynski and A. Shapiro. *Stochastic programming. Handbooks in Operations Research and Management Science, vol. 10.* Elsevier Science B.V., Amsterdam, 2003.

[174] E. Ryan, R. M. A. Azad, and C. Ryan. On the performance of genetic operators and the random key representation. In Maarten Keijzer, Una-May O'Reilly, Simon M. Lucas, Ernesto Costa, and Terence Soule, editors, *EuroGP*, volume 3003 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 2004.

[175] I. Sabuncuoglu and S. Goren. Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. *International Journal on Computer Integrated Manufacturing*, 22(2):138–157, 2009.

[176] S. J. Sadjadi, R. Soltani, M. Izadkhah, F. Saberian, and M. Darayi. A new nonlinear stochastic staff scheduling model. *Scientia Iranica*, 18(3):699–710, 2011.

[177] M. Safea, S. M. Khazraeeb, P. Setoodeha, and A. H. Jahanmiria. Model reduction and optimization of a reactive dividing wall batch distillation column inspired by re-

sponse surface methodology and differential evolution. *Mathematical and Computer Modelling of Dynamical Systems*, 19(1):29–50, 2013.

[178] A. Salhi, J. A. V. Rodriguez, and Q. Zhang. An estimation of distribution algorithm with guided mutation for a complex flow shop scheduling problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2007)*, volume 1–2, pages 570–576, 2007.

[179] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the International Conference on Genetic Algorithms and their Applications*, volume 3, pages 93–100, 1985.

[180] R. Scheffermann, M. Bender, and A. Cardeneo. Robust solutions for vehicle routing problems via evolutionary multiobjective optimization. In *CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, pages 1605–1612, Piscataway, NJ, USA, 2009. IEEE Press.

[181] M. Schoenauer and Z. Michalewicz. Evolutionary computation: An introduction. *Control and Cybernetics*, 26:307 338, 1997.

[182] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

[183] S. K. Shakya. Probabilistic model building genetic algorithm (PMBGA): A survey. Technical report, Computational Intelligence Group, School of computing, The Robert Gordon University, Aberdeen, Scotland, UK, 2003.

[184] D. Shilane, J. Martikainen, and S. J. Ovaska. Time-dependent performance comparison of evolutionary algorithms. In Mikko Kolehmainen, Pekka J. Toivanen, and Bartlomiej Beliczynski, editors, *International Conference on Adaptive and Natural Computing Algorithm*, volume 5495 of *Lecture Notes in Computer Science*, pages 223–232. Springer, 2009.

[185] A. B. Simões and E. Costa. An immune system-based genetic algorithm to deal with dynamic environments: Diversity and memory. In *Proc. of the Sixth International Conference on Neural Networks and Genetic Algorithms (ICANNGA'03)*, pages 168–174, April 2003.

[186] R. Slowinski. *Advances in project Scheduling*, chapter Multiobjective project scheduling under multiple-category resource constraints. Elsevier, 1989.

[187] A. Sprecher and A. Drexl. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, 107:431–450, 1998.

[188] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Journal of Evolutionary Computation*, 2(3):221–248, 1994.

[189] D. Stark. Heterarchy: Distributing intelligence and organizing diversity. In *The Biology of Business: Decoding the Natural Laws of Enterprise*, pages 153–179, San Francisco, CA, 1999. Jossey-Bass.

[190] R. S. Sutton and A. G. Barto. *Reinforcement learning : an introduction*. MIT Press, Cambridge, Mass., 1998.

[191] V. Tcheprasov, W. F. Punch, G. Ragatz, I. P. Norenkov, and E.D. Goodman. A genetic algorithm to generate a predictive planner for assembly of printed circuit boards. *Review of Applied and Industrial Mathematics*, 3(5):733–749, 1996.

[192] A. Thamarajah. A self-organizing model for scheduling distributed autonomous manufacturing systems. *Cybernetics Systems*, 29(5):461–480, 1998.

[193] R. Tins and S. Yang. Use of the q-Gaussian mutation in evolutionary algorithms. *Soft Computing*, 15(8):1523–1549, 2011.

[194] T. Tometzki and S. Engell. Risk management in production planning under uncertainty by multi-objective hybrid evolutionary algorithms. In S. Pierucci and B.G.

Ferraris, editors, *20th European Symposium on Computer Aided Process Engineering*, volume 28 of *Computer-Aided Chemical Engineering*, pages 151–156, 2010.

[195] M. Tripathi, S. Agrawal, and M. K. Pandey. Real world disassembly modeling and sequencing problem: Optimization by algorithm of self-guided ants (asga). *Robotics And Computer-Integrated Manufacturing*, 25(3):483–496, 2009.

[196] M. Tripathi, G. Kuriger, and H. da Wan. An ant based simulation optimization for vehicle routing problem with stochastic demands. In Ann Dunkin, Ricki G. Ingalls, Enver Ycesan, Manuel D. Rossetti, Ray Hill, and Bjrn Johansson, editors, *Winter Simulation Conference*, pages 2476–2487. WSC, 2009.

[197] S. Tustsui and A. Ghosh. Genetic algorithms with a robust solution searching. *IEEE Transactions on Evolutionary Computation*, 1(3):201–208, 1997.

[198] C. J. Tzeng and R. Y. Chen. Optimization of electric discharge machining process using the response surface methodology and genetic algorithm approach. *International Journal of Precision Engineering and Manufacturing*, 14(5):709–717, 2013.

[199] N. Umang and M. Bierlaire. Real time recovery in berth allocation problem in bulk ports, 2012.

[200] R.K. Ursem. Multinational GAs: Multimodal Optimization Techniques in Dynamic Environments. In *Proceedings of the Genetic and Evolutionary Computation Conf. (GECCO-2000)*, pages 19–26. San Francisco, CA: Morgan Kaufmann, 2000.

[201] US Department of the Army. FM 5-0: Army Planning and Orders Production, 2005.

[202] A. S. Uyar and A. E. Harmanci. A new population based adaptive dominance change mechanism for diploid genetic algorithms in dynamic environments. *Soft Computing*, 9(11):803–815, 2005.

[203] P. Valckenaers and H. Van Brussel. Holonic manufacturing execution systems. *CIRP Annals - Manufacturing Technology*, 54(1):427–432, 2005.

[204] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. Reference architecture for holonic manufacturing systems: PROSA. *Computers & Industrial Engineering*, 37(3):255–274, 1998.

[205] S. Van de Vonder, E. Demeulemeester, and W. Herroelen. A classification of predictive-reactive project scheduling procedures. Technical report, Katholieke Universiteit Leuven, 2007.

[206] A. Viana and J. P. de Sousa. Using metaheuristics in multiobjective resource constrained project scheduling. *European Journal of Operational Research*, 120(2):359–374, 2000.

[207] A. E. P. Villa, W. Duch, P. Erdi, F. Masulli, and G. Palm, editors. *Artificial Neural Networks and Machine Learning*, volume 7552 of *Lecture Notes in Computer Science*. Springer, 2012.

[208] S. Vonder, E. Demeulemeester, and W. Herroelen. Proactive heuristic procedures for robust project scheduling: An experimental analysis. *European Journal of Operational Research*, 189(3):723–733, 2008.

[209] L. Wang and C. Fang. An effective estimation of distribution algorithm for the multimode resource-constrained project scheduling problem. *Computers & Operation Research*, 39(2):449–460, 2012.

[210] M. Wang, J. Dong, W. Wang, J. Zhou, Z. Dai, X. Zhuang, and X. Yao. Optimal design of medium channels for water-assisted rapid thermal cycle mold using multiobjective evolutionary algorithm and multi-attribute decision-making method. *International Journal of Advanced Manufacturing Technology*, 68:2407–2417, 2013.

[211] Y. M. Wang, Q. J. Liu, and T. Yu. A reinforcement learning approach to dynamic optimization of load allocation in AGC system. In *2009 IEEE Power and Energy Society General Meeting*, volume 1–8, pages 3704–3709, 2009.

[212] K. Weicker. An analysis of dynamic severity and population size. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. M. Guervs, and H.-P. Schwefel, editors, *PPSN*, volume 1917 of *Lecture Notes in Computer Science*, pages 159–168. Springer, 2000.

[213] R. L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *IEEE Transactions Evolutionary Computation*, 10(1):29–38, 2006.

[214] K. Willick, S. Wesolkowski, and M. Mazurek. Multiobjective evolutionary algorithm with risk minimization applied to a fleet mix problem. In *IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE, 2010.

[215] M. Wineberg and F. Oppacher. Enhancing the GAs ability to cope with dynamic environments. In *Proceedings of Genetic and Evolutionary Computation Conference D.Whitley et al., Eds.*, pages 3–10, 2000.

[216] L. Wu, K.L. Yick, S.P. Ng, and J. Yip. Application of the Box-Behnken design to the optimization of process parameters in foam cup molding. *Expert Systems Applications*, 39(9):8059–8065, 2012.

[217] M. C. Wu and W. J. Chang. A multiple criteria decision for trading capacity between two semiconductor fabs. *Expert Systems Applications*, 35(3):938–945, 2008.

[218] Y. Wu, Y. Wang, X. Liu, and J. Ye. Multi-population and diffusion UMDA for dynamic multimodal problems. *Journal of Systems Engineering and Electronics*, 21(5):777–783, 2010.

[219] Y. H. Wu. Multi-objective optimization design of vehicle clutch diaphragm spring. In *2nd Internationl Conference on Intelligent Computation Technology and Automation*, volume 3, pages 194–197, 2009.

[220] H. Xu, W. Liu, and H. Chen. Multiple objectives optimization problems in supply chain management under dynamic and stochastic circumstance. *Business and Information Management, International Seminar on*, 1:339–342, 2008.

[221] J. Xu, Y. Zhu, B. Jiang, and Z. Wang. Robust operation strategy design for an electronic market enabled supply chain with uncertain variable costs. In *ICMECG '09: Proceedings of the 2009 International Conference on Management of e-Commerce and e-Government*, pages 359–362, Washington, DC, USA, 2009. IEEE Computer Society.

[222] J. Yang, H. Xu, L. Pan, P. Jia, F. Long, and M. Jie. Task scheduling using bayesian optimization algorithm for heterogeneous computing environments. *Applied Soft Computing*, 11(4):3297–3310, 2011.

[223] S. Yang. Genetic algorithms with memory and elitism based immigrants in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 16(3):385–416, 2008.

[224] S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. *IEEE Transactions on Evolutionary Computation*, 12(5):542–561, 2008.

[225] Y. Yang and J.M. Lee. Probabilistic modeling and dynamic optimization for performance improvement and risk management of plant-wide operation. *Computers & Chemical Engineering*, 34(4):567–579, 2010.

[226] E. Yeguas, R. Joan-Arinyo, and M. V. Luzn. Modeling the performance of evolutionary algorithms on the root identification problem: A case study with PBIL and chc algorithms. *Evolutionary Computation*, 19(1):107–135, 2011.

[227] F. Yu, F. Tu, and K. R. Pattipati. Integration of a holonic organizational control architecture and multiobjective evolutionary algorithm for flexible distributed scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 38(5):1001–1017, 2008.

[228] X. Yu, Y. Jin, K. Tang, and X. Yao. Robust optimization over time - a new perspective on dynamic optimization problems. In *IEEE Congress on Evolutionary Computation*, pages 1–6. IEEE, 2010.

[229] B. Zeng, J. Wei, and J. Zhang. Optimal deployment strategy of sensing platform based on multi-objective genetic algorithm. *IEEE International Conference on Information and Automation*, 1–4:35–40, 2008.

[230] H. Zhang, Z. Jiang, and H. Hu. Multi-criteria dynamic scheduling methodology for controlling a semiconductor wafer fabrication system. In *IEEE International Conference on Automation Science and Engineering*, volume 3, pages 79–84, 2007.

[231] L. Zhang, L. Falzon, M. Davies, and I. Fuss. On relationships between key concepts of operational level planning. In *In Proceedings of 5th International Command and Control Research and Technology Symposium*, 2000.

[232] L. Zhang, L. Falzon, M. Davies, and I. Fuss. On relationships between key concepts of operational level planning. In *Proceedings of 5th International Command and Control Research and Technology Symposium*, 2000.

[233] Y. Zhang and X. Li. Estimation of distribution algorithm for permutation flow shops with total flowtime minimization. *Computers & Industrial Engineering*, 60(4):706–718, 2011.

[234] Y. Zhang and P. Rockett. Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection. In Hans-Georg Beyer and Una-May O'Reilly, editors, *GECCO*, pages 795–802. ACM, 2005.

[235] J. Zhou, L. Yu, S. Mabu, K. Hirasawa, J. Hu, and S. Markon. Service area-based elevator group supervisory control system using gnp with RL. In *SICE-ICASE International Joint Conference 2006, Bexco, Busan, Korea*, volume 1–13, pages 3028–3033, 2006.

[236] G. Zhu, J. Bard, and G. Yu. Disruption management for resource-constrained project scheduling. *Journal of the Operational Research Society*, 56:365–381, 2005.

[237] E. Zitzler. Evolutionary algorithms for multiobjective optimization: Methods and applications. Ph.D. dissertation, Swiss Federal Inst. Technol. (ETH), Zurich, Switzerland, 1999.

[238] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. C. Giannakoglou, D. T. Tsahalis, J. Periaux, K. D. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100. Int. Center for Numerical Methods in Engineering (Cmine), 2001.

[239] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257 – 271, 1999.

[240] E. Zitzler, L. Thiele, and K. Deb. Comparision of multiobjective evolutionary algorithms: Emprical results. *Evolutionary Computation*, 8(1):173–195, 2000.

[241] X. Zuo, H. Mo, and J. Wu. A robust scheduling method based on a multi-objective immune algorithm. *Information Science*, 179(19):3359–3369, 2009.