# Bio-Inspired Computing for Chance-Constrained Combinatorial Optimisation Problems

*Author*:
Yue Xie

*Supervisor*:
Prof. Frank Neumann

*Co-Supervisor*:
Dr. Aneta Neumann
Dr. Andrew M. Sutton

A thesis submitted for the degree of
DOCTOR OF PHILOSOPHY
The University of Adelaide

in the

Optimisation and Logistics
School of Computer Science

December 9, 2021

# Contents

# List of Figures

# List of Tables

University of Adelaide

# *Abstract*

**Bio-Inspired Computing for Chance-Constrained Combinatorial Optimisation Problems**

by Yue Xie

Bio-inspired methods have been widely used to solve stochastic optimisation problems, and they can find high-quality solutions. Motivated by real-world applications such as mining engineering problems where constraint violations have distributive effects, we discuss using chance-constrained optimisation to solve optimisation problems under various uncertainties. This thesis contributes to the theoretical analysis and application of evolutionary algorithms on chance-constrained combinatorial optimisation problems. We address complex problems under stochastic settings and are subject to chance constraints. We start our investigations in two significant areas: chance-constrained knapsack problem (CCKP) and a real-world application problem. The CCKP is a stochastic version of the classical knapsack problem, which aims to maximise the profit of selected items under a constraint that the knapsack capacity bound is violated with a small probability. We first show how to use well-known deviation inequalities as surrogate functions when tacking the chance constraint. Then, we investigate the performance of some classical approaches for solving knapsack problems and the simplest single- and multi-objective evolutionary algorithms on solving the CCKP instances. Our experimental results show that evolutionary algorithms perform better than those classical approaches in computation time and quality of solutions. Afterwards, to improve the performance of the evolutionary algorithms on solving the chance-constrained knapsack problems, we examine the use of two problem-specific operators and present a new multi-objective model of the problem. Our experimental results show that this leads to significant performance improvements when using the proposed operators in multi-objective evolutionary algorithms. Then, we perform a runtime analysis of a randomised search algorithm and a basic evolutionary algorithm for the chance-constrained knapsack problem with correlated uniform weights.

Furthermore, we investigate a real-world problem in this thesis, the stockpile blending problem, which aims to blend material from stockpiles to construct concentrate parcels containing optimal metal grades. For this problem, we first present the model of the problem without chance constraints and name it the "deterministic model" and then show that the results obtained by a differential evolution approach are better than the actual results for all instances. We then consider the problem with uncertain source supply, named the stockpile blending problem with chance constraints. We introduce chance-constrained optimisation to guarantee the constraints are violated with a small probability to tackle the stochastic material grades and evaluate the performance of the differential evolution algorithm.

# Declaration of Authorship

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Yue Xie

August 2021

# *Acknowledgements*

To take this opportunity, I am grateful to those who have offered encouragement and support during the three years of my PhD study.

First of all, my deepest gratitude to my supervisor, Professor Frank Neumann, for his constant encouragement and guidance. I am most grateful to him for inviting me into his team and supporting the scholarships that enabled me to focus on my study. He has walked me through all the stages of my PhD. Without his consistent and illuminating instruction, I could not have had such an inspiring journey.

I would also thank my two co-supervisors, Dr Aneta Neumann and Dr Andrew M. Sutton. They generously shared their advice, experience and feedback, which have been crucial for forming my understanding of research. They provided support whenever I needed aid and encourage me in both researching and teaching. I have immensely appreciated if working with both of them.

I want to express my thanks to the group members in the Optimisation and Logistics team at the School of Computer Science, the University of Adelaide, Bradley Alexander, Markus Wagner, María-Yaneli Ameca-Alducin, Wanru Gao, Mojgan Pourhassan, Junhua Wu, Vahid Roostapour, Maryam Hasani Shoreh, Hirad Assimi and other HDR students in our team. I had a great time working with them.

I want to thank the Research Consortia Program (RCP) "Unlocking Complex Resources through Lean Processing" to offer me the opportunity to collaborate with industry companies in mining engineering. I appreciate Professor Nigel Cook, Dr Tatiana Khmeleva, Professor Peter Dowd, Mrs Natalie Craven and other fellow researchers in the program. I learned a lot in sharing ideas with people from different backgrounds and collaborations with others.

Moreover, I would like to thank the School of Computer Science of the University of Adelaide and Research Consortium for funding the scholarships that have covered my tuition fees, living expenses, and conference travels; researchers who have contributed to the foundation of this domain in which I can study; the anonymous reviewers of my papers as well as this thesis; Joe Miller for his language editing and proofreading of this thesis.

Finally, I offer my wholehearted gratitude to my parents Hong Yu and Dongsheng Xie, and my dear families and friends. Without your understanding and support, I would not have been able to go this far. I also want to thank my mother city Wuhan, without her survival from the terrible Covid-19, I would not have been able to release the stress and focus on my research.

# Chapter 1

# Introduction

Bio-inspired computing methods such as evolutionary algorithms are well-known general problem-solvers that are applied to various problems. These algorithms are inspired by processes that occur in nature. For example, evolutionary algorithms (EAs) are inspired by processes that make creatures suited to their environment during generations. The main idea of bio-inspired methods is to randomly generate a number of solutions as the initial population, evolve them by using random operators and replace the worst solutions of the population with new solutions that are better in terms of fitness. The variation operators that generate new solutions from old solutions are crossover and mutation. Although bio-inspired methods such as evolutionary algorithms (EAs) and genetic algorithms (GAs) are not able to guarantee optimal solutions, they usually achieve high-quality solutions in a reasonable computational time. During last decades, bio-inspired algorithms have been used very successfully for a wide range of problems, such as combinatorial optimisation problems (Neumann and Witt, 2010; Chiong, Weise, and Michalewicz, 2012; Sim and Kaufmann, 2018).

Combinatorial optimisation problems are a kind of problem that have a finite number of feasible solutions. A combinatorial optimisation problem can be expressed as a minimisation problem or a maximisation problem, which depends on a given objective function and is subject to a set of constraints. These problems include a variety of theoretical and real-world applications with different levels of computational complexity. Many combinatorial optimisation problems are NP-hard problems and need superpolynomial time to be solved by a deterministic algorithm unless $P = NP$. Bio-inspired algorithms can explore the search space of complex problems and, therefore, achieve with a better solution tat employs random sampling of the solution space with no guarantee of ever proving optimality of any solution.

The complexity of many combinatorial problems comes from their objective functions. However, in real-world applications, another challenge is to deal with the uncertainty of input parameters. If we ignore the uncertainty, for example, by using the expectations of input parameters, the optimal solution obtained is likely to be infeasible. To consider the uncertainty, we apply chance-constrained optimisation in this thesis. Chance-constrained optimisation was first introduced by Charnes and Cooper (1959) and Miller and Wagner (1965), whose resulting decision ensures the probability of respecting to the constraints or the confidence level of being feasible to have received significant attention in the literature. For a general chance-constrained problem, Prékopa (1990) and Prékopa (1995) proposed a dual-type algorithm, and they investigated the performance of their approaches and compared them with a primal simplex algorithm. Hillier (1967) used linear constraints to generate a procedure for tacking approximate chance constraints. Chance-constrained optimisation has been widely applied in different disciplines for optimisation under uncertainty (Uryasev, 2013). For example, chance-constrained optimisation has been used in analogue integrated circuit design (McConaghy et al., 2009), mechanical engineering (Mercado et al., 2005), and other disciplines (Liu, 2007; Poojari and Varghese, 2008). Some details of the chance-constrained optimisation are given in Chapter 2.

## 1.1 Contributions and Background

Motivated by real-world problems where constraint violations have disruptive effects, we conduct research in this thesis on chance-constrained optimisation and its real-world applications. Moreover, a wide range of studies has investigated the performance of evolutionary algorithms in practice for their ability that can easily be applied to solve various problems. However, so far, chance-constrained optimisation has received little attention in the evolutionary computation literature (Liu et al., 2013).

In this thesis, we investigate the chance-constrained knapsack problem (CCKP), a variant of the binary knapsack problem. In general, the weights of items are not known precisely in a CCKP, but all weights are chosen from a given distribution. Each item has a stochastic weight and a deterministic profit. The goal of the CCKP is to find a set of items of maximal profit subject to the condition that the probability with which the total weight exceeds the given capacity bound is less than or equal to a given threshold. Several research papers that study the stochastic knapsack problem by using chance-constrained optimisation have been published in the literature (Goel and Indyk, 1999; Kleinberg, Rabani, and Tardos, 1997; Klopfenstein and Nace, 2008; Goyal and Ravi, 2010). Goel and Indyk (1999) proposed an algorithm that relaxes the chance constraints by a factor of $(1 + \epsilon)$ to solved instances where weights have a Poisson distribution or an exponential distribution. Goyal and Ravi (2010) investigated the problem where the weights of items have the Normal distribution, and the proposed linear optimisation approach can satisfy the chance constraint strictly.

Evolutionary algorithms are suitable for solving various combinatorial optimisation problems (Chiong, Weise, and Michalewicz, 2012; Azzouz, Bechikh, and Said, 2017). Han and Kim (2002) presented a quantum-inspired evolutionary algorithm and showed it performs well for knapsack problems. Segura et al. (2015) proposed a novel memetic scheme incorporating a new diversity-based replacement strategy and applied the algorithm to a large instance of travelling salesmen problem. Riff et al. (2008) introduced an evolutionary algorithm for solving copper mine planning problems and demonstrated that this algorithm could find better feasible solutions than the approach that has been used before. Therefore, we investigate the performance of EAs in solving the CCKP. To evaluate a solution concerning the chance constraint, we use well-known probability tools to calculate an upper bound on the probability of violating the capacity bound and as surrogate functions for the chance constraint. The probabilistic tools we employ allow us to estimate the probability of a constraint violation mathematically without the need for sampling. Moreover, we develop a single-objective approach and a multi-objective approach in terms of solving CCKP instances, and we compare the performance of several EAs to presented deterministic methods. The comparison provides a reasonable justification for using EAs when dealing with the CCKP. Therefore, we introduce a problem-specific crossover operator and a heavy-tail mutation operator applied in EAs to improve their performances.

The critical feature of EAs is their use of random operators to explore the search space, and theoretical analyses of random processes are challenging. The theoretical analysis is important to understand the characteristics and behaviours of EAs (Papadimitriou and Steiglitz, 1982). In the past few decades, many researchers have worked to improve the theoretical understanding of algorithms. Many studies first focus on simple example functions (He and Yao, 2001; Droste, Jansen, and Wegener, 2002), leading to analyses about combinatorial optimisation problems (Neumann and Wegener, 2007; Oliveto, He, and Yao, 2007). Useful tools for algorithm complexity analysis include fitness-based partitioning (Wegener, 2002), deviation inequalities (Raghavan and Motwani, 1995) and drift analyses (He and Yao, 2001; Doerr, Johannsen, and Winzen, 2012). Recently, Neumann and Sutton (2019) investigated the runtime of the (1+1) EA for the CCKP and proved that it is possible to have local optimal in the search space even in the simplest case. However, the problem still has not gained much attention in the theoretical analysis of evolutionary algorithms literature.

In Chapter 5, we analyse the expected optimisation time of a proposed randomised local search (RLS) and the (1+1) EA in the chance-constrained knapsack problem with correlated weights. This variant partitions the set of items into groups, and pairs of items within the same group have correlated weights. We prove bounds on both the time to find a feasible solution and the time to obtain the optimal solution, which has both maximal profit and minimal probability of violating the chance constraint. In particular, we first prove that RLS can find a feasible solution in time bounded by $O(n \log n)$, and by $O(n^2 \log n)$ in terms of the (1+1) EA. Then, we investigate the optimisation time for these algorithms when the profit values are uniform, which is similar to the study in the deterministic constrained optimisation problems (Friedrich et al., 2020). However, the items in our case are divided into different groups, and it becomes necessary to take the number of chosen items from each group into account. Therefore, the optimisation time bound for RLS becomes $O(n^3)$ and $O(n^3 \log n)$ for the (1+1) EA. After that, we consider the more general and complicated case in which profits may be arbitrary as long as each group has the same set of profit values. We show that an upper bound of $O(n^3)$ holds for RLS and $O(n^3(\log n + \log p_{max}))$ holds for the (1+1) EA where $p_{max}$ denotes the maximal profit among all items.

Due to their ability to provide high quality solutions for complexity and dynamic problems within acceptable time, EAs have attracted interest from many researchers and practitioners. Chiong, Weise, and Michalewicz (2012) presented how to incorporate evolutionary algorithms in solving optimisation problems in the fields of planning and scheduling, engineering and data science. To empirically investigate the applicability of EAs to complex real-world problems, we investigate in Chapter 6 the stockpile blending problem, which is a crucial component of open-pit mine production scheduling (OPMPS) problem in mining engineering (Lamghari and Dimitrakopoulos, 2012; Moreno et al., 2017; Sotoudeh et al., 2020). In this problem, stockpiles are used to store and blend raw material transported from mines. The stockpile blending problem aims to construct parcels of concentrate by combining material from stockpiles and contain optimal metal grades based on the material available in stockpiles. The volume of material each stockpile provides to each parcel depends on a set of mine schedule conditions, mill-feed limitation, and customer demands. Due to the complexity of the problem, we formulate the problem as a non-linear optimisation problem with continuous decision variables. We first investigate the large-scale stockpile blending problem. As can be inferred from its name, the problem is characterised by challenges relating to scale. To the best of our knowledge, the large-scale stockpile blending problem has never been studied independently, although it is vital in real-world mining engineering. A realistic model of the stockpile blending problem is introduced in this thesis and follows by describing related input parameters of the production process in a real-world situation. To tackle the complexity constraints in the model, we present two repair operators and apply them in the solver algorithm later. We introduce an approach based on a differential evolution (DE) algorithm for the large-scale stockpile blending problem and investigate the method's performance by examining real-data instances. The DE algorithm is an evolutionary algorithm that facilities a population-based search in continuous multi-dimensional space. DE algorithms were first proposed by Price (1996) and Storn and Price (1997), after then, DE and its variants have been successfully applied to solve many practical problems from different scientific and engineering fields (Das and Suganthan, 2010; Neri and Tirronen, 2010). We show that this DE-based approach improves the results of real-world cases.

Motivated by the uncertainty in geologic input data which affects optimisation, we investigate the stockpile blending problem by taking uncertainty into account. It is recognized in the relevant technical literature that this uncertainty is the main reason for not meeting the production expectations (Baker and Giacomo, 1998; Asad and Dimitrakopoulos, 2012). Given its significant impact on the financial results of mining operations, we focus on the uncertainty of metal content within a mineral deposit being mined. For the stochastic variables of the stockpile blending problem, we apply the chance-constrained optimisation to tackle the uncertainty of material grades by reformulating inequality constraints to chance constraints. The objective of the stockpile blending problem with chance constraints is the same as the large-scale stockpile blending problem, and it is subject to the condition that the probability of violating a given bound is less than a given threshold. Moreover, we investigate how to solve the problem using DE algorithms and evaluate the effectiveness of different chance constraints.

## 1.2 Outline of the Thesis

This thesis is composed of the following three main parts: introduction and background of the thesis, evolutionary algorithm implementation and real-world applications of chance-constrained optimisation. The rest of this paper is organised as follows.

Chapters 2 and 3 briefly introduce the background of the research in this thesis. In Chapter 2, OneMax problem and knapsack problem as two well-known combinatorial optimisation problems will be introduced in detail. Including background knowledge about chance-constrained optimisation, which laid the foundation for the thesis. The last presented problem of Chapter 2 is the chance-constrained knapsack problem that will be discussed in detail in the second part of the thesis. Chapter 3 consists of an introduction about the bio-inspired algorithms and the analytical methods that are used in this thesis.

Chapter 4 and 5 focus on evolutionary algorithms for solving the chance-constrained knapsack problem. We introduce how to incorporate well-known probability tail inequalities into the search process of an evolutionary algorithm in Chapter 4. We introduce single-objective and multi-objective evolutionary algorithms for the CCKP and develop operators to improve the performance of evolutionary algorithms. We examine the different performances of different evolutionary algorithms combined with different operators for the CCKP. The comparison between the deterministic approaches and the evolutionary algorithms shows that using evolutionary algorithms for solving CCKP instances leads to higher quality results in many test cases regarding the running time. This chapter is based on two conference papers (Xie et al., 2019; Xie, Neumann, and Neumann, 2020). In Chapter 5, we present our investigation on the behaviour of evolutionary algorithms and randomised local search optimising CCKP. The main contribution of this chapter is that it provides some insights into the theoretical understanding of CCKP with relevant weights through rigorous runtime analysis. This chapter is expanded from work published in Xie et al. (2021). After theoretical analysis, our research has turned to real-world application areas.

Chapter 6 present a real-world problem named the stockpile blending problem. As an important component in the mining engineering problem, we define the problem as a non-linear optimisation problem in a continuous search space. We introduce two repair operators used in bio-inspired algorithms to tackle the complex constraints by guiding the searching process. Then, we propose an approach for solving large-scale stockpile blending problems based on a differential evolution algorithm. The DE-based approach can solve real-data instances of the large-scale stockpile blending problem, and we show that it outperforms the realistic method. The contents are based on the published paper (Xie, Neumann, and Neumann, 2021a).

Motivated by the uncertainty in geology data, the stockpile blending problem with chance constraints is extended from the deterministic setting of the problem and is detailed in Chapter 7. We consider the stockpile blending problem with uncertainty in material grades and denote them as stochastic variables. We apply chance-constrained optimisation to handle the inequality constraints of the problem, and investigate the performance of differential evolution algorithms. This chapter is based on the result published by Xie, Neumann, and Neumann (2021b). In the end, Chapter 8 concludes the thesis.

# Chapter 2

# Combinatorial Optimisation Problems and Chance-Constrained Optimisation

## 2.1 Introduction

Combinatorial optimisation is one of the most active fields in Applied Mathematics and is widely used in industry and commerce, which combining technology from combinatorics, linear programming and algorithm theory (Du and Pardalos, 1998). A combinatorial optimisation problem consists of finding the best feasible solution according to the objective function when the problem's solution space is discrete, and the feasibility is determined by satisfying given constraints. Due to the successful application of combinatorial optimisation in solving complex problems in many real-world application areas, this field has gained more attention and interest (Chiong, Weise, and Michalewicz, 2012; Shambour, 2019). This chapter provides a general overview of combinatorial optimisation and introduces two classical problems in Section 2.2.

Motivated by real-world problems where constraint violations have disruptive effects, we present the chance-constrained optimisation, which has been widely applied in different disciplines for optimisation under uncertainty. In the second part of this chapter, we introduce a general overview of chance-constrained optimisation and define the chance-constrained knapsack problem in Section 2.3. The chance-constrained knapsack problem is the core problem discussed in this thesis presented in Chapter 4 and 5.

## 2.2 Combinatorial Optimisation Problems

Many applications can be abstracted as combinatorial optimisation problems. For any instance of a problem, a specified parameter setting is given. Formally, the definition of a combinatorial optimisation problem is given a triple $(S, f, \Omega)$, where $S$ is the search space, $f$ is the objective function, and $\Omega$ is the set of constraints, the goal is to find a globally optimal solution in $S$ with respect to $f$ that fulfils all constraints given by $\Omega$ (Neumann and Witt, 2010).

The optimisation time of the algorithm is analysed according to the size of the input. The input of combinatorial optimisation problem is usually a graph or a set of integers and can be expressed as a series of symbols. The size entered is equal to the length of this sequence. The search space of a combinatorial optimisation problem is exponential with respect to the size of the input, and for most combinatorial optimisation problems, a polynomial-time algorithm that finds the optimal solution can not be found unless $P = NP$.

In the following subsection, we introduce two well-known combinatorial optimisation problems, Knapsack problem (KP) and OneMax problem.

### 2.2.1 Knapsack Problem

The knapsack problem (KP) is a well-known combinatorial optimisation problem that has been studied extensively in the past decades. Toth and Martello (1990) formulated the knapsack problem as a set of $n$ items and a knapsack. Each item has a weight $w_i$ and a profit $p_i$. The knapsack problem aims to find a subset of the $n$ items that maximises the total profit, and the total weight of this subset must not exceed the knapsack capacity. The definition of a KP is as follows:

$$\max \quad P(x) = \sum_{i=1}^{n} p_i x_i \tag{2.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i x_i \leq C, \tag{2.2}$$

$$x_i \in \{0,1\}, i = 1, \ldots, n. \tag{2.3}$$

Here $x = (x_1, \ldots, x_n)$ represents the packing plan of the item set, and the item $i$ is chosen when $x_i = 1$, and vice versa. As $x_i$ is binary, this problem is so called the binary knapsack problem.

The binary knapsack problem has been thoroughly studied by Toth (1980), Vance (1993), Martello and Toth (1997), Martello and Toth (2003), and Kellerer, Pferschy, and Pisinger (2004). This work mainly concentrates on presenting exact algorithms to tackle the binary knapsack problem. The exact approaches for solving binary knapsack problems can be divided into two classes: dynamic programming (Ahrens and Finke, 1975; Martello, Pisinger, and Toth, 1999) and branch and bound methods (Kolesar, 1967; Greenberg and Hegerich, 1970). We give the detail of using dynamic programming to solve the binary knapsack problem in the next section.

#### Dynamic Programming

Dynamic programming was first introduced by Bellman (1966), and the first dynamic programming is based on the following idea. If $x_n = 0$, that is we do not choose the last item, then the best profit possible is obtained by whatever is attained from the remaining $n-1$ objects with knapsack capacity $C$. If $x_n = 1$, then the best profit possible obtained is the profit $p_n$ plus the best profit obtained by the remaining $n-1$ items with knapsack capacity $C - w_n$, where $w_n \leq C$. Therefore, the optimal profit will be the maximum of these two profits.

This idea leads to the following recurrence relation,

$$P(i,m) = \begin{cases} P(i-1,m) & w_i > m \\ \max\{P(i-1,m), P(i-1,m-w_i) + p_i\} & w_i \leq m \end{cases} \tag{2.4}$$

**Algorithm 1:** Dynamic Programming for Binary Knapsack Problem

```
1:  for $w = 0$ to $C$ do
2:      $P(0, m) = 0$
3:  end for
4:  for $i = 1$ to $n$ do
5:      for $m = 0$ to $C$ do
6:          if $w_i \leq w$ then
7:              $P(i, m) = \max\{P(i-1, m), P(i-1, m-w_i) + p_i\}$
8:          else
9:              $P(i, m) = P(i-1, m)$
10:         end if
11:     end for
12: end for
13: return $P(n, C)$
```

where $P(i,m); i = \{1, \ldots, n\}, m = \{0, \ldots, C\}$, is the best profit obtainable form the items $1, \ldots i$ with capacity $m$. The initial conditions are

$$P(1, m) = \begin{cases} 0 & w_1 > m \\ p_1 & w_1 \leq m. \end{cases} \tag{2.5}$$

So, the optimal profit is expressed as $P(n, C)$.

In general, a dynamic programming has to construct an $n \times C$ table and calculate the entries $P(i, m)$, in a bottom-up fashion. When the optimal profit $P(n, C)$ has been calculated, the optimal solution can be found by backtracking through the table and assigning the decision variables to the solution according to the selections of the max function. The algorithm has a time complexity of $O(nC)$. However, it is not polynomial, since $W$ can be exponentially large, such a complexity is called pseudo-polynomial. The dynamic programming for binary knapsack problem is presented in Algorithm 1.

### 2.2.2 OneMax Problem

If the capacity constraint is removed, all subsets of items are feasible and the profit of a solution $x$ can be expressed as

$$f(x) = w_0 + \sum_{i=1}^{n} w_i x_i, \tag{2.6}$$

which is a linear pseudo-Boolean function. Here, $x = \{x_1, x_2, \ldots, x_n\}$ be a search point in search space $\{0, 1\}^n$, and $w_i; i = \{1, \ldots, n\}$ positive real weight. If we set $w_0 = 0$ and optimisation of a linear objective function under a linear constraint, the problem is equivalent to the binary knapsack problem (Koza, 1993).

The OneMax problem, as a specific case of a linear pseudo-boolean function with weights equal to one, unlike the general knapsack problem, the OneMax problem is trival and is only of interest for theoretical analysis of evolutionary algorithms, which we introduce in Chapter 3. The OneMax problem consists of $n$ binary variables, and it aims to maximise the number of 1's among those variables. A solution is represented by a bit-string with length $n$, e.g. $x = \{x_1, \dots, x_n\}$ . The problem is defined as

$$OneMax(x) = \sum_{i=1}^{n} x_i.$$

## 2.3 Chance-Constrained Optimisation

In real-world applications, uncertainty is a natural attribute of a complex system, and it will have a significant impact on the system's performance. The goal of optimisation under uncertainty is to provide economical and reliable decisions for problems with such uncertainties. To tackle the formulation of optimisation problems under uncertainty, we introduce the chance-constrained method. Its resulting decision ensures the probability of complying with the constraints and the confidence level of being feasible to have received significant attention in the literature.

Consider the following optimisation problem:

$$\begin{aligned} \min \quad & g(x) && (2.7)\\ \text{s.t.} \quad & c(x, \xi) \leq 0, && (2.8)\\ & x \in X, && (2.9) \end{aligned}$$

where $X \subseteq \mathbb{R}^m$ is a deterministic set, $x$ is a decision vector, $\xi$ is a multi-dimensional parameter vector, $g(x)$ is a objective function and $c(x, \xi)$ is a constraint. Moreover, we assume that $g(x)$ and $c(x, \xi)$ are convex in $x$, and $X$ is a compact and convex set. This kind of problem has broad application in many real-world problems.

However, in practical problems, the parameter $\xi$ can be uncertain. If we solve the problem using the expected value of $\xi$ without considering the uncertainty, the optimal solution obtained under this assumption might be infeasible with high probability. Therefore, to consider the uncertainty, the problem can be formulated as a chance-constrained optimisation problem as follows:

$$\begin{aligned} \min \quad & g(x) && (2.10)\\ \text{s.t.} \quad & P_r\left(c(x, \xi) \leq 0\right) \geq \epsilon, && (2.11)\\ & x \in X, && (2.12) \end{aligned}$$

where $\xi$ is a random vector and $\epsilon \in (0, 1)$ is the tolerance probability. Then, the solution of this chance-constrained optimisation problem is guaranteed to be a feasible solution to the original problem with a probability of at least $\epsilon$, say 0.9 or 0.99. The chance constraint (2.11) can also be expressed as

$$P_r(c(x, \xi) > 0) \leq \alpha,$$

where $\alpha \in (0, 1)$ is a small acceptable probability, say 0.01 or 0.001. As shown in inequality (2.11), there is only one chance constraint in this model. We call this problem an individual chance-constrained problem.

If there is more than one chance constraint in the problem, the problem is called a joint chance constraint problem, we show the it as follows:

$$\min \quad g(x) \tag{2.13}$$
$$\text{s.t.} \quad P_r\left(c_1(x,\xi) \le 0, \ldots, c_d(x,\xi)\right) \ge \epsilon, \tag{2.14}$$
$$x \in X. \tag{2.15}$$

However, the joint chance-constrained problem is obviously more difficult to solve, mainly because it considers the multivariate distribution.

Due to two main reasons, chance-constrained optimisation problems are usually difficult to solve. First of all, calculating $P_r(c(x,\xi) \le 0)$ is usually difficult, especially for joint chance-constrained problems. Secondly, the feasible region defined by the chance constraint is usually not convex (Ahmed and Shapiro, 2008; Luedtke, Ahmed, and Nemhauser, 2010). Therefore, the solution method depends on the characteristics of the chance-constrained problem. Some problems will not arise in any of these difficulties. For example, when $\xi$ is a multivariate normal vector, Prékopa (2003) gives the deterministic equivalent of the right hand side of the chance constraint (2.11). Then, the deterministic equivalent of the chance constraint problem is given in the paper.

If the solution space is not convex, $\xi$ has a finite distribution (Dentcheva, Prékopa, and Ruszczynski, 2000). Prékopa (1971) has taken an important step by proving the convexity of the feasible solution set of the linear chance-constrained problem and the related random uncertainty in the chance constraint. To extend their results on convexity, Prékopa, Yoda, and Subasi (2011) make a statement about their proposed structural description applying more widely. He proposes that if the rows are independent normal distributions and the covariance matrices of the rows are constant multiples of each other. Henrion (2007) proposes a structural description of the feasible solution set defined by a single linear chance constraint.

For problems in which the difficulty of computing the probability occurs, the probability of chance constraints is difficult to compute due to the requitement of solving multiple integrals. Many equivalent reformulations for chance constraints have been proposed to handle this issue. Lagoa (1999) and Calafiore and El Ghaoui (2006) considered individual chance-constrained linear programs and gave deterministic quadratic reformulations. For the case in which chance constraints have continuous distributions, sampling approximations are used in solving chance-constrained problems, such as the case in which joint chance constraint with a feasible convex region (Ahmed and Shapiro, 2008), and the case in which uncertainty is represented with discrete distributions (Ruszczyński, 2002). However, sampling approximations can be computationally very demanding.

An alternative to the approximated approaches mentioned above includes an approximation based on chance constraints to provide a deterministic boundary analysis, which we also use in this thesis. For the case of an individual chance constraint, the boundary is mainly based on the expansion of Chebyshev's inequality (Pinter, 1989; Hoeffding, 1994; Birge and Louveaux, 2011). For joint chance constraints, the deterministic equivalent approximation has been extensively studied by Cheng and Lisser (2012), Cheng and Lisser (2013), and Cheng, Houda, and Lisser (2015). We will introduce the details in Section 2.3.2.

### 2.3.1 Chance-Constrained Knapsack Problem

The binary knapsack problem introduced in Section 2.2.1, and different variants of the knapsack problem have been examined in the stochastic setting. The chance-constrained knapsack problem (CCKP) is one kind of stochastic version of the traditional knapsack problem.

In this thesis, we consider the case of CCKP, where the weight of each item is not known exactly, but all weights are chosen from a given distribution. Given $n$ items, and each item has weight $w_i$ and profit $p_i$, and a knapsack capacity $C$. We encode a solution $x$ as a bit string of $\{0,1\}^n$, and $x_i = 1$ denotes selecting $i$-th item. The problem aims to find a subset of items with maximised profit, and the probability of violating the capacity bound is less than or equal to a given threshold denoted as $\alpha$. The chance-constrained knapsack problem can be formulated as follows:

$$\max \quad P(x) = \sum_{i=1}^{n} p_i x_i \tag{2.16}$$

$$\text{s.t.} \quad P_r\left(\sum_{i=1}^{n} w_i x_i \geq C\right) \leq \alpha \tag{2.17}$$

$$x_i \in \{0,1\}. \tag{2.18}$$

The chance-constrained knapsack problem has been studied in several published papers. Kleinberg, Rabani, and Tardos (1997) studied the problem with the weights of items that are only chosen from two possible options. Goel and Indyk (1999) proposed an algorithm that relaxes the chance constraints by a factor of $(1+\epsilon)$ to solved instances where items have a Poisson distribution, or an experimental distribution.

### 2.3.2 Related Work

In general, it is difficult to calculate the probability of chance constraints. Therefore, many equivalent reconstructions of the chance constraint or its approximation have been proposed. Jagannathan (1974) and Calafiore and El Ghaoui (2006) introduced to reformulated a single linear chance-constrained problem as a second-order conic programming (SOCP) constraint, where the random vector $\xi$ in the problem is chosen according to the Normal distribution, elliptic distribution or radial distribution. Cheng and Lisser (2012) studied the problem where chance constraints have normal distribution coefficients and independent matrix rows. A SOCP approximation using piecewise linear and piecewise tangent approximation is proposed by Luedtke, Ahmed, and Nemhauser (2010). When the random vector support is limited, construct a mixed-integer linear programming reconstruction for the joint linear chance constraint problem. Hillier (1967) uses linear constraints to generate programs that handle approximate chance constraints.

Furthermore, chance-constrained optimisation has been widely applied in different disciplines for optimisation under uncertainty (Uryasev, 2013). For example, a chance-constrained method has been applied in analogue integrated circuit design (McConaghy et al., 2009), mechanical engineering (Mercado et al., 2005). Bhattacharya (2009) presented a chance-constrained optimisation model, which has been designed to decide the number of advertisements in different advertising media and the optimal allocation of the budget assigned to the various media. They assumed that the parameter corresponding to connect for different media as random variables could be considered values with known mean and standard deviations. Wang, Tang, and Fung (2014) designed a chance-constrained stochastic programming model for daily operating room planning. The model combines opportunity constraints to enforce the upper limit of the risk of canceling surgery due to exceeding the ability to work overtime. Ravichandran et al. (2018) reformulated the optimisation problem that an online optimal control strategy for power flow management in microgrids as a stochastic chance constraints optimisation problem. They highlighted the significant improvement of this model in the system robustness over conventional rolling horizon controller while dealing with uncertainties in the predictions of generation. Ordoudis et al. (2021) developed a two-stage stochastic plan for the energy and reserve scheduling of the joint power and natural gas system, and adopted opportunity constrained optimisation to ensure that there is no high probability of load shedding and renewable resource spillover.

Until now, chance-constrained optimisation has not gained much attention in the research field of evolutionary computation. The content in Chapter 4 first considers the chance-constrained optimisation to tackle the stochastic variables in a combinatorial optimisation problem that we introduce in the next section. Neumann and Sutton (2019) first analysed the runtime of the most straightforward evolutionary algorithm for the chance-constrained knapsack problem. Neumann and Neumann (2020) presented a first runtime analysis of evolutionary multi-objective algorithms for chance-constrained submodular functions. Assimi et al. (2020) considered the chance-constrained knapsack problem under a dynamic environment and introduced an additional objective used in multi-objective evolutionary algorithms. Doerr et al. (2020) provided a first analysis on the approximation behaviour of popular greedy algorithms for submodular problems with chance constraints.

For an overview of the theory, solution and applications of chance-constrained optimisation problems, one can refer to the monographs (Kall, Wallace, and Kall, 1994; Calafiore and Dabbene, 2006; Liu, 2007; Shapiro, Dentcheva, and Ruszczyński, 2014).

## 2.4 Conclusion

In this chapter, we briefly introduce combinatorial optimisation problems and chance-constrained optimisation problems. Since many complex real-world problems can be expressed as combinatorial optimisation problems, this leads to greater attention of researchers in this research field. We presented the formal definitions of two of the most well-known problems: the OneMax problem and the knapsack problem. Next, we introduced chance-constrained optimisation, one of the major approaches to solving optimisation problems under various uncertainties. Finally, we presented the chance-constrained knapsack problem, which will be deeply studied in Chapter 4.

Chapter 3

# Bio-Inspired Computing and Analytical Methods

## 3.1 Introduction

Methods inspired by biology such as local search, evolutionary algorithms and genetic algorithms have been proved suitable for solving various combinatorial optimisation problems (Hoos and Stützle, 2004; Johannsen, 2011). They are also part of a broader class of stochastic search algorithms. Generally, bio-inspired algorithms iteratively search for better solutions and employ heuristics that involve randomness, and they are widely used for solving NP-hard problems due to their ability to provide high-quality solutions for problems in reasonable computation times. Bio-inspired methods usually start with an initialised solution $x$ that is randomly generated or pre-designed. Then, a fitness function $f : D \to \mathbb{R}$ is defined, where $D$ is the search space, and the algorithms explore the $D$ to find another solution $y$ with a better fitness value than $x$. These algorithms repeat the search procedure iteratively to find a solution $x^*$ with optimal fitness value.

While stochastic search algorithms are developed, the theoretical analysis of their behaviour is still far behind their practical implementation. The area of runtime analysis for bio-inspired computing techniques starts with the first runtime analysis of an evolutionary algorithm, given by Mühlenbein (1992). After then, this area of research has provided many insights into the working behaviour and process of bio-inspired computing methods when solving combinatorial optimisation problems (Neumann and Witt, 2006; Neumann and Witt, 2010; Auger and Doerr, 2011; Oliveto and Yao, 2011). The computational analysis of these algorithms considers the number of evaluations that is required to find an optimal solution concerning the size of input parameters and plays a significant role in their theoretical understanding.

This chapter is organised as follows. In Section 3.2 and Section 3.3, we introduce Randomised Local Search and the main concepts of evolutionary algorithms, respectively. In Section 3.4, we present the analytical methods which are commonly used to analyse the performance of bio-inspired algorithms.

## 3.2 Randomised Local Search

Local search is a widely used heuristic method for solving optimisation problems. It is an iterative algorithm and starts with an initial solution (randomly selected or initialised by other algorithms). As the name suggests, the algorithm considers the local neighbourhood $N(x)$ of the current solution $x$, and searches for a better

**Algorithm 2:** Local Search

```
1: The initial solution $x$ and neighbourhood function $N(x)$ are given;
2: while stopping criterion not met do
3:     Replace $x$ with the better solution $y \in N(x)$.
4: end while
```

**Algorithm 3:** RLS$_1$

```
1: The initial solution $x$ is given;
2: while stopping criterion not met do
3:     $y \leftarrow$ flip one bit of $x$ chosen uniformly at random;
4:     if $f(y) \geq f(x)$ then
5:         $x \leftarrow y$
6:     end if
7: end while
```

quality solution $x'$ for the fitness function (see Algorithm 2). Note that an appropriate neighbourhood function is required to define the neighbourhood. When the $x'$ is found, the algorithm replaces $x$ by $x'$, then the search continues, and $x'$ is the next solution. The process of searching the neighbourhood of $x$ for a better neighbour starts over. The local neighbourhood of a search point needs to be predefined and should not be too large. The algorithm stops when it fails to find improvements in the neighbourhood of the current solution. At this point, either the optimal solution is found, or the algorithm is stuck at a local optimum.

Randomised local search (RLS) is one of the simplest stochastic search algorithms. Considering the current solution $x$, RLS chooses one solution $y$ at each iteration in the neighbourhood of $x$ at random. The current solution $x$ is replaced by $y$ if the fitness value of $y$ is at least as fit as $x$. The process continues until RLS can not find a better solution within a pre-set number of iterations. Note that the size of the neighbourhood plays a vital role in this algorithm.

A too-small neighbourhood results in fast convergence to a locally optimal solution, while a large neighbourhood makes it possible to choose a new solution that is very different from the current solution. Therefore, the algorithm can not guide the search properly. A standard RLS algorithm (RLS$_1$) is presented in Algorithm 3. Starting with an initial solution $x$, RLS$_1$ creates a new solution $y$ by flipping exactly one bit of $x$ uniformly at random and replacing $x$ with if the new solution is superior or equal in terms of fitness. The algorithm repeats these steps until the desired condition is satisfied. The most useful desired condition is exceeding the maximum number of fitness evaluations that the algorithm is allowed to perform. The RLS$_1$ considers a neighbourhood comprised of solutions with unit Hamming distance, and it only flips a single bit in each iteration.

Here we introduce another kind of RLS, named RLS$_2$ (see Algorithm 4). This algorithm starts with a randomly initialised solution and iteratively improves it by applying a series of mutations. In each mutation step, it applies either *one-* or *two-bit* mutation with equal probability. Specifically, with probability $1/2$, it selects a single index uniformly at random from $\{1, \dots, n\}$ and flips the corresponding bit in the current solution. Otherwise, it selects two distinct indexes uniformly at random to flip.

**Algorithm 4:** RLS$_2$

```
1: Choose $x \in \{0,1\}^n$ to be a decision vector.
2: while stopping criterion not met do
3:     Choose $b \in \{0,1\}$ uniformly at random.
4:     if $b = 0$ then
5:         choose $i \in \{1, \dots, n\}$ uniformly at random and define $y$ by flipping the $i$th
           bit of $x$.
6:     else
7:         choose $(i,j) \in \{(k,l) \mid 1 \leq k < l \leq n\}$ uniformly at random and define $y$ by
           flipping the $i$th and the $j$th bit of $x$.
8:     end if
9:     if $f(y) \geq f(x)$ then
10:        $x \leftarrow y$ ;
11:    end if
12: end while
```

## 3.3 Evolutionary Algorithms

Evolutionary algorithms (EAs) are a random search algorithms, which use the mechanism of biological evolution, such as reproduction, mutation, recombination and selection. The candidate solution of the optimisation problem is the individual in the population. EAs prioritises individuals with high fitness values as potential parents of the next generation according to the fitness function. Then, after repeated application of the above operators, the population evolves.

Evolutionary Algorithms (EAs) have shown to be very successful when applied to combinatorial optimisation problems, in particular, in tackling combinatorial NP-hard optimisation problems (Deb, 2001; Chiong, Weise, and Michalewicz, 2012). EAs often perform well approximating solutions because they ideally do not make any assumptions about the underlying fitness landscape. Evolutionary algorithm provides a framework, which makes it easier to combine prior knowledge about the problem. Combining this information, we will focus on evolutionary search, so as to explore the state space of possible solutions more effectively. In the following sections, we describe the main components of evolutionary algorithms and the terminology used for referring to essential concepts of these methods.

### 3.3.1 Components of Evolutionary Algorithms

Now we describe the main issues to be addressed properly before performing an evolutionary algorithm on a given problem.

#### Representation

The first step of using EAs to solve problems is to select a solution representation. A single real number, a binary value, a string, or a combination of different values can represent a solution. For example, in the OneMax problem, the representation is a bit-string of size $n$, such as $x = (x_1, \dots, x_n)$, where $x_i$ denotes the value of the $i$-th variable in the problem. Each solution needs to be evaluated so that the algorithm can compare different solutions according to the fitness function $f$. In practice, a fitness evaluation is usually costly. Therefore, an optimization algorithm should find a good or near-optimal solution while minimizing the number of fitness evaluations needed.

In EA, the population refers to a set of candidate solutions. The formation of the population changes with iterations. Generally, a group is defined as a group of individuals of a fixed size. EA needs to introduce new solutions for the current population. The role of the reproduction operator is to create offspring from the parents of the current population, and they should fit the selected representation to work correctly. Generally, two operators (crossover and mutation) are used to perform this step.

### Mutation Operator

Because of the dominance of candidate solutions and their high chance of being selected for reproduction, most individuals converge on the best-found solution. Mutation operators ensure that the population is not trapped in a local area, and the outputs of this step depend on the outcomes of a series of random choices. A mutation operator acts on a single individual and produces a modified offspring from it. When executing the mutation individual, the mutation probability is set in the evolution algorithm. The following introduces a typical example of a binary string mutation operator. In a bit string of length $n$, mutator can be defined as flipping each bit independently with a certain probability $p$. A common choice for $p$ is $1/n$, which prevents operators from getting offspring as if they were generating random bit strings from scratch.

We introduce another mutation operator that is investigated in Chapter 4, named heavy-tail mutation operator. Doerr et al. (2017) pointed out that when a multi-bit flip is necessary to leave a local optimum, it may take a longer time for the algorithm to find the right bits to flip if using standard bit mutations. The heavy-tail mutation operator overcomes the mentioned negative effect when using standard bit mutations and is at the same time structurally close to the traditional way of performing mutations. A general belief that a dynamic choice of the mutation rate as done in heavy-tail mutation can be profitable. Theoretical studies show that the performance of the $(1+1)$ EA using a heavy-tail mutation operator is better than the standard $(1+1)$ EA in solving *jump* functions (Doerr et al., 2017).

In the heavy-tail mutation operator, Doerr et al. (2017) introduced that the mutation rate is chosen randomly in each iteration according to a power-law distribution with (negative) exponent $\beta > 1$. The heavy-tailed choice of the mutation rate ensures that with probability $\ominus(k^{-\beta})$, exactly $k$ bits are flipped. The power law distribution is given as follows.

**Theorem 3.3.1** (Discrete power-law distribution: $D^{\beta}_{n/2}$)**.** *Let $\beta > 1$ be a constant. If a random variable $X$ follows the distribution $D^{\beta}_{n/2}$, then*

$$P_r(X = \theta) = \left(C^{\beta}_{n/2}\right)^{-1} \theta^{-\beta} \tag{3.1}$$

*for all $\theta \in [1, .., n/2]$, where the normalization constant is $C^{\beta}_{n/2} := \sum_{i=1}^{n/2} i^{-\beta}$.*

In this thesis, we use the definition of the heavy-tail mutation operator proposed by Doerr et al. (2017) as follows: when the parent individual is a bit string $x \in \{0,1\}^n$, the mutation operator first chooses a random mutation rate $\theta/n$ with $\theta \in [1, .., n/2]$ chosen according to the power-law distribution $D^{\beta}_{n/2}$ and then creates an offspring by

**Algorithm 5:** The heavy-tail mutation operator

```
1: $x = \{x_1, .., x_n\} \in \{0,1\}^n$;
2: Choose $\theta \in [1, .., n/2]$ randomly according to $D^{\beta}_{n/2}$;
3: for $j = 1$ to $n$ do
4:     if $rand([0,1]) \leq \theta/n$ then
5:         $y_i \leftarrow 1 - x_i$
6:     else
7:         $y_i \leftarrow x_i$
8:     end if
9: end for
10: return $y = \{y_1, .., y_n\}$
```

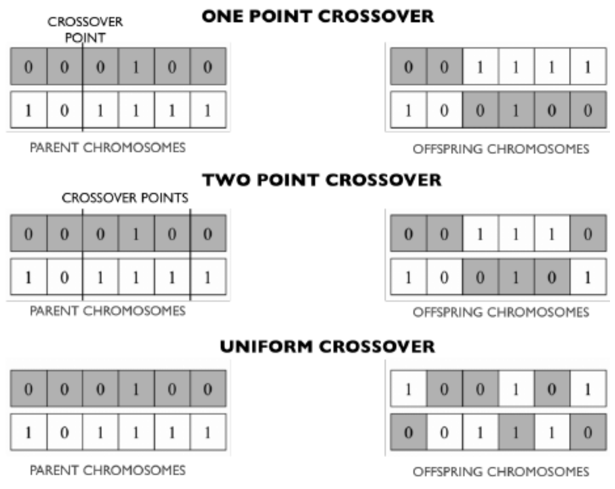


FIGURE 3.1. Examples of crossover operators

flipping each bit of string independently with probability $\theta/n$. The working principle of this operator is given in Algorithm 5.

### Crossover Operator

The crossover is responsible for recombination of the parents, and it results in inheritance. Different to mutation which performs on one parent and produces one child, crossover is done on two selected parents and produces one or more offspring. For bit-string representations, the single-point crossover and uniform crossover are commonly used operators. For example, consider two solution $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$, the single-point crossover operator chooses a random number $1 \leq i \leq n-1$, cuts both individuals from the $i$-th bit, and swaps the sub-parts. The outcome individuals of this operation will be $z = (x_1, \dots, x_i, y_{i+1}, \dots, y_n)$ and $w = (y_1, \dots, y_i, x_{i+1}, \dots, x_n)$. Uniform crossover selects the $i$-th bits from $\{x_i, y_i\}$, uniformly at random and assigns to the outcome individuals. This operator randomly chooses a bit for the first offspring and sets the other bit for the second offspring. Figure 3.1 gives examples of how these crossover operators produce offspring.

### Selection Mechanism

In EAs, there are two possible types of selection mechanisms, parental selection and survival selection. Selection in EAs is used to select individuals according to their quality (i.e. the value of the fitness function of the problem). For a maximisation problem, an individual with higher fitness value is better.

Parental selection is to distinguish individuals according to their quality in fitness function, especially to allow better individuals to become the parents in the following operations. In this way, the algorithm can ensure that future generations inherit the beneficial characteristics of better solutions. Together with the survival choice mechanism, parental choice is responsible for promoting quality improvement. However, it can not overcome the randomness of evolutionary algorithm. Therefore, high-quality individuals have more opportunities to become parents than low-quality individuals. However, low-quality individuals often have a slight but positive opportunity to become parents.

Roulette wheel selection is a commonly used selection operator which calculates the probability of selecting each individual based on its fitness value. The other selection operator in many standard evolutionary algorithms is tournament selection. Tournament selection chooses pre-given number of individuals uniformly at random and selects the better one as the parent. Here, the number of chosen parents depends on the reproduction operators, which determine how to combine parents to create offspring.

After finalising the offspring reproduction, evolutionary algorithms distinguish the individuals based on their quality by using survival selection. Then EAs create a new population by replacing individuals from the current population with offspring. It is similar to parent selection but used in a different stage of the evolutionary cycle. Depending on the number of generated offspring and the size of the population, there are different strategies to perform the survival selection. In this step, the algorithms either compare offspring with the parents, select the ones with better fitness value, or select the next generation from the offspring solutions. Moreover, elitism survival selection transfers the best of the previous population directly to the next population. The selection mechanism could also include moving some random solutions to the next population in order to ensure random inheritance.

#### Termination Condition

The evolutionary cycle continues until a certain threshold is reached. Because the evolutionary algorithms are random and do not guarantee to reach the optimal solution, this condition may never be satisfied and the algorithms may never stop. Therefore, the following options are usually used for thresholds: either the algorithms reach a preset number of generations, or the algorithms perform a determined number of fitness evaluations, or there is no significant additional improvement in the quality of the results.

### 3.3.2 (1+1) EA

(1+1) EA (see Algorithm 6) is a fundamental single-objective evolutionary algorithm in which the population contains only one solution, and only one offspring is generated for each iteration. If the new solution is better than or at least as good as the current solution in terms of the fitness function, the new solution will replace the current solution. This algorithm is quite similar to RLS but different in constructing the new solution. In RLS, a solution in the defined neighbourhood is selected, while in (1+1) EA, each part of the representation (e.g. each bit of the bit-string in the knapsack problem example) is mutated with a small probability, which named

**Algorithm 6:** (1+1) EA

```
1: Choose $x \in \{0,1\}^n$ uniformly at random.
2: while stopping criterion not met do
3:     $y \leftarrow$ flip each bit of $x$ independently with probability of $\frac{1}{n}$;
4:     if $f(y) \geq f(x)$ then
5:         $x \leftarrow y$ ;
6:     end if
7: end while
```

mutation rate.. Usually, this probability is equal to $1/n$, where $n$ is the size of the solution. Therefore, (1+1) EA might flip more than one bit, significantly improving its performance compared to RLS.

### 3.3.3 Multi-Objective Evolutionary Algorithms

In many disciplines, optimisation problems have two or more objectives, which usually conflict with each other. We want to optimise those objectives at the same time. These problems are called "multi-objective", and their solutions involve designing different algorithms when dealing with single objective optimisation problems. Let $k$ denote the number of objectives, then each solution $x$ will be a $k$-dimensional point $f(x) = (f_1(x), \dots, f_k(x))$ in the objective space. As the name implies, multi-objective optimisation handles multiple objectives which may conflict each other potentially. Usually, the optimal solution for one objective is not guarantee to be the optimal solution for any other objectives. In a maximisation problem, we say that a solution $x$ (weakly) dominates another solution $y$ and denote it by $f(x) \succeq f(y)$, if $f(x) \neq f(y)$ and $f_i(x) \geq f_i(y)$, $\forall i \in \{1, \dots, n\}$. In such a scenario, not only one optimal solution can be obtained, but a number of solutions are all optimal and we call these solutions Pareto-optimal solutions.

Evolutionary algorithms have been widely used in solving multi-objective optimisation problems, mainly because their are easy to use and have a wide range of applications (Deb, 2001; Coello, Veldhuizen, and Lamont, 2002). Multi-objective Evolutionary Algorithms (MOEAs) are stochastic population-based meta-heuristics that employ the principles of natural selection to drive a set of solutions toward the Pareto optimal front. In the recently research on evolutionary multi-objective optimisation, researchers concentrate on finding a set of well distributed Pareto optimal solutions, and then select a specific solution according to the compromise information of the solution.

Several multi-objective evolutionary algorithms have been introduced in the literature, starting with Vector Evaluation Genetic Algorithm (VEGA) (Schaffer, 1985). The basic idea of the algorithm is to divide the population into $M$ subpopulations of equal size. Then, in each of those subpopulations, the selection is made by considering the unique corresponding target. Once the selection mechanism is implemented, the remaining evolutionary operators apply to the outcome population. All these processes are repeated in each generation. Later, a weight-based genetic algorithm (WBGA) was introduced in Hajela and Lin, 1992 to assign a weighting coefficient to the objective function used in the algorithm. Compared with the classic weighted summation method, each individual in the group has its weighting coefficient vector.

**Algorithm 7:** GSEMO

```
1: Choose $x \in \{0,1\}^n$ uniformly at random ;
2: $S \leftarrow \{x\}$;
3: while stopping criterion not met do
4:    choose $x \in S$ uniformly at random;
5:    $y \leftarrow$ flip each bit of $x$ independently with probability of $\frac{1}{n}$;
6:    if $(\nexists w \in S : w \succeq y)$ then
7:       $S \leftarrow (S \cup \{y\}) \backslash \{z \in S \mid y \succeq_{GSEMO} z\}$ ;
8:    end if
9: end while
```

In other words, WBGA can find multiple non-dominated solutions in one run. However, WBGA is a weight-based method; therefore, it cannot find the Pareto optimal solution in the non-convex part of the front edge.

Later, other multi-objective evolutionary algorithms were introduced to guide the search to find non-dominated solutions unambiguously. Multi-objective genetic algorithm (MOGA) (Murata and Ishibuchi, 1995) definitive uses Pareto-based ranking to guide the search for the actual Pareto front while maintaining population diversity. Each individual is assigned a function, which calculates the number of individuals dominating it, and this function is shown as a rank. In the next two sections, we theoretically and empirically describe the multi-objective methods we consider in this paper (GSEMO and NSGA-II).

### GSEMO

Now, we introduce a simple multi-objective evolutionary algorithm called Global SEMO (GSEMO), which is present in Algorithm 7 for a multi-objective optimization problem. GSEMO is proposed by Giel (2003) and is a simple extension of the (1+1) EA. The GSEMO starts with an initial population generated with one randomly selected solution. GSEMO randomly selects one solution from the current population in its main loop, then the mutation operator is performed on it, and the survival selection procedure is done afterwards. In the survival selection procedure, the new solution is checked whether it is dominated by at least one other solution in the population. If the new solution is not dominated by any other solutions in the current population, then it is added to the population, and all other solutions that are dominated by the new solution are removed from the population. In other words, GSEMO only keeps non-dominated solutions; this is how it controls its population size.

### NSGA-II

Non-dominated sorting genetic algorithm (NSGA) (Srinivas and Deb, 1994) and its improved version, NSGA-II (Deb et al., 2002) are well-known multi-objective evolutionary algorithms that focus on diversity as well as finding near-optimal solutions. The non-dominated sorting strategy divides the individuals in the population into several fronts. NSGA first identifies non-dominated individuals from all members of the population. The front row of these individuals is 1, and they are assigned an important virtual fitness value. After that, the algorithm deletes the members in the first front from the current population. The non-dominated individuals were identified from the truncated population and ranked at 2. This process continues until

**Algorithm 8:** NSGA-II

```
1:  Generate initial population set $P_0$ and offspring set $Q_0$ ;
2:  set $t \leftarrow 0$ ;
3:  while stopping criterion not met do
4:      $R_t \leftarrow P_t \bigcup Q_t$; combine parent and offspring population
5:      $F \leftarrow$ fast-non-dominated-sort $(R_t)$; $F = (F_1, F_2, \ldots)$, all non-dominated front
        of $R_t$
6:      $P_{t+1} \leftarrow \emptyset$ and $i \leftarrow 1$;
7:      while $|P_{t+1}| + |F_i| \leq N$ do
8:          crowding-distance-assignment $(F_i)$;
9:          $P_{t+1} \leftarrow P_{t+1} \bigcup F_i$;
10:         $i \leftarrow i + 1$;
11:     end while
12:     Sort $F_i$ based on the crowding distance in descending order;
13:     $P_{t+1} \leftarrow P_{t+1} \bigcup F_i[1 : (N - |P_{t+1}|)]$;
14:     $Q_{t+1} \leftarrow$ make-new-population $(P_{t+1})$;
15:     $t \leftarrow t + 1$;
16: end while
```

all members of the population are classified. Furthermore, using a parameter $\sigma_{share}$ that indicates the neighbourhood size, a fitness sharing strategy is applied to change the fitness of solutions based on how close they are positioned in the objective space. Diversity maintenance is achieved in NSGA by using this strategy.

However, NSGA can lose good solutions in the survival selection procedure. In order to fix this, NSGA-II is proposed (Deb et al., 2000; Deb et al., 2002). In the initial state, NSGA-II randomly generates a population $P_0$, and assigns fitness (or rank) to each solution according to its non-dominated rank. Then, the commonly used selection, recombination and mutation operators are used to generate the offspring population $Q_0$. Algorithm 8 describes the implementation of NSGA-II in the initial steps and after each dynamic change.

The basic iteration of NSGA-II is different from the first generation. Deb et al. (2002) described the working process of each iteration. First, merge the two populations $P_t$ and $Q_t$ into a population $R_t$ with a size of $N$. Second, perform a non-dominated sorting to classify the entire group $R_t$ and subdivide $R_t$ into several categories $F = (F_1, F_2, \ldots)$. Then, the individuals in the best non-dominated front are insert to the new population $P_{t+1}$. Starting with the first front, it adds the solution to $P_{t+1}$ until the parent population is filled. NSGA-II calculates the crowding distance in $F_i$. The crowding distance estimates the perimeter of the box around the solution, which is formed using the nearest neighbour of the solution as the vertex. Therefore, a large crowding distance means that the solution is located in a sparse area. It assigns an infinite value to the solutions which are boundary solutions with largest or smallest value of each fitness function. Finally, the algorithm uses evolutionary operator to generate $Q_{t+1}$ from $P_{t+1}$ for ranking and crowding distance.

### 3.3.4 Differential Evolution

Differential Evolution (DE) is a simple and effective evolutionary algorithm used to solve global optimisation problems in a continuous domain (Storn and Price, 1997; Neri and Tirronen, 2010; Pham, Malinowski, and Bartczak, 2011). The first written article on DE appeared as a technical report by Storn and Kenneth (1995) and then coming with a series of papers (Price, 1996; Storn and Price, 1997). Comprehensive survey papers (Das and Suganthan, 2010; Das, Mullick, and Suganthan, 2016) provide an up-to-date view of this algorithm and discuss its various modifications, improvements and uses. Moreover, DE and its variants have been successfully applied to solve numerous real-world problems from diverse domains of science and engineering (Das and Suganthan, 2010; Neri and Tirronen, 2010).

The DE algorithm processes the population $X = \{x_1, x_2, \dots, x_{N_p}\}$ consisting of $N_p$ individuals encoded as n-dimensional vectors of real numbers. DE is a simple real parameter optimisation algorithm. After random initialisation, it works through a simple cycle of stages, mutation, crossover and selection mechanism. We explain each stage separately in the following section.

### Mutation

The mutation operator has a few basic variants, the details of which can be found in (Storn and Price, 1997; Price, Storn, and Lampinen, 2006). Storn and Price (1997) introduced notations $DE/X/Y/Z$ of the mutation operator, where $X$ denotes the reproduction method, $Y$ the number of difference vectors, $Z$ stands for the type of crossover being used. In each literature, DE algorithm selects a vector from the current population and this vector named *target* vector. A mutation operator is applied on the *target* vector and then obtains a *donor* vector. Finally, a *trial* vector is generated by recombining the *donor* with the *target* vector. The most common one is $DE/rand/1$ which consists of randomly choosing three individuals from the population to create the donor vector for each $i$-th target vector, and adding to the first of them $x_{r_1}$ a scaled difference between two others $x_{r_2}$ and $x_{r_3}$

$$u_i \leftarrow x_{r_1} + F \cdot (x_{r_2} - x_{r_3}). \tag{3.2}$$

Parameter $F$ is called a scaling factor, as it shrinks the length of the difference vector $(x_{r_2} - x_{r_3})$. Storn and Price (1997) also defined the $DE/rand/k$ variant which uses a larger number of difference vectors:

$$u_i \leftarrow x_{r_1} + F_1 \cdot (x_{r_2} - x_{r_3}) + \ldots + F_k \cdot (x_{r_{2k}} - x_{r_{3k}}), \tag{3.3}$$

where the scaling factors are usually assumed to be equal. In practice, the most frequently encountered mutation operators have one or two difference vectors.

The other most frequently referred mutation operators are listed below:

$$\text{DE/best/1}: \quad u_i \leftarrow x_{best} + F \cdot (x_{r_1} - x_{r_2}) \tag{3.4}$$
$$\text{DE/current - to - best/1}: \quad u_i \leftarrow x_i + F \cdot (x_{best} - x_i) + F \cdot (x_{r_1} - x_{r_2}) \tag{3.5}$$
$$\text{DE/best/2}: \quad u_i \leftarrow x_{best} + F \cdot (x_{r_1} - x_{r_2}) + F \cdot (x_{r_3} - x_{r_4}) \tag{3.6}$$
$$\text{DE/rand/2}: \quad u_i \leftarrow x_{r_1} + F \cdot (x_{r_2} - x_{r_3}) + F \cdot (x_{r_4} - x_{r_5}) \tag{3.7}$$

### Crossover

In DE, the crossover operator is based on exchanging elements between vectors encoding the parent and the mutant. The crossover follows the mutation operator. For each mutated vector $u_i$, a trial vector $v_i$ is generated using the following equation,

**Algorithm 9:** Differential Evolution Algorithm

```
Read values of the control parameters of DE: scale factor $F$, crossover rate $CR$,
  and the population size $NP$ from user.
Set the generation number $G = 0$ and randomly initialize a population of $NP$
  individuals, and each individual uniformly distributed in the given range of
  decision variables.
while stopping criterion not met do
    for each individual $i$ in the population do
        Mutation Step
        Generate a donor vector $u_i$ corresponding to the $i$-th target vector via the
          differential mutation scheme of DE;
        Crossover Step
        Generate a trail vector $v_i$ for the target vector through crossover operator;
        Selection Step
        Evaluate the trail vector, keep the better one between the target vector
          and the trail vector to the next generation.
    $G = G + 1$
return the best solution in the final population according to the fitness function.
```

so-called binomial crossover.

$$v_{it} = \begin{cases} u_{it} & \text{if } rand(0,1) \leq CR \text{ or } t = k \\ x_{it} & \text{otherwise} \end{cases} \tag{3.8}$$

where $rand(0,1)$ randomly generates a number between 0 and 1, $t$ is the gene under consideration. $k$ is a random integer selected from $\{1, \ldots, n\}$ to ensure that at least one parameter is considered when constructing the trail vector. The parameter $CR$ is the crossover probability, which assumes a value between 0 and 1. Generally, the crossover probability $CR$ is more sensitive to the nature and complexity of the problem (such as multi-modality), and the mutation constants regulate the convergence rate. We found that $CR = 0.5$ is a good choice (an example is given by Kaelo and Ali (2006)).

### Selection

The selection operator employs a straightforward one-to-one competition scheme between $v_i$ and $x_i$, regarding the fitness function, only the better of them is chosen for the next generation. The pseudo-code for the DE algorithm can be seen in Algorithm 9.

## 3.4 Analytical Methods

The history of runtime analysis of bio-inspired computing begins with Mühlenbein (1992). Since then, this area has provided many rigorous new insights into the working behaviour of bio-inspired computation methods for combinatorial optimisation problems (Auger and Doerr, 2011; Neumann and Witt, 2010). The computational analysis of bio-inspired methods considers the number of evaluations required to find an optimal solution concerning the input size and plays a significant role in their theoretical understanding. In this section, we introduce some of the techniques in runtime analysis for combinatorial optimisation problems. Some of the techniques can be applied to analyse the behaviour of evolutionary algorithms and randomised local search

(He and Yao, 2001; Doerr, 2020). The following sections introduce techniques and analytical tools that we used in this thesis.

### 3.4.1 Probabilistic Tools

This section introduces the well-known mathematical tools that are mainly used to bound the probability of a random variable deviating from its expected value by considering its variance. Two of the most used probabilistic tools, namely Chebyshev's inequality and Chernoff bounds, are presented in the following section. In this thesis, we briefly introduce these bound without proof, while the in-detail proof can be found in the textbooks by (Motwani and Raghavan, 1995; Doerr, 2020).

Chebyshev's inequality, also called Bienayme-Chebyshev inequality as it was first stated by (Bienaymé, 1853) and later proven by (Chebyshev, 1867).

**Theorem 3.4.1** (Chebyshev's inequality)**.** *Let $x$ be a random variable with expectation $E(x)$ and variance $Var(x)$. Then for any $\lambda \in \mathbb{R}^+$, the following inequalities hold.*

$$P_r\left[|x - E(x)| \geq \lambda\sqrt{Var(x)}\right] \leq \frac{1}{\lambda^2},$$
$$P_r\left[|x - E(x)| \geq \lambda\right] \leq \frac{Var(x)}{\lambda^2}.$$

Note that the Chebyshev's inequality automatically generates bilateral tail boundaries. Chebyshev's inequality has a one-sided version. It is called Cantelli-Chebyshev ineqaulity (Marshall, Olkin, and Arnold, 1979).

**Theorem 3.4.2** (Cantelli-Chebyshev inequality)**.** *Let $x$ be a random variable with $Var(x) > 0$, then for all $\lambda > 0$, the following inequalities hold.*

$$P_r\left[x \geq E(x) + \lambda\sqrt{Var(x)}\right] \leq \frac{1}{\lambda^2 + 1}, \tag{3.9}$$

$$P_r\left[x \leq E(x) - \lambda\sqrt{Var(x)}\right] \leq \frac{1}{\lambda^2 + 1}. \tag{3.10}$$

The bounds presented below are typically called Chernoff bounds or Chernoff-Hoeffding bounds, referring to the seminal papers by (Chernoff et al., 1952; Hoeffding, 1994). Chernoff bounds give exponentially decreasing bounds on the probability that the sum of some independent random variables deviate from their expected values and provide sharper tails with exponential decay behavior. Those bounds are sharper than other known tail bounds such as Markov inequality and Chebyshev's inequality.

**Theorem 3.4.3** (Chernoff bounds)**.** *Let $x_1, \ldots, x_n$ be indenpendent random variables taking values in $[0, 1]$. Let $x = \sum_{i=1}^n x_i$. Then, the following inequalities hold:*

$$P_r[x \geq (1+\epsilon)E(x)] \leq \left(\frac{e^{\epsilon}}{(1+\epsilon)^{(1+\epsilon)}}\right)^{E(x)} \qquad \epsilon > 0 \tag{3.11}$$
$$P_r[x \geq (1+\epsilon)E(x)] \leq e^{\left(\frac{-E(x)\epsilon^2}{3}\right)} \qquad 0 < \epsilon \leq 1 \tag{3.12}$$

### 3.4.2 Fitness Based Partitions

For this simple method introduced by Wegener (2002), we assume that the considered algorithm is a stochastic search algorithm working with one solution that produces one offspring in each iteration. RLS and (1+1) EA are examples of these kind of algorithms. Let $S$ be the search space and $f : S \to \mathbb{R}$ be the objective function that should be maximised. Dividing $S$ into $m$ partitions, denoted by $A_1, A_2, \ldots, A_m$, assume that for any $i < j$, $x \in A_i, y \in A_j$, we have $f(x) < f(y)$. This assumption indicates that the fitness value of solutions increases when increasing the index of partitions. Moreover, let $A_m$ only include optimal solutions. For a search point $x \in A_i$, the probability is denoted by $p(x)$ that in the next step a solution $y \in A_{i+1} \cup \ldots \cup A_m$ is produced. Let $p_i = \min_{x \in A_i} p(x)$ denote the smallest probability of producing a solution with higher partition number. With Lemma 4.1 of Neumann and Witt (2010) (presented here as Lemma 3.4.4), that the expected optimization time is upper bounded by $\sum_{i=1}^{m}(1/p_i)$.

**Lemma 3.4.4.** *The expected optimization time of a stochastic search algorithm that works at each time step with a population of size* 1 *and produces at each time step a new solution from the current solution is upper bounded by* $\sum_{i=1}^{m}(1/p_i)$.

### 3.4.3 Drift Analysis

Runtime analysis is a relatively recent and increasingly popular approach in the theory of randomised search heuristics. Drift analysis considers the drift of a stochastic process, which is the expected progress of the underlying process from one-time step to another. Generally speaking, this method counts the number of steps required for the algorithm to fill a gap defined by a potential auxiliary function. Drift analysis is first introduced by Sasaki and Hajek (1988), and He and Yao (2001) were the first to apply drift analysis to evolutionary algorithms. In this section, we first give the definition of the drift and then introduce variants of drift theorems.

The expected one-step change $\delta_t := E[X_t - X_{t+1} | F_t]$ for $t \geq 0$ is called drift. Here, we consider the $X_t$ as a non-negative random variable and a natural filtration $F_t = (X_0, \ldots, X_t)$, i.e. the information available up to time $t$. Since the outcomes of $X_0, \ldots, X_t$ are random, the $\delta_t$ is a random variable. Suppose we try to bound $\delta_t$ to some $\delta^* > 0$ from below to get all possible results of $\delta_t$, where $t < T$. Then, we know that the process will reduce its expected value by at least $\delta^*$, and *additive drift analysis* will provide a bound on $T$ that depends only on $X_0$ and $\delta^*$. While the additive drift analysis is very powerful and have gained success, a variation of it is introduce in Doerr, Johannsen, and Winzen (2012). *Multiplicative drift analysis* considers the improvements that are proportional to the current value of the auxiliary function.

The first formal drift theorem was given by He and Yao (2001). We present a version due to Lehre and Witt (2013) which removed the discrete search space and the Markov property and only demand a bounded state space.

**Theorem 3.4.5.** *(Additive Drift) Let* $(X_t)_{t \geq 0}$, *be a stochastic process over some bounded state space* $S \in \mathbb{R}_0^+$. *Assume that* $E(T|X_0) < \infty$. *Then:*

1. *If* $E(X_t - X_{t+1} | F_t; X_t > 0) > \delta_u$ *then* $E(T|X_0) \leq \frac{X_0}{\delta_u}$.

2. *If* $E(X_t - X_{t+1} | F_t) \leq \delta_\ell$ *then* $E(T|X_0) \geq \frac{X_0}{\delta_\ell}$.

By applying the law of total expectation, the statement (1) implies $E(T) \leq \frac{E(X_0)}{\delta_u}$ and analogously for statement (2). This theorem links the expected change in potential to the first time the potential research zero.

In the multiplicative drift theorem, the value of random variable $X_t$ is improved by a proportion of its current value at each step. Concerning this theorem, runtime bounds of evolutionary algorithms on several combinatorial optimisation problems have been found, and the definition of the multiplicative drift introduced here is the version presented by Doerr, Johannsen, and Winzen, 2012.

**Theorem 3.4.6.** *(Multiplicative Drift) Let $S \subseteq \mathbb{R}$ be a finite set of positive numbers with minimum $s_{min}$. Let $\{X_t\}_{t\in\mathbb{N}}$ be a sequence of random variables over $S \cup \{0\}$. Let $T$ be the random variable that denotes the first point in time $t \in \mathbb{N}$ from which $X_t = 0$. Suppose that there exists a real number $\delta > 0$ that*

$$E[X_t - X_{t+1} | X_t = s] \geq \delta s$$

*holds for all $s \in S$ with $P_r[X_t = s] > 0$. Then for all $s_0 \in S$ with $P_r[X_0 = s_0] > 0$, we have*

$$E[T|X_0 = s_0] \leq \frac{1 + \ln(s_0/s_{min})}{\delta}. \tag{3.13}$$

### Runtime analysis of OneMax

The function OneMax is easy to analyse, and the upper limit of the expected runtime is determined by Mühlenbein (1992). When using (1+1) EA to solve the OneMax problems, the mutation steps which increase the number of one-bits will be treated as successful steps. Since the probability of flipping each bits is equal to $1/n$ where $n$ is the length of decision variables, there are at most $n$ successful mutation steps for each OneMax problem. If the number of zero-bits in the current bit string is equal to $i$, the probability of successful steps is bounded by $\frac{i}{n}\left(1 - \frac{1}{n}\right)^{n-1}$, so if $T$ is the waiting time until the optimum is produced, we have

$$E[T] \leq \sum_{i=1}^{n} \left( \frac{i}{n} \left(1 - \frac{1}{n}\right)^{n-1} \right)^{-1}, \tag{3.14}$$

we have $\lim_{n\to\infty}\left(1 - \frac{1}{n}\right)^n = \frac{1}{e}$ and $\sum_{i=1}^{n} \frac{1}{i} = O(\ln n)$, then,

$$\sum_{i=1}^{n} \left( \frac{i}{n} \left(1 - \frac{1}{n}\right)^{n-1} \right)^{-1} \leq en \sum_{i=1}^{n} \frac{1}{i} = O(n \ln n). \tag{3.15}$$

This is an upper bound for the expected running time of the (1+1) EA. Doerr, Johannsen, and Winzen (2012) studied in the time the (1+1) EA needs to find the minimum of OneMax function. Therefore, in the selection step of each iteration, the (1+1) EA only accept the candidate solution with the number of one bits that does not increase. Then, consider the progress $\Delta_t := OneMax(X_t) - OneMax(X_{t+1})$ of the (1+1) EA in the $t$-th iteration. By the working principle of the (1+1) EA, $\Delta_t$ cannot be negative, and by definition of OneMax, $X_t = OneMax(X_t)$. For exactly flip each of these one-bits, the probability of increasing the value of $OneMax(X_t)$ by one is $\frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{1}{en}$. Hence,

$$E[\Delta_t|X_t] \geq \frac{OneMax\,(X_t)}{en}. \tag{3.16}$$

Thus, multiplicative drift analysis gives the result as follows:

$$\begin{aligned} E[T] \leq en\,(1+\ln E[OneMax(X_0)]) &= en\left(1+\ln\left(\frac{n}{2}\right)\right) \\ &= O(n\log n), \end{aligned} \tag{3.17}$$

which is the same to the result in (3.14).

## 3.5 Conclusion

In this chapter, we went into detail with the randomised local search and the main concepts of evolutionary computation. After describing the simplest evolutionary algorithm (1+1) EA, we turned our attention to multi-objective evolutionary algorithms and presented an algorithm as an extension of (1+1) EA for multi-objective optimisation called GSEMO. Then, we introduced NSGA-II, which is a well-known advanced multi-objective algorithm.

Moreover, we presented some techniques for the runtime analysis of evolutionary algorithms. We introduced probabilistic tools for finding the maximum probability that a random variable deviates from its expected value. Then, we introduced the method of fitness-based partitions, which tracks the quality of solutions moving through defined levels. Finally, we included important theorems from the field of drift analysis, which provide a strong tool for finding upper and lower bounds on the expected optimisation times of bio-inspired algorithms.

Chapter 4

# Evolutionary Algorithms for the Chance-Constrained Knapsack Problem

## 4.1 Introduction

Evolutionary algorithms have been used for various stochastic optimisation problems (Till et al., 2007; Horng, Lin, and Yang, 2012; Nguyen and Yao, 2012; Rakshit, Konar, and Das, 2017). Evolutionary algorithms can obtain good-quality solutions in most cases within a reasonable amount of time, and can easily apply them to the solution of stochastic problems. However, the mathematical model of the chance-constrained optimisation problem is more difficult so it has received comparatively little attention in the evolutionary computation literature.

In this chapter, we consider the chance-constrained knapsack problem (CCKP; described in Section 2.3.1). Given a set of items with stochastic weights and deterministic profits, the goal of the CCKP is to find a set of items of maximal profit, subject to the condition that the probability with which the total weight will exceed the capacity bound is less than or equal to a given threshold. Here, the threshold is a small value limiting the probability of the constraint violation. To evaluate a solution concerning the chance constraint, we make use of two inequalities a Chebyshev's inequality and a Chernoff bound to calculate an upper bound on the probability of violating the capacity bound and as surrogate functions for the chance constraint. The probabilistic tools we employ allow us to estimate the probability of a constraint violation mathematically without the need for sampling.

We develop a single-objective approach and a multi-objective approach for evolutionary algorithms to examine the performance of evolutionary algorithms in terms of solving CCKP instances. First, we consider the simplest single-objective evolutionary algorithm (1+1) EA (see Algorithm 6) and its multi-objective version, GSEMO (see Algorithm 7), then we compare the performance of these algorithms with deterministic approaches. Since the results of this comparison provide a reasonable justification for using evolutionary optimisation when dealing with the chance-constrained knapsack problem, we then introduce two reproduction operators: the problem-specific crossover operator and the heavy-tail mutation operator. These operators are used to improve the solutions produced by EAs. We then investigate the performance of some single-objective and multi-objective evolutionary algorithms in terms of solving the CCKP.

This chapter extends the work published at the GECCO conference (Xie et al., 2019; Xie, Neumann, and Neumann, 2020).

This chapter is organized as follows. In Section 4.2, we introduce the surrogate of the chance constraint. In Section 4.3 and Section 4.4, we present the deterministic approaches and evolutionary algorithms that can be employed to solve the CCKP, respectively. Computational experiments and an investigate of the results are described in Section 4.5, followed by a conclusion in Section 4.6.

## 4.2 Surrogate Functions for the Chance Constraint

Recall the definition of the chance-constrained knapsack problem in Section 2.3.1. We assume the weights of items are independent of each other, with each weight $w_i$ corresponding to the expected value $a_i = E[w_i]$ and variance $\sigma_i^2 = Var[w_i]$. Let $W(x) = \sum_{i=1}^{n} w_i x_i$ be the total weight of a given solution $x = \{x_1, ..., x_n\}$, with $E[W(x)] = \sum_{i=1}^{n} a_i x_i$ denoting the expected weight of the solution derived by linearity of expectation. Furthermore, $Var(x) = \sum_{i=1}^{n} \sigma_i^2 x_i$ denotes the variance of the weight under the assumption that the variables of items are independent. This section introduces how address the chance constraint by estimating the probability of violating the capacity bound.

For cases where the weights of items are chosen according to the Normal distribution, the violation probability of a given solution can be computed using the cumulative distribution function. For other distributions, such as the uniform, exponential, and Poisson distributions, Monte Carlo simulation can compute the probability of a given solution to an arbitrary level of accuracy. Monte Carlo method is a general term for a class of algorithms that rely on repeated random sampling to produces distributions of possible outcome values. Raychaudhuri (2008) gives an introductory tutorial on Monte Carlo simulation and briefly describes the nature and relevance of Monte Carlo simulation. Kroese et al. (2014) introduce some typical uses of Monte Carlo methods and offer the reasons why Monte Carlo methods are so popular. For cases where the weights of items follow the uniform distribution, Monte Carlo simulation samples from the uniform distribution of each chosen item of a given solution. Each set of samples is called an iteration, and the resulting outcome (feasible or infeasible) from that set of samples is recorded. Monte Carlo simulation does this enough times to numerically guarantee the probability distribution of possible outcomes to an acceptable level of accuracy. However, the computation efficiency of Monte Carlo simulation is low because it repeats sampling even more than a thousand times. Watson and Gordon (1986) provide an approximation particularly suited to cases where the variables follow a uniform distribution but with a small $\alpha$. Luo and Shevchenko (2009) propose an efficient direct numerical integration algorithm for computing the probability of compound distributions by using a characteristic function.

We discuss two distributions of the weights in CCKP, which are the Normal and uniform distribution. However, as mentioned above, it is hard to evaluate the violation probability for both distributions by using the same method effectively in an acceptable amount of time. Therefore, we estimate the violation probability by applying Chebyshev's inequality and Chernoff bounds, introduced in Section 3.4.1. We present the surrogate functions of the chance constraint (2.17) in the rest of this section by applying these two bounds on the tails of probabilities.

### 4.2.1 Chebyshev's Inequality

Since this chapter only considers the violation of the capacity bound $C$, we use the one-sided Chebyshev's inequality (3.9) to reformulate the chance constraint. The inequality can be applied to any probability distribution with known expectation and variance. To match the expression of Chebyshev's inequality, we set $E[W(x)] + \lambda\sqrt{Var(x)} = C$ and we have $\lambda = \frac{C-E[W(x)]}{\sqrt{Var(x)}}$, then we have a general formula to calculate the upper bound of the chance constraint as follows:

$$P_r\left(W(x) \geq C\right) \leq \frac{Var(x)}{Var(x) + (C - E[W(x)])^2}. \tag{4.1}$$

Hence, the constraint (2.17) can be reformulated as follows

$$\frac{Var(x)}{Var(x) + (C - E[W(x)])^2} \leq \alpha. \tag{4.2}$$

**Definition 4.2.1.** Given a solution $x$ with stochastic weight $W(x)$, we call $E[W(x)] + \sqrt{Var(x)\frac{1-\alpha}{\alpha}}$ the surrogate weight of $x$, denoted by $S(x)$.

**Theorem 4.2.1.** *If $x$ is a solution vector with surrogate weight $S(x) = E[W(x)] + \sqrt{Var(x)\frac{1-\alpha}{\alpha}}$, then, according to Chebyshev's inequality, the chance constraint is satisfied when $S(x) \leq C$ for all $\alpha \in (0, 1)$.*

To be notice here, the converse of Theorem 4.2.1 is not true so that the optimal solution may not satisfy the constraint. Furthermore, we consider two special cases where each item in the first case has a uniform distribution and takes value in $[a_i - \delta, a_i + \delta]$, which is named the additive uniform distribution. In the second case, each item takes value in $[(1-\beta)a_i, (1+\beta)a_i]$, having a uniform distribution called the multiplicative uniform distribution. Here, $\delta$ and $\beta$ define the uncertainty of the weights of items. For the variable which has a uniform distribution on the interval $[g, h]$, the expectation of this variable is $\mu = \frac{g+h}{2}$ and the variance is $\sigma^2 = \frac{(h-g)^2}{12}$. Applying Chebyshev's inequality to the chance constraint, we require

$$P_r\left(W(x) \geq C\right) \leq \frac{\delta^2 \sum_{i=1}^n x_i}{\delta^2 \sum_{i=1}^n x_i + 3(C - \sum_{i=1}^n a_i x_i)^2} \leq \alpha \tag{4.3}$$

for the additive uniform distribution and

$$P_r\left(W(x) \geq C\right) \leq \frac{\beta^2 \sum_{i=1}^n {a_i}^2 x_i}{\beta^2 \sum_{i=1}^n {a_i}^2 x_i + 3(C - \sum_{i=1}^n a_i x_i)^2} \leq \alpha \tag{4.4}$$

for the multiplicative uniform distribution. When each weight $w_i$ is chosen according to the Normal distribution $N(a_i, \sigma_i)$ (with all weights being independent of each other), we get

$$P_r\left(W(x) \geq C\right) \leq \frac{\sum_{i=1}^n \sigma_i^2 x_i}{\sum_{i=1}^n \sigma_i^2 x_i + (C - \sum_{i=1}^n a_i x_i)^2} \leq \alpha \tag{4.5}$$

### 4.2.2 Chernoff bounds

Chernoff bounds provide sharper tails with exponential decay behaviour. Those bounds are sharper than other known tail bounds, such as the Markov inequality and Chebyshev's inequality. Chernoff bounds assume that variables are independent and take on values in $[0, 1]$. There are several types of Chernoff bounds, but this thesis is only concerned with the Chernoff bound (3.11). The Chernoff bound is used to calculate the probability of violating the constraint incorporated into a surrogate function.

**Theorem 4.2.2.** *Let the stochastic weights $w_1, \ldots, w_n$ be independent variables with expected values $a_i, \ldots, a_n$, respectively. Let $C > 0$ be the capacity of the knapsack, $\sum_{i=1}^n w_i x_i$ denotes the total weight of a solution $x = (x_1, ..., x_n)$, and $E[W(x)] = \sum_{i=1}^n a_i x_i$ be the expected weight of the solution, Furthermore, let $\delta \geq 0$ be the uncertainty of the stochastic weights, and we have*

$$P_r\left(\sum_{i=1}^n w_i x_i \geq C\right) \leq \left(\frac{e^{\frac{C-E[W(x)]}{\delta\sum_{i=1}^n x_i}}}{\left(\frac{\delta\sum_{i=1}^n x_i + C - E[W(x)]}{\delta\sum_{i=1}^n x_i}\right)^{\frac{\delta\sum_{i=1}^n x_i + C - E[W(x)]}{\delta\sum_{i=1}^n x_i}}}\right)^{\frac{1}{2}\sum_{i=1}^n x_i} . \tag{4.6}$$

*Proof.* In the chance-constrained knapsack problem, the Chernoff bound can be applied in a unique case that weights of items are taken value according a uniform distribution in range $[a_i - \delta, a_i + \delta]$. All random variables have the same uncertainty $\delta$ but different initial and final boundaries. We than normalize the stochastic weights to make them chosen values in $[0, 1]$. Therefore, we set

$$y_i = \frac{w_i - (a_i - \delta)}{2\delta} \in [0, 1],\ Y(x) = \sum_{i=1}^n y_i x_i.$$

We then have,

$$E(y_i) = \frac{a_i - (a_i - \delta)}{2\delta} = \frac{1}{2}$$

and

$$E[Y(x)] = \frac{E[W(x)] - (\sum_{i=1}^n a_i x_i - \sum_{i=1}^n \delta x_i)}{2\delta} = \frac{1}{2}\sum_{i=1}^n x_i.$$

We introduce $Y(x)$ in the Chernoff bound, such that,

$$\begin{aligned}
\left(\frac{e^\epsilon}{(1+\epsilon)^{(1+\epsilon)}}\right)^{E[Y(x)]} &\geq P_r\left[Y(x) \geq (1+\epsilon)E[Y(x)]\right] \\
&= P_r\left[\sum_{i=1}^n \frac{w_i - (a_i - \delta)}{2\delta} x_i \geq (1+\epsilon)\frac{\sum_{i=1}^n x_i}{2}\right] \\
&= P_r\left[\sum_{i=1}^n w_i x_i - \sum_{i=1}^n (a_i - \delta) x_i \geq (1+\epsilon)\delta\sum_{i=1}^n x_i\right] \\
&= P_r\left[\sum_{i=1}^n w_i x_i \geq \epsilon\delta\sum_{i=1}^n x_i + \sum_{i=1}^n a_i x_i\right].
\end{aligned}$$

Now, let $C = \epsilon\delta\sum_{i=1}^n x_i + \sum_{i=1}^n a_i x_i$. We have $\epsilon = \frac{C-E[W(x)]}{\delta\sum_{i=1}^n x_i}$. We substitute $C$ and $\epsilon$ into the last expression, which completes the proof. □

The proof can be distinguished by the following two characteristics. On the one hand, we study the random variable $y_i$ rather than $w_i$, on the other hand, the interval lengths we discussed are the same for all stochastic weights. We reformulate the chance constraint by using the Chernoff bound to estimate the upper bound of the probability of violating the capacity. It should be noted that the interval of all weights should be the equivalent and equal to $2\delta$. The surrogate function of the chance constraint is as follows:

$$P_r\left(W(x) \geq C\right) \leq \left( \frac{e^{\frac{C-\sum_{i=1}^n a_i x_i}{\delta \sum_{i=1}^n x_i}}}{\left(\frac{\delta \sum_{i=1}^n x_i + C - \sum_{i=1}^n a_i x_i}{\delta \sum_{i=1}^n x_i}\right)^{\frac{\delta \sum_{i=1}^n x_i + C - \sum_{i=1}^n a_i x_i}{\delta \sum_{i=1}^n x_i}}} \right)^{\frac{1}{2}\sum_{i=1}^n x_i} \leq \alpha \quad (4.7)$$

**Theorem 4.2.3.** *If a solution $x$ is satisfied for the surrogate function by applying Chernoff bound, then it has the surrogate weight*

$$S'(x) = E[W(x)] + \delta \sqrt{-\frac{2}{\sum_{i=n}^n x_i} \ln \alpha} \sqrt{\sum_{i=n}^n x_i},$$

*and $S'(x) \leq C$.Let $w_i \in [a_i - \delta, a_i + \delta]$ for all $\alpha \in (0, 1)$.*

*Proof.* Let $\epsilon = \frac{C - E[W(x)]}{\delta \sum_{i=1}^n x_i} \geq 0$, by the surrogate function (4.6) obtained by the Chernoff bound, this is bounded above by

$$\begin{aligned}
&\left(\frac{e^\epsilon}{(1+\epsilon)^{(1+\epsilon)}}\right)^{\frac{\sum_{i=1}^n x_i}{2}} \leq \alpha \\
\iff & \frac{e^\epsilon}{(1+\epsilon)^{(1+\epsilon)}} \leq (\alpha)^{\frac{2}{\sum_{i=1}^n x_i}} \\
\iff & e^\epsilon \leq (1+\epsilon)^{(1+\epsilon)} (\alpha)^{\frac{2}{\sum_{i=1}^n x_i}} \\
\iff & \epsilon \leq \frac{2}{\sum_{i=1}^n x_i} \ln \alpha + (1+\epsilon) \ln(1+\epsilon) \\
\iff & \epsilon - (1+\epsilon) \ln(1+\epsilon) \leq \frac{2}{\sum_{i=1}^n x_i} \ln \alpha \\
\implies & \epsilon - (1+\epsilon)\epsilon \leq \frac{2}{\sum_{i=1}^n x_i} \ln \alpha \\
\iff & -\epsilon^2 \leq \frac{2}{\sum_{i=1}^n x_i} \ln \alpha \\
\iff & \epsilon^2 \geq -\frac{2}{\sum_{i=1}^n x_i} \ln \alpha \\
\iff & \epsilon \geq \sqrt{-\frac{2}{\sum_{i=1}^n x_i} \ln \alpha}.
\end{aligned}$$

Hence, we have $\frac{C-E[W(x)]}{\delta \sum_{i=1}^n x_i} \geq \sqrt{-\frac{2}{\sum_{i=1}^n x_i} \ln \alpha}$, and $C \geq E[W(x)] + \delta \sqrt{-\frac{2}{\sum_{i=1}^n x_i} \ln \alpha} \sqrt{\sum_{i=1}^n x_i}$ where we have used the claimed surrogate weight. $\square$

### 4.2.3 Comparison of tail inequalities

Next, we theoretically investigate the effectiveness of Chebyshev's inequality and the Chernoff bound for tackling the chance constraint. The goal of this analysis is to examine which estimation method outperforms the other under various conditions. Let $p_{\texttt{Cher}}(x)$ denotes the probability bound obtained by the Chernoff bound and $p_{\texttt{Cheb}}(x)$ be the probability bound calculated by the Chebyshev's inequality for a solution $x$. The following theorem states a condition under which one inequality should be preferred over the other.

**Theorem 4.2.4.** *Let $x$ be a solution with expected weight $E[W(x)]$ and variance of weight $Var(x)$. Let $\epsilon = \frac{C-E[W(x)]}{\delta\sum_{i=1}^{n} x_i}$. We have $p_{\texttt{Cher}}(x) \leq p_{\texttt{Cheb}}(x)$ if and only if*

$$\frac{\left(\frac{e^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{E[W(x)]}(\epsilon E[W(x)])^2}{1-\left(\frac{e^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{E[W(x)]}} \leq Var(x). \tag{4.8}$$

*Proof.* Using the variable $\epsilon$ from the Chernoff bound, we set $k = \epsilon E[W(x)]$ in the Chebyshev's inequality. Then, we have

$$\begin{aligned}
&\left(\frac{e^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{E[W(x)]} \leq \frac{Var(x)}{Var(x)+(\epsilon E[W(x)])^2}\\
\iff &\left(\frac{e^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{E[W(x)]}(Var(x)+(\epsilon E[W(x)])^2) \leq Var(x)\\
\iff &\left(\frac{e^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{E[W(x)]}(\epsilon E[W(x)])^2) \leq Var(x)\left(1-\left(\frac{e^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{E[W(x)]}\right)\\
\iff &\frac{\left(\frac{e^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{E[W(x)]}(\epsilon E[W(x)])^2}{1-\left(\frac{e^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{E[W(x)]}} \leq Var(x)
\end{aligned}$$

which demonstrates our claim. □

We now further investigate the relationship between Chebyshev's inequality and the Chernoff bound. In Theorem 4.2.4, the three parameters: $\epsilon, E[W(x)]$ and $Var(x)$ establish the relationship between Chebyshev's inequality and the Chernoff bound. Among the parameters, $\epsilon$ indicates the deviation from the expected value. After fixing the value of $\epsilon$, for any instance, the relationship between $E[W(x)]$ and $Var(x)$ can determine which inequality is more suitable for the purpose of solving the instance. As shown in Figure 4.1, the values of $\epsilon$ set independently at $\{0.01, 0.05, 0.1, 0.2, 0.3\}$. The figure is based on a test instance with 100 items, and the weights of items are chosen uniformly at random in the interval $[0, 1]$. Every curve in Figure 4.1 corresponds to a fixed value of $\epsilon$. When the tuple of $(E[W(x)], Var(x))$ is located on the curve, the probability of the constraint violation calculated by Chebyshev's inequality and the Chernoff bound are the same. For the situation where a tuple of $(E[W(x)], Var(x))$ is located above the curve, the Chernoff bound gives a better estimate, and if it is located below the curve, Chebyshev's inequality provides a better upper bound on the probability of the constraint violation. As can be seen from the figure, the greater the value of $\epsilon$, the more suitable the Chernoff bound is for obtaining a superior bound.

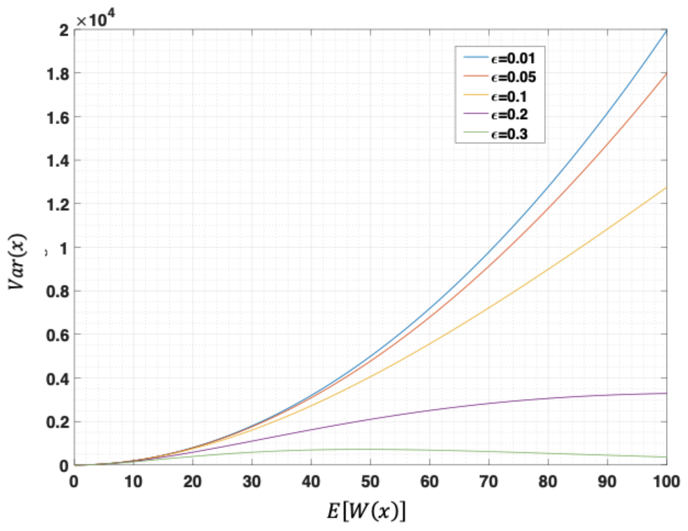


FIGURE 4.1. The relationship between expected weight and variance of weight of a solution based on different values of $\epsilon$.

## 4.3 Deterministic Approaches

In this section, we consider several deterministic approaches for the chance-constrained knapsack problem. We present Integer Linear Programming (ILP), a Nemhauser-Ullmann-based (NU-base) heuristic approach and dynamic programming (DP) in the following subsections.

### 4.3.1 Integer Linear Programming

Firstly, we linearise the surrogate functions which replace the chance constraint in the CCKP, then we apply Integer Linear Programming (ILP) to solve the problem. We consider Chebyshev's inequality and provide the linear approximation that characterises the ILP approach. The surrogate functions in Section 4.2.1 can be replaced by the following equations:

$$
\begin{aligned}
&\frac{\delta^2 \sum_{i=1}^n x_i}{\delta^2 \sum_{i=1}^n x_i + 3(C - \sum_{i=1}^n a_i x_i)^2} \leq \alpha \\
\iff & \delta^2 \sum_{i=1}^n x_i \leq \alpha\delta^2 \sum_{i=1}^n x_i + 3\alpha \left( C - \sum_{i=1}^n a_i x_i \right)^2 \\
\iff & \frac{1-\alpha}{3\alpha}\delta^2 \sum_{i=1}^n x_i \leq \left( C - \sum_{i=1}^n a_i x_i \right)^2 \\
\iff & \frac{1-\alpha}{3\alpha}\delta^2 \sum_{i=1}^n x_i \leq C^2 - 2C \sum_{i=1}^n a_i x_i + \left( \sum_{i=1}^n a_i x_i \right)^2 \\
\iff & \frac{1-\alpha}{3\alpha}\delta^2 \sum_{i=1}^n x_i \leq C^2 - 2C \sum_{i=1}^n a_i x_i + \sum_{i=1}^n a_i^2 x_i + 2 \sum_{i<j:i,j=\{1,\dots,n\}} a_i a_j x_i x_j \qquad (4.9)
\end{aligned}
$$

for cases in which the weights of items follow the additive uniform distribution (4.3). Then, we replace the term $x_i x_j$ in the right-hand side of equation 4.9 with a new added variable $y_{ij}$ with domain $\{0, 1\}$. We define linear constraints for $y_{ij}$ as follows:

$$2y_{ij} \leq x_i + x_j \leq 1 + y_{ij}; \quad \forall i < j. \tag{4.10}$$

The chance constraint can be reformulated as follows:

$$\begin{aligned}
&\frac{1-\alpha}{3\alpha}\delta^2 \sum_{i=1}^{n} x_i \leq C^2 - 2C\sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} a_i^2 x_i + 2 \sum_{i<j:i,j=\{1,\ldots,n\}} a_i a_j y_{ij} \\
\Longleftrightarrow &\frac{1-\alpha}{3\alpha}\delta^2 \sum_{i=1}^{n} x_i + 2C\sum_{i=1}^{n} a_i x_i - \sum_{i=1}^{n} a_i^2 x_i - 2 \sum_{i<j:i,j=\{1,\ldots,n\}} a_i a_j y_{ij} \leq C^2. \qquad (4.11)
\end{aligned}$$

For the chance-constrained knapsack problem, any feasible solution should submit to equation (4.11) which indicates that regardless of whether the value of $x_i x_j$ is equal to 0 or 1, setting $y_{ij}$ will not make the feasible solution infeasible. For example, if a solution $X$ is a feasible solution, it should satisfy equation (4.10), assuming there exists some $x_i x_j = 0$ and we set the corresponding $y_{ij} = 1$. So, we have $\sum_{i<j:i,j\in N} a_i a_j x_i x_j \leq \sum_{i<j:i,j\in N} a_i a_j y_{ij}$ of the solution $x$ which does not make the solution infeasible. Therefore, we can remove the right-hand side of equation (4.10). We then formulate the ILP model of the chance-constrained knapsack problem with weights having the additive uniform distribution as follows:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} p_i x_i & (4.12)\\
\text{s.t.} \quad & \frac{1-\alpha}{3\alpha}\delta^2 \sum_{i=1}^{n} x_i + 2C\sum_{i=1}^{n} a_i x_i - \sum_{i=1}^{n} a_i^2 x_i - 2 \sum_{i<j:i,j=\{1,\ldots,n\}} a_i a_j y_{ij} \leq C^2 & (4.13)\\
& 2y_{ij} \leq x_i + x_j & (4.14)\\
& x_i, y_{ij} \in \{0, 1\}^n & (4.15)
\end{aligned}$$

Similarly, for multiplicative uniform distribution (4.4), we have

$$\begin{aligned}
&\frac{\beta^2 \sum_{i=1}^{n} {a_i}^2 x_i}{\beta^2 \sum_{i=1}^{n} {a_i}^2 x_i + 3(C - \sum_{i=1}^{n} a_i x_i)^2} \leq \alpha \\
\Longleftrightarrow &\beta^2 \sum_{i=1}^{n} a_i^2 x_i \leq \alpha\beta^2 \sum_{i=1}^{n} x_i + 3\alpha\left(C - \sum_{i=1}^{n} a_i x_i\right)^2 \\
\Longleftrightarrow &\frac{(1-\alpha)}{3\alpha}\beta^2 \sum_{i=1}^{n} a_i^2 x_i \leq \left(C - \sum_{i=1}^{n} a_i x_i\right)^2 \\
\Longleftrightarrow &\frac{(1-\alpha)}{3\alpha}\beta^2 \sum_{i=1}^{n} a_i^2 x_i \leq C^2 - 2C\sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} a_i^2 x_i + 2 \sum_{i<j:i,j\in N} a_i a_j x_i x_j \qquad (4.16)
\end{aligned}$$

**Algorithm 10:** NU-base Heuristic approach

```
1: Input: $n$ items with expected weights $\{a_1, ..., a_n\}$, variances of weights
   $\{v_1, ..., v_n\}$ and profits of items $\{p_1, ..., p_n\}$. Knapsack capacity $C$.
2: let lists $L_1, .., L_n$.
3: Initialize $L_0 = \{(0, 0)\}$.
4: for $i = 1$ to $n$ do
5:    $L'_{i-1} = L_{i-1}$;
6:    for each solution $K \in L'_{i-1}$ do
7:       add item $i$ to $K$ to generate a new solution $K' = K \cup \{i\}$,
8:       if solution $K'$ is feasible then
9:          replace $K$ with $K'$.
10:      else
11:         delete $K$ from $L'_{i-1}$.
12:      end if
13:   end for
14:   let $P_{max} = -1$, $E = \{\}$
15:   while true do
16:      Let $k \in L_{i-1}$ be the first one with $p_k > P_{max}$,
17:      let $k' \in L'_{i-1}$ be the first one with $p_{k'} > P_{max}$,
18:      if cannot find $k$ then
19:         insert remaining points from $L'_{i-1}$ into $E$,
20:         break.
21:      end if
22:      if cannot find $k'$ then
23:         insert remaining points from $L_{i-1}$ into $E$,
24:         break.
25:      end if
26:      if $S(k) < S(k')$ or $(S(k) = S(k')$ and $p_k > p_{k'})$ then
27:         insert $k$ into $E$, and set $P_{max} = p_k$,
28:      else
29:         insert $k'$ into $E$, and set $P_{max} = p_{k'}$,
30:      end if
31:   end while
32: end for
33: return $L_n$.
```

and for the case that the weight of each item is chosen according to the Normal distribution $N(a_i, \sigma_i^2)$ (4.5), we have

$$\begin{aligned}
&\frac{\sum_{i=1}^{n} \sigma_i^2 x_i}{\sum_{i=1}^{n} \sigma_i^2 x_i + (C - \sum_{i=1}^{n} a_i x_i)^2} \leq \alpha \\
\Longleftrightarrow &\sum_{i=1}^{n} \sigma_i^2 x_i \leq \alpha \sum_{i=1}^{n} \sigma_1^2 x_i + \alpha \left( C - \sum_{i=1}^{n} a_i x_i \right)^2 \\
\Longleftrightarrow &\frac{(1-\alpha)}{\alpha} \sum_{i=1}^{n} \sigma_i^2 x_i \leq \left( C - \sum_{i=1}^{n} a_i x_i \right)^2 \\
\Longleftrightarrow &\frac{(1-\alpha)}{\alpha} \sum_{i=1}^{n} \sigma_i^2 x_i \leq C^2 - 2C \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} a_i^2 x_i + 2 \sum_{i<j:i,j \in N} a_i a_j x_i x_j. \qquad (4.17)
\end{aligned}$$

Replacing the terms $x_i x_j$ in equations (4.16) and (4.17) with the added variables $y_{ij}$, we can obtain the ILP models which take into account different surrogate constraints.

### 4.3.2 Heuristic Approach

Now, we introduce a heuristic approach (see Algorithm 10) adapted from the Nemhauser-Ullmann algorithm (NU algorithm) proposed by Nemhauser and Ullmann (1969). The NU algorithm can be viewed as a sparse dynamic programming approach introduced by Beier and Vöcking (2004), and it computes a list of all dominating sets in an iterative manner, adding one item after the other. For $i \in \{1, \dots, n\}$, let $L_i$ denote the list of Pareto-optimal points considering item 1 to $i$. The solutions in $L_i$ are assumed to be listed in increasing order of weight (profit).

In the heuristic approach, we use surrogate weights obtained using Chebyshev's inequality and the Chernoff bound introduced in Definition 4.2.1 and Theorem 4.2.3 to replace the exact weights of the solution used to find the Pareto front in the NU algorithm. The heuristic approach starts with the empty set and then adds items one by one until it finally obtains the Pareto-optimal packing for all $n$ items. $L_i$ can be computed using $L_{i-1}$ and the item $i$ as follows: first generate $L'_{i-1}$ by adding item $i$ to each element in $L_{i-1}$ if the new solution is feasible and inserting it to $L'_{i-1}$. Then, we merge the two lists $L_{i-1}$ and $L'_{i-1}$ according to the surrogate weight and the profit of the solutions. Finally, we obtain an order sequence $L_i$ of dominating solutions over the items $1, \dots, i$. The resulting list $L_n$ contains all Pareto-optimal points for $n$ items. For this list, we choose the point with maximal profit, and the packing belonging to this result is the approximate optimal solution.

In the merging step, both lists are sorted according to the surrogate weights. Thus, this task can be completely by scanning only once through both of these lists. However, this heuristic approach is effective in solving the knapsack problem where the weights of items are deterministic in value. In the chance-constrained knapsack problem, the weights of items are stochastic variables. Using the surrogate weights and the profits of solutions to find the Pareto front does not guarantee that the optimal solution to the problem will be found. Therefore, we introduce dynamic programming for the chance-constrained knapsack problem in the next subsection.

### 4.3.3 Dynamic programming

We now introduce a dynamic programming approach for the purpose of solving the chance-constrained knapsack problem. Dynamic programming is one of the traditional approaches to the classical binary knapsack problems (Toth, 1980). In the DP approach, items are processed in order according to their index, from 1 to $n$.

The key idea behind the DP approach for CCKP is to assume the weights of items are random variables with corresponding expected weights and variances. The approach is applied in a similar manner to the process which is undertaken for the classical binary knapsack problem. The program table of the chance-constrained knapsack problem consists of $n+1$ rows and $C+1$ columns which are used to compute the optimal solution. Each cell $M_{ij}$ consists of a set of feasible solutions which selected items from the items set $\{1, \dots, i\}$ and knapsack capacity $j$, and those solutions are not dominated by each other. Here we choose to use the surrogate functions to replace the chance constraint. To initialise the program table, we set the first cell as $M_{00} = \{\emptyset\}$ which only contains an empty set of items.

It can be observed of the surrogate functions obtained by Chebyshev's inequality (4.1) and the Chernoff bound (4.7) that for a solution, not only its expected weight but also its variance affects the probability with which it will violate the knapsack bound. With a fixed expected weight, when the value of the variance decreases, the probability of the chance constraint will decrease. Similarly, with a fixed variance, a decrease in

**Algorithm 11:** Dynamic programming

```
1: Input: $n$ items with expected weights $\{a_1, ..., a_n\}$, variances of weights
   $\{v_1, ..., v_n\}$ and profits of items $\{p_1, ..., p_n\}$. Knapsack capacity $C$.
2: let $M[0, \dots, n][0, \dots, C]$ be a new table, each cell $M_{ij}$ stores a set of solutions.
3: Initialize $M_{00} = \{\emptyset\}$.
4: for $i = 1$ to $n$ do
5:    for $j = 0$ to $C$ do
6:       $M_{ij} = M_{(i-1)j}$;
7:       if $a_i \leq j$ then
8:          for each solution $S \in M_{(i-1)(j-a_i)}$ do
9:             add item $i$ to $S$ to generate a new solution $S' = S \cup \{i\}$,
10:            if solution $S'$ is feasible and not dominated by any solution in $M_{ij}$
               then
11:               remove solutions in $M_{ij}$ which are dominated by $S'$.
12:               add $S'$ to $M_{ij}$.
13:            end if
14:         end for
15:      end if
16:   end for
17:   $M_{ij}$ stores all feasible and no-dominate to each other solutions.
18: end for
```

the expected weight leads to a decrease of probability that chance constraint will be violated. The smaller the value of this probability, the higher the capability to insert items into the knapsack. Therefore, we say that solution $S$ dominates solution $S'$, denoted by $S \succeq S'$, iff $w(S) < w(S'), v(S) < v(S')$ and $p(S) > p(S')$, where $w(S)$ denotes the expected weight of solution $S$, $v(S)$ denotes the variance of solution $S$ and $p(S)$ denotes the profit of solution $S$.

Let item $(i-1)$ be the predecessor of item $i$ and $a_i \leq j$. Based on the set of feasible solutions in $M_{(i-1)(j-a_i)}$ we compute $M_{ij}$ where $a_i$ denotes the expected weight of item $i$, and giving us $a_i \leq j$. We calculate $M_{ij}$ by adding item $i$ using the following steps. Firstly, all elements from $M_{(i-1)(j)}$ are copied to $M_{ij}$. Secondly, for every solution $S$ in $M_{(i-1)(j-a_i)}$, item $i$ is added to the set of items. If the new set of items $S \cup \{i\}$ is feasible and not dominated by any solution in $M_{ij}$, we remove solutions from $M_{ij}$ which are dominated by $S \cup \{i\}$ and add $S \cup \{i\}$ to $M_{ij}$. For the resulting algorithm (Algorithm 11), we can state the following theorem:

**Theorem 4.3.1.** *For each set of item $\{1, .., i\}$, $M_{ij}$ stores a set of feasible solutions none of which are dominated by each other with respect to all subsets of $\{1, ..., i\}$ and knapsack capacity $j$, and the optimal solution is among them. In particular, $M_{nC}$ contains the optimal solution with considering all items, which can be obtained using the DP approach.*

*Proof.* The statement is true for the first item as there are only two options: choosing or not choosing the first item. So $M_{00}$ stores the solution: $(\emptyset)$. Now we assume that $M_{(i-1)j}$ stores all feasible solutions which are not dominated by each other with respect to all subsets of $\{1, .., i-1\}$ with capacity $j$.

Now, we construct $M_{ij}$ by first adding all subsets in $M_{(i-1)j}$. Then, if $a_i > j$, item $i$ cannot be added to any solution in $M_{(i-1)(j-a_i)}$ and $M_{ij} = M_{(i-1)j}$. Otherwise, adding item $i$ to a subset $S' \in M_{(i-1)(j-a_i)}$, if the new set of items is still feasible according to capacity $j$, then the expected weight of this solution is $w(S' \cup \{i\}) = w(S') + a_i$, the variance of this solution is $v(S' \cup \{i\}) = v(S') + \sigma_i^2$ and the profit is $p(S' \cup \{i\}) = p(S') + p_i$. Then, if the new solution is not dominated by any solution in $M_{ij}$, we insert this solution into $M_{ij}$, removing all solutions in $M_{ij}$ which are dominated by this solution.

Therefore, $M_{ij}$ stores all feasible solutions which are not dominated by each other, and we can pick the optimal solution in $M_{nC}$. This concludes the proof. $\square$

We now investigate the runtime for this dynamic program. The feasible solutions in the cell $M_{ij}$ have been calculated by considering all possible combinations of variances and expected weights from the set $\{1, ..., i\}$, in which $2^i$ denotes the number of different expected weights and variances in the worst case. We then give the upper bound of the computation time that DP takes to calculate all possible combinations of the variances and expected weights of $M_{ij}$ as $O(2^{2i})$. Therefore, the time until the DP approach calculates the optimal solution to the chance-constrained knapsack problem instance is the summary of $O(nC2^{2n})$. However, in the case that the weights of items are chosen according to a uniform distribution and take values in $[a_i - \delta, a_i + \delta]$, then the variance of items are the same. So for the set $\{1, \ldots, i\}$, the possible combinations of the variance is $O(i)$, and the runtime of the instances, in this case, is bounded by $O(Cn^2 2^n)$. In the case that the weights of items are chosen according to the uniform distribution $[(1-\beta)a_i, (1+\beta)a_i]$ or the Normal distribution $N(a_i, a_i\beta)$, the variance of an item is equal to the expected weight of this item times the uncertainty $\beta$. Therefore, when tackling those cases, the DP does not need to incorporate the variance of solutions into the domination comparison, and the runtime of the approach is bounded by $O(Cn2^n)$.

## 4.4 Evolutionary Algorithms

In this section, we discuss the use of evolutionary algorithms to solve the CCKP. We begin by designing fitness functions for a single-objective approach and a multi-objective approach. Next, we investigate the effectiveness of the simplest single-objective evolutionary algorithm (the (1+1) EA) and its multi-objective version (GSE MO) to solve the problem using an experimental study.

Experimental results are presented in Section 4.5.2, which show that the GSEMO outperforms the (1+1) EA for all instances and performs better than deterministic approaches in terms of computation time. A new multi-objective model is then presented and compared with the standard model. Moreover, we develop specific operators that can be used in single- and multi-objective evolutionary algorithms to improve their performance of EAs.

### 4.4.1 Single-Objectives Approach

In this section, we present the single-objective approach and design a fitness function that can be used in single-objective evolutionary algorithms. The fitness function $f$ for the approach needs to take into account the profit of the selected items and the requirement that the probability of overloading the capacity must to meet the threshold of chance constraint.

We define the fitness of a solution $x \in \{0, 1\}^n$ as

$$f(x) = (u(x), v(x), P(x)) \tag{4.18}$$

where $u(x) = \max\{\sum_{i=1}^{n} a_i x_i - C, 0\}$, $v(x) = \max\{P_r\{\sum_{i=1}^{n} w_i x_i > C\} - \alpha, 0\}$ where the probability is calculated by surrogate functions, and $P(x) = \sum_{i=1}^{n} p_i x_i$. For this fitness function, $u(x)$ and $v(x)$ need to be minimised while $P(x)$ needs to be maximised, and we optimise $f$ in lexicographic order. The fitness function takes into account two types of infeasible solutions: (1) solutions the expected weight of which exceeds the bounds of the capacity, (2) solutions for which the probability that the total weight of the solution will violate the capacity is greater than $\alpha$. Note that $\alpha$ is usually a small value, and that throughout this paper, we work with $\alpha \leq 0.1$. The reason we account for the first type of infeasible solution is that we cannot use the inequalities to guide the search if the expected weight is above the given capacity bound. Furthermore, the first type of infeasible solution violates the chance constraint with a probability of at least 1/2 for all probability distributions studied in the experimental part of this paper. Among solutions that meet the chance constraint, we maximise the profit $P(x)$.

Formally, we have the following relation between two search points $x$ and $y$

$$\begin{aligned} & f(x) \succeq f(y) \\ \iff & (u(x) < u(y)) \quad \text{or} \quad (u(x) = u(y) \wedge v(x) < v(y)) \\ & \text{or} \quad (u(x) = u(y) \wedge v(x) = v(y) \wedge P(x) > P(y)) \end{aligned} \tag{4.19}$$

When comparing solutions, a feasible solution is preferred to any infeasible. When comparing two feasible solutions, the one with the better profit is preferred. When comparing two infeasible solutions, the one with a lower degree of constraint violation is better than the other.

The fitness function can be used in any single-objective evolutionary algorithm. In this chapter, we investigate the performance of the classical (1+1) EA (see Algorithm 6). We generate the initial solution with items chosen uniformly at random for the algorithm, and then the (1+1) EA flips each bit of the current solution with a probability of $1/n$ in the mutation step. In the selection step, the algorithm accepts the offspring if it is at least as good as the parent according to the fitness function $f$.

### Problem-Specific Crossover Operator

The proposed crossover operator is a combination of the uniform crossover and factors specific to CCKP and the standard KP. The uniform crossover operator can easily preserve all parent similarities when generating new offspring. Indeed, for many combinatorial optimisation problems, good solutions which are close in the objective space are expected to be rather similar in the decision space (Jaszkiewicz, 2001; Ishibuchi et al., 2008). Therefore, the uniform crossover can maintain so-called good gene combinations which are constructed during the search process.

The problem-specific crossover operator, which we shall call the PS crossover operator, adopts the benefits of the uniform crossover operator. We evaluate the quality of all genes that are different in the two parents, specifically in relation to KP. We use the *profit/weight* ratio to determine the quality of genes (i.e., items). Then, the genes are sorted in descending order according to the quality, and we apply a greedy insertion heuristic to iteratively insert the candidate item that has the highest profit/weight ratio. At this stage, we insert the first $k$ items according to the ordering of the non-common genes. $k$ is an integer number that round the value which randomly chosen according to the Normal distribution with expectation and variance both equal to $\frac{m}{2}$, where $m$ denotes the number of genes on which the two parents differ.

**Algorithm 12:** $(\mu + 1)$ EA

```
1: Randomly generate $\mu$ initial solutions as the initially population;
2: while stopping criterion not meet do
3:     Choose $x_1 \in \{0,1\}^n$ and $x_2 \in \{0,1\}^n$ uniformly at random from the
       population $X$; $x_1 \neq x_2$.
4:     apply the PS crossover operator in $x_1$ and $x_2$, generate an offspring $y$;
5:     apply the heavy-tail mutation operator to $y$;
6:     if $y$ is better than the worse solution in $X$ then
7:         replace the worst solution with $y$;
8:     end if
9: end while
```

In this chapter, to investigate the effectiveness of the presented operators, we introduce a population-based single-objective evolutionary algorithm. This kind of algorithm maintains a population of binary solutions presented as a bit string. We examine the performance of this $(\mu+1)$ EA (see Algorithm 12) using the heavy-tail mutation operator and the problem-specific crossover operator separately, as well as a combination of the two operators.

### 4.4.2 Standard Multi-Objective Model

To compare the performance between single-objective evolutionary algorithms and multi-objective evolutionary algorithms, we set the search point $x$ as a two-dimensional point in the objective space. Then, we have the following standard fitness functions which aim to obtain a solution with maximal profit and minimal probability of violating the capacity bound.

$$g_1(x) = \begin{cases} P_r(W(x) \geq C) & E[W(x)] < C \\ 1 + (E[W(x)] - C) & E[W(x)] \geq C \end{cases} \tag{4.20}$$

$$g_2(x) = \begin{cases} P(x) & g_1(x) \leq \alpha \\ -1 & g_1(x) > \alpha \end{cases} \tag{4.21}$$

where $W(x)$ denotes the weight of the solution $x$ and $E[W(x)]$ denotes the expected weight of the solution. We say that solution $y$ weakly-dominates solution $x$ (denoted by $y \succcurlyeq x$), **iff** $g_1(y) \leq g_1(x) \wedge g_2(y) \geq g_2(x)$. When comparing two solutions, the objective function $g_1$ guarantees that a feasible solution dominates all infeasible solutions. The objective function $g_2$ ensures that the search process is guided towards feasible solutions and that trade-offs in terms of confidence level and profit are computed for feasible solutions. The multi-objective algorithm we consider with the standard fitness functions is the GSEMO (see Algorithm 7) which is a simple multi-objective evolutionary algorithm (SEMO) that searches globally.

### 4.4.3 Improved Multi-Objective Model

We introduce a new multi-objective model for the chance-constrained knapsack problem. The model accounts for both feasible solutions and the second type of infeasible solutions mentioned in Section 4.4.2. To keep more diversity in the solution space, the new model makes other solutions dominate the infeasible solutions such that the expected weight of selection items is overloading the capacity. The difference between the improved multi-objective model and the standard multi-objective model is that the standard model made all feasible solutions dominate all infeasible solutions. The fitness functions of this new model are proposed as follows:

$$g'_1(x) = \begin{cases} P_r(W(x) \geq C) & E[W(x)] < C \\ 1 + (E[W(x)] - C) & E[W(x)] \geq C \end{cases} \tag{4.22}$$

$$g'_2(x) = \begin{cases} \sum_{i=1}^{n} p_i x_i & g_1(x) \leq 1 \\ -1 & g_1(x) > 1 \end{cases} \tag{4.23}$$

The first function calculates the probability of a solution by overloading the capacity of the knapsack, and it forces the probability of an infeasible solution whose expected weight exceeds the capacity of 1. The second fitness function is the objective of the chance-constrained knapsack problem. It calculates the profit of feasible solutions with a probability less than $\alpha$ and infeasible solutions with a probability greater than $\alpha$ but less than 1. We say solution $y$ dominates solution $x$ w.r.t.$g$, denoted by $y \succcurlyeq x$, iff $g'_1(y) \leq g'_1(x) \wedge g'_2(y) \geq g'_2(x)$.

The objective function $g_2$ guarantees that the search process is guided towards all available solutions, meaning that trade-offs in terms of confidence level and profit are computed for the solutions in the Pareto front. However, even though the algorithm can store feasible solutions and a collection of infeasible solutions, it outputs the best feasible solution in every iteration.

In the experiment section, we apply the improved model to GSEMO. GSEMO can generate a Pareto front with both feasible solutions and infeasible solutions. For further investigation of our multi-objective optimisation, we also apply the nondominated sorting genetic algorithm II (NSGA-II; see Algorithm 8), which was introduced in Section 3.3.3.

## 4.5 Experiments

In this section, we first compare the results obtained by using deterministic approaches, (1+1) EA and GSEMO, and investigate the performance of these approaches with different surrogate functions of the chance constraint. Secondly, to show the differences between the ways in which the EAs use different operators, we carry out a test into combining algorithms with operators for the purpose of solving the CCKP instances.

### 4.5.1 Experimental Setup

Firstly, we describe the experimental design and the chance-constrained knapsack problem instances. In this chapter, all experiments implemented evolutionary algorithms, the heuristic approach and DP were performed using Java (version 11.0.1). Investigations implementing ILP were conducted in Gurobi (version 9) solver on a MacBook with a 2.3 GHz Intel Core i5 CPU. The benchmark we used is from the literature (Roostapour, Neumann, and Neumann, 2018), which was created following the approach in Vance (1993). We choose two types of instances from the benchmark: *uncorrelated* and *bounded strongly correlated*. The weights and profits of items in the uncorrelated instances are integers that are chosen uniformly at random within

$[1, 1000]$. The bounded strongly correlated instances have the tightest bound of knapsack among all type of instances where the weights of items are chosen uniformly at random within $[1, 1000]$, and the values of profits are set by the weights.

We adapt the above instances to the chance-constrained knapsack problem by randomising the weights. Since the weights of items must be positive, we add a value $\gamma$ to every deterministic weight from the benchmark and take it as the expected weight of this item to ensure we can factor in more uncertainty in all instances. Since we change the weights of items, we need to adjust the considered constraint bound. However, shifting the knapsack bound is challenging, as it is necessary to ensure that a solution remains feasible after bound has been changed. Moreover, increasing the knapsack's capacity expands the feasible search space and may introduce additional feasible solutions. Hence, when shifting the capacity of the knapsack, one should consider keeping the feasibility of original solutions and the size of the new feasible search space adaptive.

We adjust the original knapsack problem instances from the benchmark set as follows. First, we sort the weights of items in ascending order. Then, the first $k$ items with smaller weights are chosen to be added to the knapsack until the original capacity is exceeded. Hence, this number of items $k$ represents the largest number of items that any feasible solution may include. We adapt the capacity bound according to this and set:

$$C' \leftarrow C + k\gamma. \tag{4.24}$$

We set $\gamma = 100$ and apply it to the initial benchmark. In this section, we consider three instance categories: (1) instances in which every item weight is chosen according to the additive uniform distribution and takes the value in $[a_i - \delta, a_i + \delta]$; (2) instances in which every item weight follows the multiplicative uniform distribution and takes the value in $[(1-\beta)a_i, (1+\beta)a_i]$; (3) instances in which every item weight follows the Normal distribution $N \sim (a_i, a_i\beta)$. The tuples $(\alpha, \delta, \beta)$ are the combinations of the elements from the sets $\alpha = [0.001, 0.01, 0.1]$, $\delta = [25, 50]$ and $\beta = [0.01, 0.05, 0.1]$. Based on this arrangement, we compare the performance of all the algorithms on the chance-constrained knapsack problem. Since (1+1) EA and GSEMO are bio-inspired algorithms, they cannot produce exact optimal solutions, and the solutions are different in independent runs. Statistical comparisons (comparing each pair of algorithms with surrogate functions) are carried out using the Kruskal-Wallis test (with a 95% confidence interval) integrated with the Wilcoxon sum rank test (with a 95% confidence level). For more detail of these statistical tests, we refer to Dunn and Dunn (1961), Driscoll (1996), Ghasemi and Zahediasl (2012), and Corder and Foreman (2014).

In the next subsection, we compare the performance of all proposed algorithms for instances of different types and sizes. Then, we highlight the differences between the algorithms using Chebyshev's inequality and the Chernoff bound as the surrogate functions of the chance constraint.

### 4.5.2 Experimental Results of Deterministic Approaches and Baseline Evolutionary Algorithms

We benchmark our approach with the combinations from the experimental setting described above. Table 4.1 and Table 4.2 list the results obtained from the two

Table 4.1. Statistical results for instances with 100 items based on additive uniform distribution and using Chebyshev's inequality

| | Capacity | $\delta$ | $\alpha$ | ILP | | | Heuristic approach | | DP | | (1+1)EA | | GSEMO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 2mins | 6mins | 10mins | profit | time(ms) | profit | time(ms) | profit | time(ms) | profits | time(ms) |
| bou-s-c 1 | 11775 | 25 | 0.001 | 13701 | 13701 | 13701 | **13707** | 261 | – | – | 13614.8 | 1200.521 | **13707** | 21090.318 |
| | | | 0.01 | **15252** | **15252** | **15252** | **15252** | 303 | – | – | 15150.47 | 1207.292 | **15252** | 19888.28 |
| | | | 0.1 | 15768 | 15775 | 15775 | **15782** | 492 | – | – | 15680.87 | 1206.771 | **15782** | 26239.498 |
| | | 50 | 0.001 | 11757 | 11793 | 11795 | **11900** | 517 | – | – | 11756.27 | 1195.313 | 11888.1 | 11637.863 |
| | | | 0.01 | 14503 | **14505** | **14505** | **14505** | 691 | – | – | 14416.8 | 1203.125 | **14505** | 16181.898 |
| | | | 0.1 | **15585** | **15585** | **15585** | **15585** | 304 | – | – | 15431.37 | 1208.854 | **15585** | 18520.158 |
| bou-s-c 2 | 31027 | 25 | 0.001 | 35045 | 35045 | 35045 | **35069** | 2015 | – | – | 34874.8 | 1226.042 | 35068.933 | 51648.881 |
| | | | 0.01 | 37005 | 37005 | 37005 | **37027** | 2287 | – | – | 36850.73 | 1232.813 | 37019.133 | 61361.251 |
| | | | 0.1 | 37647 | 37647 | 37657 | **37673** | 5056 | – | – | 37467.43 | 1233.854 | 37670.367 | 64439.556 |
| | | 50 | 0.001 | 32096 | 32270 | 32357 | 32547 | 2092 | – | – | 32332.97 | 1220.833 | **32552.733** | 42825.912 |
| | | | 0.01 | 36019 | 36033 | 36061 | **36131** | 2172 | – | – | 35937.9 | 1228.125 | 36121.667 | 58301.656 |
| | | | 0.1 | 37391 | 37391 | 37391 | **37406** | 2857 | – | – | 37202.63 | 1235.938 | 37370.9 | 62496.325 |
| bou-s-c 3 | 58455 | 25 | 0.001 | 64190 | 64175 | 64265 | **64389** | 8042 | – | – | 64185.73 | 1250 | 64387.067 | 203302.72 |
| | | | 0.01 | 66630 | 66630 | 66630 | **66641** | 10749 | – | – | 66404.37 | 1253.125 | 66639.9 | 154383.75 |
| | | | 0.1 | 67339 | 67339 | 67339 | **67357** | 31192 | – | – | 67164.5 | 1256.771 | 67356.733 | 162281.6 |
| | | 50 | 0.001 | 61111 | 61111 | 61111 | 61155 | 8374 | – | – | 61007.6 | 1242.709 | **61220.2** | 168099.53 |
| | | | 0.01 | 65478 | 65496 | 65491 | **65603** | 8420 | – | – | 65372.4 | 1250.521 | 65601.8 | 200503.73 |
| | | | 0.1 | 66953 | 67001 | 66953 | **67059** | 17751 | – | – | 66859.37 | 1254.688 | 67057.3 | 213127.4 |
| uncorr 1 | 7715 | 25 | 0.001 | **14354** | **14354** | **14354** | **14354** | 128 | **14354** | 12605162 | 14273.6 | 1205.729 | **14354** | 11479.932 |
| | | | 0.01 | **16481** | **16481** | **16481** | **16481** | 120 | **16481** | 23461453 | 16433.1 | 1213.021 | **16481** | 13291.141 |
| | | | 0.1 | **17247** | **17247** | **17247** | **17247** | 153 | **17247** | 49890459 | 17176.57 | 1213.021 | **17247** | 13480.268 |
| | | 50 | 0.001 | **11599** | **11599** | **11599** | **11599** | 53 | **11599** | 33147184 | 11478.83 | 1196.875 | **11599** | 11852.237 |
| | | | 0.01 | **15504** | **15504** | **15504** | **15504** | 188 | **15504** | 205023878 | 15424.1 | 1207.292 | **15504** | 16188.438 |
| | | | 0.1 | **16890** | **16890** | **16890** | **16890** | 134 | **16890** | 36045784 | 16814.53 | 1214.583 | **16890** | 14843.431 |
| uncorr 2 | 19545 | 25 | 0.001 | 27027 | 27027 | 27027 | **27029** | 601 | – | – | 26932.67 | 1232.292 | **27029** | 38636.347 |
| | | | 0.01 | 28786 | **28825** | **28825** | **28825** | 879 | – | – | 28724.13 | 1238.021 | **28825** | 54770.144 |
| | | | 0.1 | **29415** | **29415** | **29415** | **29415** | 853 | – | – | 29315.6 | 1241.667 | **29415** | 56148.136 |
| | | 50 | 0.001 | 24561 | 24561 | 24561 | **24565** | 625 | – | – | 24449.3 | 1228.125 | **24565** | 32166.648 |
| | | | 0.01 | 27962 | 27962 | 27962 | **27985** | 504 | – | – | 27918.43 | 1237.5 | **27985** | 40521.42 |
| | | | 0.1 | **29165** | **29165** | **29165** | **29165** | 648 | – | – | 29091.97 | 1238.542 | **29165** | 43018.24 |
| uncorr 3 | 36091 | 25 | 0.001 | 39181 | 39182 | 39182 | **39245** | 1150 | – | – | 39192.53 | 1258.333 | **39245** | 32204.292 |
| | | | 0.01 | **40581** | **40581** | **40581** | **40581** | 998 | – | – | 40530.27 | 1261.458 | **40581** | 46855.019 |
| | | | 0.1 | **40991** | **40991** | **40991** | **40991** | 1300 | – | – | 40890.8 | 1262.5 | 40990.8 | 44955.924 |
| | | 50 | 0.001 | 36531 | 37068 | 37098 | **37180** | 1299 | – | – | 37120.03 | 1256.25 | **37180** | 45597.177 |
| | | | 0.01 | 39739 | 39917 | 39917 | **39960** | 1186 | – | – | 39906.7 | 1260.938 | **39960** | 34677.722 |
| | | | 0.1 | 40754 | **40811** | **40811** | **40811** | 1315 | – | – | 40751.47 | 1263.542 | **40811** | 38348.789 |

types of instances which contain 100 items. The weights of items have an additive uniform distribution, and the best solutions among all approaches are emphasised in bold. For each instance, we investigate different settings together with different levels of uncertainty determined by $\delta$ and the requirement on the chance constraint determined by $\alpha$. We apply Chebyshev's inequality to ILP by fixing running time $\{2\text{mins}, 6\text{mins}, 10\text{mins}\}$ for all instances; the results are listed in Table 4.1. We use both chance-constrained estimation methods to tackle the chance constraint when using the heuristic approach and the DP approaches. Under the heuristic approach and the DP headings, the *profit* denotes the object value of each approach, and the *time(ms)* denotes the computation time associated with each approach. The computation time associate with DP is one or orders of magnitude longer than for other approaches, for all instances. Where units are presented in $-$, this means that the run time for DP to solve these instances is longer than ten hours. For (1+1) EA and GSEMO, we provide the results from 30 independent runs with $10^6$ generations for all instances. In these cases, *profit* denotes the average profit associated with the 30 runs, and *time(ms)* denotes the average running time in milliseconds associated with the 30 runs.

The first insight from Table 4.1 and Table 4.2 is according to the deterministic approaches, the objective values obtained by the DP are at least as good as the results obtained by heuristic approach and ILP. When using Chebyshev's inequality to estimate the probability of violating the capacity, the heuristic approach performs at least

TABLE 4.2. Statistical results for instances with 100 items based on additive uniform distribution and using Chernoff bound

| | Capacity | $\delta$ | $\alpha$ | Heuristic approach | | DP | | (1+1)EA | | GSEMO | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | profit | time(ms) | profit | time(ms) | profit | time(ms) | profits | time(ms) |
| bou-s-c 1 | 11775 | 25 | 0.001 | **15208** | 293 | – | – | 15104.57 | 1236.979 | **15208** | 20055.67 |
| | | | 0.01 | **15348** | 613 | – | – | 15232.07 | 1236.458 | **15348** | 25803.75 |
| | | | 0.1 | **15599** | 1116 | – | – | 15454.03 | 1231.25 | **15599** | 20023.96 |
| | | 50 | 0.001 | 14367 | 715 | – | – | 14298.13 | 1243.75 | **14406** | 23056.49 |
| | | | 0.01 | **14742** | 556 | – | – | 14613.8 | 1240.104 | **14742** | 16970.46 |
| | | | 0.1 | **15144** | 317 | – | – | 15011.27 | 1235.417 | **15144** | 17916.35 |
| bou-s-c 2 | 31027 | 25 | 0.001 | 36893 | 2545 | – | – | 36824.33 | 1264.063 | **37027** | 42719.51 |
| | | | 0.01 | **37219** | 9410 | – | – | 36980.27 | 1232.012 | **37219** | 44305.01 |
| | | | 0.1 | **37437** | 4156 | – | – | 37241.03 | 1258.854 | **37437** | 48447.63 |
| | | 50 | 0.001 | **36071** | 1910 | – | – | 35870.83 | 1268.229 | 36060.73 | 55605.18 |
| | | | 0.01 | **36423** | 2334 | – | – | 36250.57 | 1226.667 | 36416.27 | 60741.89 |
| | | | 0.1 | **36893** | 3088 | – | – | 36712.6 | 1346.067 | 36888.93 | 57344.08 |
| bou-s-c 3 | 58455 | 25 | 0.001 | 65983 | 14478 | – | – | 66404.37 | 1289.063 | **66635** | 174822.8 |
| | | | 0.01 | 66175 | 13885 | – | – | 66612.5 | 1286.458 | **66840.87** | 119783.3 |
| | | | 0.1 | 66407 | 6809 | – | – | 66853.47 | 1286.979 | **67095** | 123643.2 |
| | | 50 | 0.001 | 64972 | 20041 | – | – | 65304.87 | 1292.708 | **65552.8** | 162435.2 |
| | | | 0.01 | 65355 | 10396 | – | – | 65755.47 | 1292.188 | **65952.93** | 165467.6 |
| | | | 0.1 | 65879 | 11290 | – | – | 66270.53 | 1291.667 | **66451** | 174008.9 |
| uncorr 1 | 7715 | 25 | 0.001 | **16432** | 129 | **16432** | 169791803 | 16342.87 | 1255.729 | **16432** | 12182.77 |
| | | | 0.01 | **16638** | 136 | **16638** | 194176080 | 16575.9 | 1240.625 | **16638** | 11075.88 |
| | | | 0.1 | **16945** | 144 | **16945** | 216867406 | 16833.13 | 1235.938 | **16945** | 10966.21 |
| | | 50 | 0.001 | **15299** | 98 | **15299** | 150530586 | 15243.03 | 1254.688 | **15299** | 15636.04 |
| | | | 0.01 | **15812** | 161 | **15812** | 161619933 | 15711.03 | 1253.125 | **15812** | 16543.41 |
| | | | 0.1 | **16362** | 142 | **16362** | 174426689 | 16287.27 | 1245.313 | **16362** | 18197.93 |
| uncorr 2 | 19545 | 25 | 0.001 | **28786** | 634 | – | – | 28690.93 | 1275 | **28786** | 29056.24 |
| | | | 0.01 | **28982** | 839 | – | – | 28864.17 | 1271.354 | **28982** | 37327.55 |
| | | | 0.1 | **29165** | 496 | – | – | 29090.6 | 1267.708 | **29165** | 38588.23 |
| | | 50 | 0.001 | **27962** | 469 | – | – | 27824.13 | 1279.688 | **27962** | 37555.58 |
| | | | 0.01 | **28276** | 793 | – | – | 28165.67 | 1279.167 | **28276** | 38640.84 |
| | | | 0.1 | **28732** | 823 | – | – | 28618.13 | 1276.042 | **28732** | 39601.95 |
| uncorr 3 | 36091 | 25 | 0.001 | **40581** | 935 | – | – | 40539.3 | 1305.729 | **40581** | 17069.47 |
| | | | 0.01 | **40663** | 983 | – | – | 40639.67 | 1303.646 | **40663** | 24246.41 |
| | | | 0.1 | **40830** | 1439 | – | – | 40741.6 | 1300.521 | **40830** | 40830 |
| | | 50 | 0.001 | **39960** | 1330 | – | – | 39894.23 | 1307.292 | **39960** | 51009.26 |
| | | | 0.01 | **40150** | 1505 | – | – | 40123.97 | 1306.25 | 40149.4 | 40985.79 |
| | | | 0.1 | **40482** | 1246 | – | – | 40422.3 | 1306.771 | 40481.67 | 39895.53 |

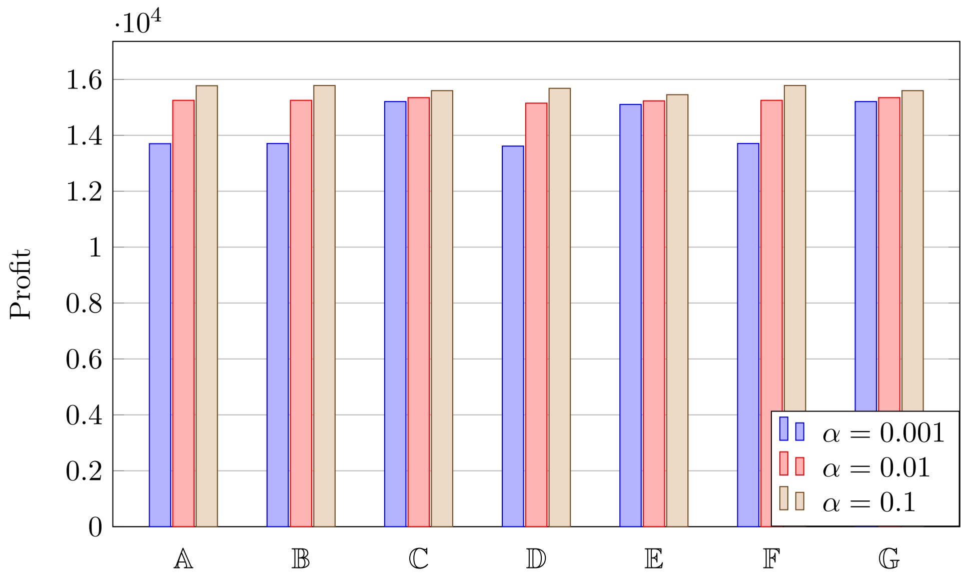


FIGURE 4.2. Comparison for different values of $\alpha$ with fixed $\delta = 25$.

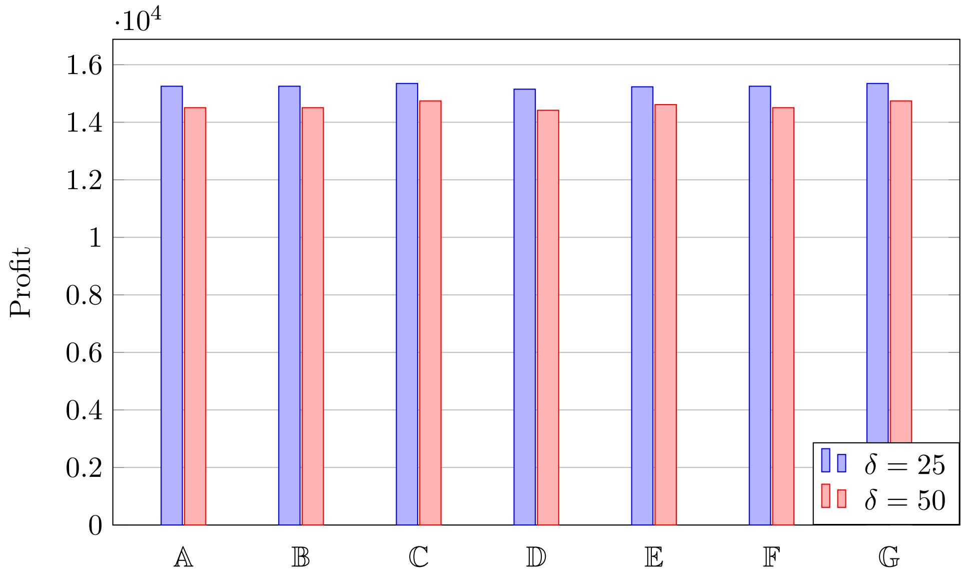


FIGURE 4.3. Comparison for different values of $\delta$ with fixed $\alpha = 0.01$.

as well as ILP. In observing the values listed in the *Heuristic approach* and *GSEMO* columns in Table 4.1, we find that, in most of the instances, the results obtained using the heuristic approach are better than those obtained using GSEMO. Furthermore, the computation time associated with the heuristic approach is shorter than that associated with GSEMO. However, when applying the Chernoff bound to the problems, the results obtained using GSEMO are better than those obtained using the heuristic approach in most instances. Moreover, it can be observed from the tables that the GSEMO outperforms (1+1) EA in combination with different estimate methods.

Furthermore, it can be seen from the values in the *profit* columns that the profit decreases as the value of $\alpha$ decreases with a fixed value of $\delta$, and that the profit decreases when the value of $\delta$ increases, while the value of $\alpha$ stays constant. We take the first instances from Table 4.1 and Table 4.2 as examples, presenting the effects of $\delta$ and $\alpha$ in Figures 4.2 and 4.3. Figure 4.2 shows how the chance-constrained bound (denoted as $\alpha$) affects the quality of the solution when the uncertainty of weights is fixed. In both figures, the $x$ labels are the algorithms combined with probability tools, where $\mathbb{A}$ denotes ILP(2mins)-Chebyshev's inequality, $\mathbb{B}$ denotes heuristic approach-Chebyshev's inequality, $\mathbb{C}$ denotes heuristic approach-Chernoff bound, $\mathbb{D}$ denotes (1+1) EA-Chebyshev's inequality, $\mathbb{E}$ denotes (1+1) EA-Chernoff bound, $\mathbb{F}$ denotes GSEMO-Chebyshev's inequality and $\mathbb{G}$ denotes GSEMO-Chernoff bound.

Figure 4.2 shows how the chance-constrained bound denoted as $\alpha$ affects the solution quality when the uncertainty of weights is fixed. Correspondingly, Figure 4.3 shows how the uncertainty of weights affects the solution quality when $\alpha$ is fixed. Both bar charts show the average values of (1+1) EA and GSEMO with respect to probability inequality. In Figure 4.2, the three bars of each group correspond to the value of $\alpha$ equal to $\{0.001, 0.01, 0.1\}$. In Figure 4.3, the two bars of each group correspond to the value of $\delta = \{25, 50\}$. We can see from the figures that profits increase as the chance-constrained bound $\alpha$ increases or the uncertainty of weights $\delta$ decreases. Intuitively, this makes sense, as the relaxing of $\alpha$ or the tightening of $\delta$ allows the algorithms to compute solutions that are closer to the capacity and thereby increase their profit.

Next, we investigate the difference between the estimated methods. It can be observed from Table 4.3, for all approaches, that when the value of $\alpha$ is small (i.e.,$\alpha = 0.001$ or 0.01), the profits obtained using the Chernoff bound are significantly better than those obtained using Chebyshev's inequality. In contrast, if $\alpha = 0.1$, the results obtained using Chebyshev's inequality are significantly better than those obtained using the Chernoff bound. The experimental results match the theoretical implications of Theorem 4.2.4 and our related discussion.

In Table 4.3, the *profit* under (1+1) EA and GSEMO denotes the average profit of the 30 runs, and *stat* denotes the statistical comparison of the algorithms combined with the estimation methods. The numbers in the *stat* column denote the significance of the results for an algorithm and constraint violation estimation method. For example, in Table 4.3, the numbers $(2(+), 3(-), 4(+))$ listed in the *stat* column in the first row under *(1+1) EA - Chernoff bound (1)* means that the current algorithm is significantly better than *Chebyshev's inequality (2)* and *Chebyshev's inequality (4)*, and significantly worse than *Chernoff bound (3)*.

We now compare the performances of (1+1) EA with GSEMO. Looking at the values listed in the *stat* columns of Table 4.3, it can be seen that the results listed under *(1+1) EA-Chernoff (1)* are significantly worse than those under *GSEMO-Chernoff (3)*. A similar relationship exists between the values under *(1+1) EA-Chebyshev (2)* and *GSEMO-Chebysehv(4)*, indicating that the performance of GSEMO is significantly better than that of (1+1) EA under all probability tails.

In the second category of instances, the weights of items have multiplicative uniform distributions and take values in the real interval $[(1-\beta)a_i, (1+\beta)a_i]$. Here, the uncertainty gap corresponding to each item is expressed as a percentage of the expected weight. In this setting, the distance of the random intervals between each item vary for each item pair. We apply Chebyshev's inequality to deal with the chance constraint in this category of instances. The results are listed in Table 4.4 and Table 4.5, and we mark the best result across all approaches in each row in bold. The grey cubes in the table highlight the instances in which ILP can be completed in the computation time listed in the second columns under the title *ILP*. It can be observed that all algorithms produce inferior solutions when the uncertainty added to the weights of items (measured by $\beta$) increases, or the upper bound of the probability of overloading the capacity in the form of $\alpha$ decreases.

Furthermore, it can be observed from Table 4.4, we found that the results obtained using the heuristic approach are better than those obtained using EAs, and that results obtained through the heuristic approach are more significant in the case of *uncorrelated* instances. However, the results for instances which have 500 items cannot be list in Table 4.5, since it takes more than two hours for the heuristic approach to obtain a result in such cases. Moreover, the results listed in the *ILP* columns are better than those in the *(1+1) EA* columns, but it is difficult to make a clear comparison of these results with the results of *GSEMO*. The computation time associated with ILP is longer than that associated with EAs for most of the instances. Besides, as shown in Table 4.5 the performance of GSEMO is significantly better than that of (1+1) EA for all instances.

In the last category of instance, we consider instances for which the weights of items are chosen according to the Normal distributions with known expectations and variances of weights. In all instances, the variances of weights are expressed as a percentage of the expected weights: $\sigma_i^2 = a_i \cdot \beta$. In this arrangement, according to the theorem of the Chernoff bound, the upper bound of the chance constraint cannot be calculated using the Chernoff bound, and we only apply Chebyshev's inequality. The results are listed in Table 4.6 and Table 4.7, and we mark the best results across all approaches in each row in bold.

The *profit* column lists the mean value for 30 runs, and the *stat* column provides statistical test results based on the comparison of the performances of (1+1) EA and GSEMO. It can be observed from the tables that algorithms produce inferior solutions when the value of $\beta$ increases or the value of $\alpha$ increases. The results show that the heuristic approach outperforms the other algorithms for instances with 100 items, but takes more than two hours for instances with 500 items. Moreover, under the condition in which algorithms are run at similar computation times, the results solved by GSEMO are better than those solved using ILP. Furthermore, the performance of GSEMO is significantly better than that of (1+1) EA.

By investigating the results list in all tables, we found that when limited the computation time into seconds, the GSEMO can provide high-quality solutions overall. Moreover, in an acceptable running time (minutes), the heuristic approach performs better than other approaches in small instances, though it cannot obtain a solution within two hours for large size instances. However, as mentioned in Section 4.4 that (1+1) EA and GSEMO are the simple evolutionary algorithms, and the performance of them can be improved by applying operators that is the problem-specific operators that we developed. In the following subsections, we investigate the performance of different evolutionary algorithms in terms of solving the chance-constrained knapsack problem with different operators.

### 4.5.3 Experimental Results of Improvements to Single-objective EAs Using Specific Operators

Now, we examine the impact of the heavy-tail mutation operator and the problem-specific crossover operator in (1+1) EA (see Algorithm 6) and ($\mu$+1) EA (see Algorithm 12). The benchmarks used here are the same as those employed in Section 4.5.1. We report the mean profit and the standard deviation of 30 independent runs for all algorithms. Each run used $5 \times 10^6$ fitness evaluations. A Kruskal-Wallis test (Corder and Foreman, 2014) with a 95% confidence interval integrated with the posterior Bonferroni test is used to compare multiple solutions.

In the following subsection, we consider all combinations of algorithms and operators. We set the $\beta$ in power-law distribution equal to 1.5, which is the recommended value of $\beta$ given by Doerr et al. (2017). For each instance, we investigate different settings as well as the difference between the uncertainty of weights and the chance-constrained probability. We report the results obtained using all algorithms with Chebyshev's inequality and the Chernoff bound separately.

#### Results for (1+1) EA

To show the differences between the evolutionary algorithms using the standard mutation operator and the heavy-tail mutation operator, we investigate the performance of $(1+1)$ EA using the heavy-tail mutation operator in order to solve the CCKP instances. Table 4.8 and Table 4.9 list the means and standard deviations of profit for 30 independent runs with regard to the probability estimate methods. For clarity, we use *Standard* $(1+1)$ *EA* to refer to the $(1+1)$ EA using standard mutation operator, and show the $(1+1)$ EA using heavy-tail mutation operator as $(1+1)$ *EA with HT* in the tables.

In Table 4.8 and 4.9, the *stat* column shows the rank of each algorithm for all instances. If it is possible to conduct a statistically significant comparison of two algorithms, X(+) denotes that the current algorithm outperforms algorithm X. Furthermore, X(-) signifies that the current algorithm is significantly worse than the algorithm X. For example, the numbers 2(-) listed in the first row under the *Standard* $(1+1)$ *EA(1)* mean that the current solution is significantly worse than the solutions obtained using $(1+1)$ *EA with HT mutation (2)*.

The results in Tables 4.8 and 4.9 indicate that there is a significant difference between the two mutation operators when they are used in the single-objective evolutionary algorithm. The $(1+1)$ EA with the heavy-tail mutation operator outperforms the standard $(1+1)$ EA in all cases. Moreover, in most of uncorrelated instances, the $(1+1)$ EA with heavy-tail mutations obtains solutions with a lower standard deviation than those obtained with the other algorithm. In summary, the results show that the heavy-tail mutation operator leads to better performance when solving CCKP instances, which we have emphasised in bold in the *stat* columns.

### Results for $(\mu+1)$ EA

The purpose of this subsection is to investigate the effectiveness of the problem-specific crossover operator associated with single-objective evolutionary algorithms. Here, we run the $(\mu+1)$ EA with a population size of 10. To simplify the names of algorithms, we use the following notations: *Standard* $(\mu+1)$ *EA* is the $(\mu+1)$ EA using standard mutation operator, $(\mu+1)$ *EA with HT* is the $(\mu+1)$ EA using the heavy-tail mutation operator and $(\mu+1)$ *EA with HT and PS* is the $(\mu+1)$ EA using the heavy-tail mutation operator and the problem-specific crossover operator.

Tables 4.10 and 4.11 list the results of using the Chernoff bound and Chebyshev's inequality to estimate the constraint violation probability of a CCKP solution separately. As can be seen from the tables, the performance these algorithms when using the heavy-tail mutation operator is significantly better than the performance when using the standard mutation for all instances. Therefore the conclusion is the same as in the case of the $(1+1)$ EA. Another insight which can be observed from these tables related to the values of the columns $(\mu+1)$ *EA with HT (4)* and $(\mu+1)$ *EA with HT and PS (5)*. We can see that the results obtained by the $(\mu+1)$ *EA with HT and PS (5)* are significantly better than those obtained by the $(\mu+1)$ *EA with HT (4)*. This demonstrates the effectiveness of the PS crossover operator in terms of solving the CCKP instances in single-objective evolutionary algorithms, when compared to mutation only.

Moreover, by comparing the values of the corresponding columns in Tables 4.8 and 4.10, and Tables 4.9 and 4.11 respectively according to the probability tails, the results for the $(\mu+1)$ *EA with HT and PS (5)* are better than those for the other combinations of algorithms and operators, for both estimated methods. In all tables we emphasise in bold the statistical results. In summary, the results in this section indicate that the heavy-tail mutation operator and the problem-specific crossover operator are active in single-objective evolutionary algorithms when dealing with instances of the CCKP. The next section, therefore, moves on to discuss the performance of these operators in multi-objective approaches.

### 4.5.4 Experimental Results of Improvements to Multi-objective EAs using Specific Operators

Next, we examine the impact of problem-specific operators and the improved multi-objective model in GSEMO (see Algorithm 7) and NSGA-II (see Algorithm 8). In the following part of this subsection, we first investigate the GSEMO using improved model. We then run NSGA-II with a population size of 20 using different surrogate functions, comparing the performance of GSEMO and NSGA-II at the end of this section.

#### Experimental Results for GSEMO

To compare the performance of the old model with the new model, we first apply the GSEMO solving the same instances as those proposed in Section 4.5.1, but with different fitness functions. Next, to test the effectiveness of the PS crossover operator in the multi-objective algorithm, we combined the PS crossover operators in GSEMO to solve the new model. Tables 4.12 and 4.13 show the results obtained using different probability tails separately. To simplify the algorithm names in the tables, we use *old* to denote the old multi-objective model and *new* to denote the new multi-objective model. *Uniform*, *HT* and *PS* denote the uniform crossover operator, heavy-tail mutation operator and problem-specific crossover operator separately.

Table 4.12 shows that, for example, for the first instance in the *bounded strongly correlated* type, GSEMO using the new model outperforms GSEMO using the old model. For the other instances, both algorithms perform as well as each other. It can be observed more clearly in Table 4.13 that, the new model performs significantly better than the old model when dealing with instances which are *bounded strongly correlated* type. Furthermore, as can be seen from the column *GSEMO with the new model and PS*, this algorithm displayed significantly better results than the other two algorithms. These results indicate that the new multi-objective model is effective in solving instances of the CCKP and performs better than the old model in most cases. The PS crossover operator can improve the performance of the multi-objective evolutionary algorithm when dealing with the CCKP.

#### Experimental Results for NSGA-II

In this section, we investigate the performance of NSGA-II with the combination of two crossover operators, the uniform crossover and the PS crossover, and the two multi-objective models. We modify NSGA-II to keep the best feasible solution for CCKP in each iteration. Since NSGA-II generates 10 offsprings in each iteration with a population size of 20 while GSEMO only generates 1 offspring, we set the iteration of NSGA-II to $5 \times 10^5$ (instead of $5 \times 10^6$ for GSEMO), which results in the same number of fitness evaluations for both algorithms.

Table 4.14 and 4.15 show the results obtained when using the Chernoff bound and Chebyshev's inequality, respectively. A significant improvement can be observed in the results obtained for the PS crossover when compared to those obtained for the uniform crossover.

We now compare the performance of the old model with the new model of the same algorithm. Considering the values in the *stat* columns of Table 4.14, it can be seen that with the uniform crossover operator, the solutions for the new model (*NSGA-II with new and uniform (11)*) are mostly better than those for the old model (*NSGA-II* *with old and uniform (9)*). However, in some instances where $\delta = 50$, the old model outperforms the new model. The solutions presented in *NSGA-II with old and PS (10)* and *NSGA-II with new and PS (12)* indicate that, in most instances, the new model performs better than the old model when using the PS crossover operator.

However, the opposite conclusion can be drawn from Table 4.15. The solutions listed in the *NSGA-II with new and uniform (11)* and *NSGA-II with old and uniform (9)* show that NSGA-II with uniform crossover performs better when dealing with the old model than the new model. However, the correlation between the old model and the new model is interesting when using NSGA-II with a PS crossover operator to solve the problem, the correlation between the old model and the new model is interesting. The relationship between the results obtained when solving the two models is related to the type of instances under consideration. In other words, for the *bounded strongly correlated* instances, the old model outperforms the new model, while for the *uncorrelated* instances, the new model is better than the old model in most cases.

The next insight can be drawn from Table 4.15 is that, observing the values in the *GSEMO with new and PS (8)* and *NSGA-II with new and PS (12)* columns, the results obtained uisng GSEMO are significantly better than those obtained using NGSA-II for all instances. This comparison points to a direction for possible further research into the use of state-of-art multi-objective evolutionary algorithms such as NSGA-II and SPEA2 to solve CCKP. Moreover, we compare the performance of the best single-objective algorithm ($\mu$+1) *EA with HT and PS* and the best multi-objective algorithm *GSEMO with new and PS* according to the estimated methods. It is observed that the performance of the multi-objective algorithm is significantly better than that of the single-objective algorithm for all instances.

## 4.6 Conclusion

Chance-constrained optimisation problems play a crucial role in various real-world applications, as they limit the probability of violating a given constraint when dealing with stochastic problems. In this chapter, we considered the chance-constrained knapsack problem and showed how to incorporate well-known probability tail inequalities into the search process of an evolutionary algorithm. Furthermore, we introduced deterministic approaches and compared them to the designed evolutionary algorithms. The disadvantage of deterministic approaches in dealing with the chance-constrained knapsack problem is their computation time. Our experimental results show that DP cannot solve large instances in 10 hours and NU cannot solve them in two hours; however, EAs can obtain a good quality solution in minutes. Even for small instances, GSEMO can arrive at solutions of similar quality to those arrived at by ILP and NU, while taking less computation time in most instances. Our investigations demonstrate the circumstances under which Chebyshev's inequality or the Chernoff bound are favoured as part of the fitness evaluation. The Chernoff bound is preferable in cases where the probability of violating the capacity of the knapsack is small. We have also shown that using a multi-objective approach when dealing with the chance-constrained knapsack problem provides a clear benefit in comparison to the use of its single-objective formulation for all kinds of instance classes.

Moreover, to improve the performance of evolutionary algorithms, this chapter proposed a problem-specific crossover operator and the heavy-tail mutation operator, which can be applied in EAs to deal with the CCKP. Our experimental results show that the proposed operators improve the performance of single-objective EAs when solving CCKP instances. We also introduced a new multi-objective model for the CCKP. The experimental results show that combining this new model with the problem-specific crossover operator in GSEMO and NSGA-II leads to significant performance improvements in terms of solving the CCKP.

TABLE 4.3. Statistical results for instances with 500 items based on additive uniform distribution

| | Capacity | $\delta$ | $\alpha$ | ILP(Chebyshev) | | | (1+1)EA | | | | | | GSEMO | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Chernoff(1) | | | Chebyshev(2) | | | Chernoff(3) | | | Chebyshev(4) | | |
| | | | | 2mins | 6mins | 10mins | profit | time(ms) | stat | profit | time(ms) | stat | profit | time(ms) | stat | profit | time(ms) | stat |
| bou-s-c 1 | 61447 | 25 | 0.001 | 74403 | 74385 | 74385 | 76464.3 | 5887.5 | 2(+),3(-),4(+) | 73213.8 | 5866.67 | 1(-),3(-),4(-) | **77750.4** | 25626.04 | 1(+),2(+),4(+) | 74207.1 | 27694.79 | 1(-),2(+),3(-) |
| | | | 0.01 | 77951 | 77951 | 77951 | 76890.33 | 5887.50 | 2(+),3(-),4(-) | 76569.53 | 5879.17 | 1(-),3(-),4(-) | **78064.47** | 25572.4 | 1(+),2(+),4(+) | 77552.07 | 28609.9 | 1(+),2(+),3(-) |
| | | | 0.1 | 78649 | **78912** | **78912** | 77194.97 | 5889.06 | 2(-),3(-),4(-) | 77642.27 | 5883.33 | 1(+),3(-),4(-) | 78429.77 | 25743.23 | 1(+),2(+),4(-) | 78672.67 | 29402.6 | 1(+),2(+),3(+) |
| | | 50 | 0.001 | 66465 | 69443 | 69443 | 74964.07 | 5894.79 | 2(+),3(-),4(+) | 68709.77 | 5851.04 | 1(-),3(-),4(-) | **76008.8** | 28145.31 | 1(+),2(+),4(+) | 69627.7 | 25955.73 | 1(-),2(+),3(-) |
| | | | 0.01 | 76121 | 76222 | 76222 | 75627.37 | 5903.13 | 2(+),3(-),4(-) | 75038.43 | 5869.27 | 1(-),3(-),4(-) | **76629.73** | 28291.67 | 1(+),2(+),4(+) | 75965.73 | 27930.73 | 1(+),2(+),3(-) |
| | | | 0.1 | 46027 | **78227** | **78227** | 76319.97 | 5892.71 | 2(-),3(-),4(-) | 77231.63 | 5878.65 | 1(+),3(-),4(-) | 77385.53 | 28717.19 | 1(+),2(+),4(-) | 78187.27 | 28957.81 | 1(+),2(+),3(+) |
| bou-s-c 2 | 162943 | 25 | 0.001 | 177096 | 186590 | 186590 | 188967.4 | 6010.94 | 2(+),3(-),4(+) | 184846.9 | 5990.10 | 1(-),3(-),4(-) | **190575.8** | 31282.81 | 1(+),2(+),4(+) | 186081.5 | 34696.35 | 1(-),2(+),3(-) |
| | | | 0.01 | 190681 | 190681 | 190757 | 189348.5 | 6015.10 | 2(+),3(-),4(-) | 189003.4 | 6017.71 | 1(-),3(-),4(-) | **190940.3** | 31718.23 | 1(+),2(+),4(+) | 190297.6 | 36672.92 | 1(+),2(+),3(-) |
| | | | 0.1 | **192116** | **192116** | **192116** | 189852.2 | 6002.08 | 2(-),3(-),4(-) | 190439.4 | 6009.38 | 1(+),3(-),4(-) | 191380.6 | 31677.6 | 1(+),2(+),4(-) | 191632 | 37396.88 | 1(+),2(+),3(+) |
| | | 50 | 0.001 | 180281 | 180348 | 180348 | 187149.1 | 6005.21 | 2(+),3(-),4(+) | 178807.7 | 5981.77 | 1(-),3(-),4(-) | **188326.5** | 36233.33 | 1(+),2(+),4(+) | 180210 | 32353.13 | 1(-),2(+),3(-) |
| | | | 0.01 | 187042 | 187079 | 187079 | 187931.4 | 6008.85 | 2(+),3(-),4(-) | 187005.2 | 5997.40 | 1(-),3(-),4(-) | **189064.4** | 36852.6 | 1(+),2(+),4(+) | 188299.7 | 35451.56 | 1(+),2(+),3(-) |
| | | | 0.1 | 189477 | 190630 | 190630 | 188798.8 | 6011.46 | 2(-),3(-),4(-) | 189713 | 6005.21 | 1(+),3(-),4(-) | 189986 | 37560.42 | 1(+),2(+),4(-) | **190993.3** | 37417.19 | 1(+),2(+),3(+) |
| bou-s-c 3 | 307286 | 25 | 0.001 | 341305 | 341305 | 341453 | 344059.7 | 6132.81 | 2(+),3(-),4(+) | 339020.5 | 6126.56 | 1(-),3(-),4(-) | **346115.3** | 38438.54 | 1(+),2(+),4(+) | 340895.1 | 40478.65 | 1(-),2(+),3(-) |
| | | | 0.01 | 346271 | 346271 | 346271 | 344460 | 6126.56 | 2(+),3(-),4(-) | 343891 | 6135.94 | 1(-),3(-),4(-) | **346523.6** | 39475 | 1(+),2(+),4(+) | 345815.1 | 43563.54 | 1(+),2(+),3(-) |
| | | | 0.1 | 347725 | 347729 | **347905** | 344976.1 | 6166.56 | 2(-),3(-),4(-) | 345622.4 | 6134.38 | 1(+),3(-),4(-) | 347048.1 | 39237.5 | 1(+),2(+),4(-) | 347418.7 | 44185.42 | 1(+),2(+),3(+) |
| | | 50 | 0.001 | 169122 | 169122 | 210932 | 341860.8 | 6127.08 | 2(+),3(-),4(+) | 332022.1 | 6108.85 | 1(-),3(-),4(-) | **343418.6** | 47959.9 | 1(+),2(+),4(+) | 333882.8 | 38178.13 | 1(-),2(+),3(-) |
| | | | 0.01 | 341530 | 343866 | 344052 | 342619.9 | 6132.29 | 2(+),3(-),4(-) | 341661.9 | 6108.85 | 1(-),3(-),4(-) | **344242** | 48193.75 | 1(+),2(+),4(+) | 343537.7 | 41935.42 | 1(+),2(+),3(-) |
| | | | 0.1 | 346321 | 347064 | **347064** | 343776 | 6128.65 | 2(-),3(-),4(-) | 344848.8 | 6131.77 | 1(+),3(-),4(-) | 345325.3 | 49109.38 | 1(+),2(+),4(-) | 346723.6 | 44240.1 | 1(+),2(+),3(+) |
| uncorr 1 | 37686 | 25 | 0.001 | 16998 | 58826 | 66238 | 85264.77 | 5956.25 | 2(+),3(-),4(+) | 80509.37 | 5919.79 | 1(-),3(-),4(-) | **86117.7** | 20050.52 | 1(+),2(+),4(+) | 81319.77 | 20571.88 | 1(-),2(+),3(-) |
| | | | 0.01 | 86128 | 86128 | 86128 | 85685.3 | 5955.73 | 2(+),3(-),4(-) | 85187.5 | 5930.73 | 1(-),3(-),4(-) | **86540.37** | 20035.94 | 1(+),2(+),4(+) | 86002.63 | 21854.17 | 1(+),2(+),3(-) |
| | | | 0.1 | 87252 | 87675 | **87675** | 86218.6 | 5950.00 | 2(-),3(-),4(-) | 86899.67 | 5939.06 | 1(+),3(-),4(-) | 87066.83 | 20544.27 | 1(+),2(+),4(-) | 87518.63 | 22421.88 | 1(+),2(+),3(+) |
| | | 50 | 0.001 | 15712 | 59369 | 70672 | 83116.43 | 5955.73 | 2(+),3(-),4(+) | 73929.1 | 5895.31 | 1(-),3(-),4(-) | **83829.9** | 23189.06 | 1(+),2(+),4(+) | 74731.03 | 20529.69 | 1(-),2(+),3(-) |
| | | | 0.01 | 58826 | 83985 | 83985 | 84020.67 | 5955.21 | 2(+),3(-),4(+) | 83065.43 | 5917.19 | 1(-),3(-),4(-) | **84668.5** | 22652.6 | 1(+),2(+),4(+) | 83820.63 | 21008.33 | 1(-),2(+),3(-) |
| | | | 0.1 | **87068** | **87068** | **87068** | 85065.93 | 5953.13 | 2(-),3(-),4(-) | 86128.03 | 5931.25 | 1(+),3(+),4(-) | 85719.73 | 23177.6 | 1(+),2(-),4(-) | 86849.87 | 21986.46 | 1(+),2(+),3(+) |
| uncorr 2 | 93559 | 25 | 0.001 | 143444 | 143448 | 143455 | 147143.5 | 6072.40 | 2(+),3(-),4(+) | 142829.3 | 6061.98 | 1(-),3(-),4(-) | **147754.5** | 24839.58 | 1(+),2(+),4(+) | 143393.7 | 24064.58 | 1(-),2(+),3(-) |
| | | | 0.01 | 101606 | 147780 | 147780 | 147539.5 | 6072.40 | 2(+),3(-),4(-) | 147115.2 | 6061.98 | 1(-),3(-),4(-) | **148125.7** | 25049.48 | 1(+),2(+),4(+) | 147605.7 | 24511.98 | 1(+),2(+),3(-) |
| | | | 0.1 | **149217** | **149217** | **149217** | 147936.9 | 6068.75 | 2(-),3(-),4(-) | 148518 | 6079.69 | 1(+),3(-),4(-) | 148594.9 | 24431.25 | 1(+),2(+),4(-) | 148960.7 | 25139.06 | 1(+),2(+),3(+) |
| | | 50 | 0.001 | 100660 | 120860 | 137219 | 145291.4 | 6063.54 | 2(+),3(-),4(+) | 136767 | 6048.44 | 1(-),3(-),4(-) | **145594.4** | 26981.25 | 1(+),2(+),4(+) | 137400 | 22478.13 | 1(-),2(+),3(-) |
| | | | 0.01 | 101554 | 145726 | 145726 | 146067.2 | 6066.67 | 2(+),3(-),4(+) | 145189.8 | 6048.44 | 1(-),3(-),4(-) | **146326** | 27126.04 | 1(+),2(+),4(+) | 145655.9 | 24484.9 | 1(-),2(+),3(-) |
| | | | 0.1 | 101648 | **148531** | **148531** | 146945.4 | 6070.83 | 2(-),3(-),4(-) | 147955.5 | 6075.52 | 1(+),3(+),4(-) | 147254 | 26751.56 | 1(+),2(-),4(-) | 148368.5 | 24806.77 | 1(+),2(+),3(+) |
| uncorr 3 | 171819 | 25 | 0.001 | 138416 | 200244 | 204498 | 207630.3 | 6189.06 | 2(+),3(-),4(+) | 204214.7 | 6166.67 | 1(-),3(-),4(-) | **208249.2** | 22107.29 | 1(+),2(+),4(+) | 204790.2 | 20989.58 | 1(-),2(+),3(-) |
| | | | 0.01 | 208260 | 208260 | 208260 | 207978.8 | 6205.73 | 2(+),3(-),4(-) | 207537.9 | 6172.92 | 1(-),3(-),4(-) | **208532** | 22053.13 | 1(+),2(+),4(+) | 208161.3 | 21864.06 | 1(+),2(+),3(-) |
| | | | 0.1 | **209344** | **209344** | **209344** | 208266.9 | 6196.35 | 2(-),3(-),4(-) | 208671.4 | 6169.79 | 1(+),3(-),4(-) | 208889.4 | 22261.46 | 1(+),2(+),4(-) | 209180.5 | 21320.31 | 1(+),2(+),3(+) |
| | | 50 | 0.001 | 114287 | 144843 | 144843 | 206180.1 | 6193.75 | 2(+),3(-),4(+) | 199212.3 | 6169.79 | 1(-),3(-),4(-) | **206374.7** | 25970.83 | 1(+),2(+),4(+) | 199810.8 | 19647.4 | 1(-),2(+),3(-) |
| | | | 0.01 | 154981 | 156922 | 172798 | 206719.9 | 6193.23 | 2(+),3(-),4(+) | 206014 | 6175.00 | 1(-),3(-),4(-) | **206915.9** | 26699.48 | 1(+),2(+),4(+) | 206622.3 | 22066.67 | 1(-),2(+),3(-) |
| | | | 0.1 | 208391 | **208899** | **208899** | 207471.5 | 6190.10 | 2(-),3(-),4(-) | 208135.4 | 6184.38 | 1(+),3(+),4(-) | 207649 | 26861.46 | 1(+),2(-),4(-) | 208754 | 21738.54 | 1(+),2(+),3(+) |

Table 4.4. Statistical results for the instance eil101 with 100 items based on multiplicative uniform distribution and Chebyshev's inequality

| | Capacity | $\beta$ | $\alpha$ | ILP | | | Heuristic approach | | DP | | (1+1)EA | | GSEMO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 2mins | 6mins | 10mins | profit | time(ms) | profit | time(ms) | profit | time(ms) | profits | time(ms) |
| bou-s-c 1 | 11775 | 0.01 | 0.001 | 15309 | 15309 | 15318 | **15350** | 690 | **15350** | 4810778 | 15230.67 | 1221.88 | 15349.6 | 62382.7675 |
| | | | 0.01 | **15801** | **15801** | **15801** | **15801** | 1072 | **15801** | 12734208 | 15684.93 | 1234.9 | **15801** | 66815.46 |
| | | | 0.1 | **15946** | **15946** | **15946** | **15946** | 1122 | **15946** | 26128269 | 15825.47 | 1218.75 | 15940.4 | 71332.5442 |
| | | 0.05 | 0.001 | 13100 | 13199 | 13199 | **13245** | 1369 | **13245** | 6293917 | 13123.27 | 1222.92 | **13245** | 55164.344 |
| | | | 0.01 | 14905 | 14905 | 14905 | **15027** | 1073 | **15027** | 16956887 | 14926.2 | 1226.56 | 14992.4333 | 56806.4804 |
| | | | 0.1 | 15702 | 15702 | 15702 | **15710** | 1181 | **15710** | 31257870 | 15581.1 | 1221.88 | **15710** | 59787.857 |
| | | 0.1 | 0.001 | 11128 | 11128 | 11128 | 11247 | 646 | **11270** | 6724128 | 11193.8 | 1222.4 | 11269.0667 | 53176.8783 |
| | | | 0.01 | 13983 | 14002 | 14043 | **14143** | 1174 | **14143** | 19649101 | 14037.23 | 1222.4 | 14141.4667 | 59414.7318 |
| | | | 0.1 | 15312 | 15346 | 15346 | **15384** | 1267 | **15384** | 35553950 | 15257.7 | 1220.31 | **15384** | 65737.6751 |
| bou-s-c 2 | 31027 | 0.01 | 0.001 | 36524 | 36634 | 36634 | **36728** | 5427 | – | – | 36554.43 | 1264.58 | 36694.4667 | 116506.762 |
| | | | 0.01 | 37475 | 37515 | 37515 | **37557** | 8078 | – | – | 37375.73 | 1261.46 | 37488.7 | 119125.093 |
| | | | 0.1 | 37796 | 37796 | 37796 | **37819** | 14631 | – | – | 37679.77 | 1257.29 | 37816.6667 | 154614.705 |
| | | 0.05 | 0.001 | 32069 | 32170 | 32239 | 32685 | 4349 | – | – | 32607.13 | 1277.08 | **32701.1** | 210753.913 |
| | | | 0.01 | 35855 | 36002 | 36002 | **36069** | 5834 | – | – | 35903.27 | 1267.19 | 36044.3 | 200180.797 |
| | | | 0.1 | 37284 | 37284 | 37284 | **37367** | 6016 | – | – | 37170.57 | 1259.38 | 37301.1667 | 172459.537 |
| | | 0.1 | 0.001 | 28246 | 28354 | 28354 | 28787 | 2575 | – | – | 28724.27 | 1286.46 | **28821** | 91837.3079 |
| | | | 0.01 | 34135 | 34135 | 34135 | **34419** | 5018 | – | – | 34278.83 | 1280.21 | 34418.5333 | 99988.7866 |
| | | | 0.1 | 36576 | 36701 | 36701 | **36784** | 5886 | – | – | 36619.67 | 1263.02 | 36758.13 | 78642.88 |
| bou-s-c 3 | 58455 | 0.01 | 0.001 | 65667 | 65667 | 65667 | **65737** | 63940 | – | – | 65510.7 | 1316.15 | 65690.53 | 97893.23 |
| | | | 0.01 | 66958 | 67011 | 67011 | **67077** | 80608 | – | – | 66831.33 | 1311.46 | 66995.6 | 101069.27 |
| | | | 0.1 | 67461 | 67461 | 67461 | **67480** | 151215 | – | – | 67290.13 | 1313.02 | 67439.1 | 100545.31 |
| | | 0.05 | 0.001 | 58770 | 58770 | 58770 | **59319** | 18667 | – | – | 59223.87 | 1344.79 | 59304.3 | 84612.5 |
| | | | 0.01 | 64472 | 64500 | 64500 | **64713** | 65809 | – | – | 64492.3 | 1320.83 | 64652.57 | 96270.83 |
| | | | 0.1 | 66671 | 66671 | 66671 | **66742** | 69379 | – | – | 66507.07 | 1310.94 | 66691.73 | 100061.46 |
| | | 0.1 | 0.001 | 48564 | 51059 | 50689 | 53359 | 8798 | – | – | 53265.03 | 1358.33 | **53359.13** | 74005.21 |
| | | | 0.01 | 61574 | 61657 | 61657 | 61974 | 58018 | – | – | 61924.97 | 1325 | **62063.7** | 90768.75 |
| | | | 0.1 | 65674 | 65738 | 65738 | **65837** | 47670 | – | – | 65633.57 | 1319.27 | 65774.27 | 97894.79 |
| uncorr 1 | 7715 | 0.01 | 0.001 | **17064** | 4s | | **17064** | 204 | **17064** | 31106720 | 16989.4 | 1218.23 | **17064** | 8394.56443 |
| | | | 0.01 | **17366** | 3.08s | | **17366** | 191 | **17366** | 49253218 | 17346.17 | 1229.69 | **17366** | 8361.6105 |
| | | | 0.1 | **17499** | 1.39s | | **17499** | 199 | **17499** | 56877261 | 17444 | 1263.54 | **17499** | 6842.4115 |
| | | 0.05 | 0.001 | 15322 | 15322 | 15322 | **15385** | 151 | **15385** | 13416371 | 15293.33 | 1210.94 | **15385** | 7834.6693 |
| | | | 0.01 | **16875** | **16875** | **16875** | **16875** | 185 | **16875** | 19806209 | 16808.23 | 1219.27 | **16875** | 7505.27357 |
| | | | 0.1 | **17325** | 4.64s | | **17325** | 176 | **17325** | 39985165 | 17290.97 | 1218.23 | **17325** | 6745.7595 |
| | | 0.1 | 0.001 | **13589** | **13589** | **13589** | **13589** | 70 | **13589** | 463375 | 13507.6 | 1209.38 | **13589** | 4679.62223 |
| | | | 0.01 | **16143** | 60s | | **16143** | 117 | **16143** | 9685826 | 16047.6 | 1216.15 | **16143** | 6412.02927 |
| | | | 0.1 | **17064** | 60s | | **17064** | 158 | **17064** | 28925543 | 16994.47 | 1218.75 | **17064** | 9153.54877 |
| uncorr 2 | 19545 | 0.01 | 0.001 | 29075 | 29085 | 29085 | **29106** | 940 | – | – | **29106** | 1265.63 | **29106** | 15434.3676 |
| | | | 0.01 | **29466** | 16.52s | | **29466** | 1195 | – | – | 29390.6 | 1165.63 | **29466** | 15147.9511 |
| | | | 0.1 | **29585** | 3.01s | | **29585** | 1001 | – | – | 29521.43 | 1265.1 | **29585** | 21475.098 |
| | | 0.05 | 0.001 | 27087 | 27087 | 27087 | **27096** | 814 | – | – | 26974.73 | 1256.25 | **27096** | 18433.7486 |
| | | | 0.01 | 28740 | 28815 | 28742 | **28815** | 728 | – | – | 28687.23 | 1267.71 | **28815** | 15621.9082 |
| | | | 0.1 | **29415** | 118.91s | | **29415** | 656 | – | – | 29327.2 | 1266.67 | **29415** | 22129.7554 |
| | | 0.1 | 0.001 | 25006 | 25033 | 25006 | **25043** | 540 | – | – | 24941.5 | 1248.44 | **25043** | 15159.6398 |
| | | | 0.01 | **27985** | **27985** | **27985** | **27985** | 958 | – | – | 27899.77 | 1261.98 | **27985** | 14720.0369 |
| | | | 0.1 | **29155** | **29155** | **29155** | **29155** | 740 | – | – | 29072.53 | 1265.63 | **29155** | 15869.881 |
| uncorr 3 | 36091 | 0.01 | 0.001 | 40651 | **40672** | **40672** | **40672** | 1351 | – | – | 40633.83 | 1342.19 | 40671.7 | 55625.4561 |
| | | | 0.01 | **40994** | 57.04s | | **40994** | 1268 | – | – | 40938.17 | 1342.71 | **40994** | 53580.4838 |
| | | | 0.1 | **41091** | 1.82s | | **41091** | 1264 | – | – | 41054.23 | 1342.19 | 41090.9333 | 62323.5451 |
| | | 0.05 | 0.001 | 38742 | 38742 | 38742 | **38743** | 1414 | – | – | 38712.4 | 1328.13 | **38743** | 53287.6329 |
| | | | 0.01 | **40463** | **40463** | **40463** | **40463** | 1587 | – | – | 40348.77 | 1339.58 | **40463** | 55654.3928 |
| | | | 0.1 | **40985** | 82.48s | | **40985** | 2100 | – | – | 40869.57 | 1345.31 | **40985** | 51387.5036 |
| | | 0.1 | 0.001 | 35650 | 35978 | 36305 | **36390** | 572 | – | – | 36317.6 | 1314.06 | **36390** | 42624.4348 |
| | | | 0.01 | **39628** | **39628** | **39628** | **39628** | 771 | – | – | 39587.9 | 1333.85 | **39628** | 48509.0626 |
| | | | 0.1 | 40668 | 40668 | 40668 | **40672** | 1047 | – | – | 40649.03 | 1342.19 | 40671.8 | 50373.7936 |

TABLE 4.5. Statistical results for the instance eil101 with 500 items based on multiplicative uniform distribution and Chebyshev's inequality

| | Capacity | $\beta$ | $\alpha$ | ILP | | | (1+1)EA(1) | | | GSEMO(2) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 2mins | 6mins | 10mins | profit | time(ms) | stat | profits | time(ms) | stat |
| bou-s-c 1 | 61447 | 0.01 | 0.001 | **77951** | **77951** | **77951** | 76680.71 | 5941.67 | 2(-) | 77364.63 | 38885.42 | 1(+) |
| | | | 0.01 | 78649 | 78666 | **78749** | 77757.42 | 5928.65 | 2(-) | 78423.77 | 39448.44 | 1(+) |
| | | | 0.1 | **79411** | **79411** | **79411** | 77937.93 | 5945.83 | 2(-) | 78774.67 | 39529.69 | 1(+) |
| | | 0.05 | 0.001 | 71001 | 71825 | 71825 | 71273.63 | 5932.81 | 2(-) | **71866.73** | 38789.06 | 1(+) |
| | | | 0.01 | **76766** | **76766** | **76766** | 75786.83 | 5937.5 | 2(-) | 76556.33 | 39010.94 | 1(+) |
| | | | 0.1 | **78751** | **78751** | **78751** | 77472.63 | 5928.13 | 2(-) | 78160.3 | 39192.19 | 1(+) |
| | | 0.1 | 0.001 | 44200 | 49668 | 49672 | 65449.73 | 5949.48 | 2(-) | **66057.46** | 37460.94 | 1(+) |
| | | | 0.01 | 45378 | **74709** | **74709** | 73659.27 | 5943.23 | 2(-) | 74308.9 | 38843.75 | 1(+) |
| | | | 0.1 | 77951 | 77951 | **78011** | 76666.37 | 5928.13 | 2(-) | 77456.63 | 39206.25 | 1(+) |
| bou-s-c 2 | 162943 | 0.01 | 0.001 | 188693 | **189540** | **189540** | 187857.09 | 6138.54 | 2(-) | 187986.63 | 62129.69 | 1(+) |
| | | | 0.01 | **191716** | **191716** | **191716** | 190031.37 | 6133.33 | 2(-) | 190110.66 | 62969.79 | 1(+) |
| | | | 0.1 | 192263 | 192329 | **192437** | 190653.74 | 6133.33 | 2(-) | 190824.75 | 63540.1 | 1(+) |
| | | 0.05 | 0.001 | 175165 | 175870 | 175870 | 176974.32 | 6151.04 | 2(-) | **177249.9** | 61518.04 | 1(+) |
| | | | 0.01 | 187042 | **187614** | **187614** | 186179.49 | 6133.85 | 2(-) | 186342.34 | 62348.44 | 1(+) |
| | | | 0.1 | 191045 | **191098** | **191098** | 189401.84 | 6130.73 | 2(-) | 189591.47 | 62891.67 | 1(+) |
| | | 0.1 | 0.001 | 120607 | 147427 | 155490 | 165606.16 | 6167.19 | 2(-) | **165991.14** | 57807.81 | 1(+) |
| | | | 0.01 | 181967 | **182660** | **182660** | 181746.03 | 6143.23 | 2(-) | 181972.58 | 61254.69 | 1(+) |
| | | | 0.1 | **189477** | **189477** | **189477** | 188041.25 | 6126.56 | 2(-) | 188168.17 | 62590.63 | 1(+) |
| bou-s-c 3 | 307286 | 0.01 | 0.001 | 343297 | **343820** | **343820** | 341568.23 | 6425 | 2(-) | 342432.33 | 85480.73 | 1(+) |
| | | | 0.01 | 346271 | 347015 | **347145** | 344745.61 | 6421.83 | 2(-) | 344843.61 | 85917.19 | 1(+) |
| | | | 0.1 | 347820 | **348151** | **348151** | 345852.22 | 6406.79 | 2(-) | 345952.21 | 86294.27 | 1(+) |
| | | 0.05 | 0.001 | 168902 | 168902 | 203124 | 324236.08 | 6435.94 | 2(-) | **324561.56** | 81405.73 | 1(+) |
| | | | 0.01 | 199323 | 248792 | 311808 | 338751.13 | 6425.52 | 2(-) | **338830.56** | 85056.25 | 1(+) |
| | | | 0.1 | **346271** | **346271** | **346271** | 343994.57 | 6425 | 2(-) | 344849.08 | 86061.98 | 1(+) |
| | | 0.1 | 0.001 | 167882 | 231201 | 231201 | 305712.67 | 6461.98 | 2(-) | **306298.87** | 76398.96 | 1(+) |
| | | | 0.01 | 169122 | 254642 | 254642 | 331962.13 | 6431.77 | 2(-) | **331991.33** | 82465.1 | 1(+) |
| | | | 0.1 | 200994 | 341453 | **343392** | 341835.52 | 6425 | 2(-) | 341914.53 | 84382.29 | 1(+) |
| uncorr 1 | 37686 | 0.01 | 0.001 | 87252 | 87333 | **87465** | 86527.83 | 5964.58 | 2(-) | 87259.37 | 22572.92 | 1(+) |
| | | | 0.01 | 88093 | **88104** | **88104** | 87167.43 | 5961.46 | 2(-) | 87897.4 | 22578.13 | 1(+) |
| | | | 0.1 | **88311** | **88311** | **88311** | 87314.5 | 5963.54 | 2(-) | 88099.57 | 22747.4 | 1(+) |
| | | 0.05 | 0.001 | 82872 | 82872 | 82872 | 82862.63 | 5951.56 | 2(-) | **83631.43** | 22251.56 | 1(+) |
| | | | 0.01 | **86847** | **86847** | **86847** | 85984.17 | 5953.65 | 2(-) | 86720.27 | 22438.54 | 1(+) |
| | | | 0.1 | 16687 | 87904 | **87926** | 87003.1 | 5959.38 | 2(-) | 87759.4 | 22661.98 | 1(+) |
| | | 0.1 | 0.001 | 16590 | 58953 | 75707 | 78590.73 | 5934.38 | 2(-) | **79491.4** | 21914.06 | 1(+) |
| | | | 0.01 | 84948 | **85449** | **85449** | 84462.23 | 5947.92 | 2(-) | 85287.3 | 22295.83 | 1(+) |
| | | | 0.1 | 16687 | 87253 | **87489** | 86561.52 | 5955.21 | 2(-) | 87314.33 | 22748.96 | 1(+) |
| uncorr 2 | 93559 | 0.01 | 0.001 | 101601 | **148534** | **148534** | 147858.42 | 6346.31 | 2(-) | 147978.3 | 30017.71 | 1(+) |
| | | | 0.01 | 37741 | 149398 | **149403** | 148717.23 | 6253.65 | 2(-) | 148800.6 | 30072.4 | 1(+) |
| | | | 0.1 | 37741 | 149695 | **149696** | 149021.6 | 6248.44 | 2(-) | 149097 | 29593.75 | 1(+) |
| | | 0.05 | 0.001 | 101527 | **143671** | **143671** | 143106.43 | 6220.31 | 2(-) | **143671** | 98357.81 | 1(+) |
| | | | 0.01 | 101593 | 142271 | **147820** | 147109.2 | 6248.44 | 2(-) | 147676.7 | 99925.52 | 1(+) |
| | | | 0.1 | 148834 | 148834 | **149218** | 148500.83 | 6247.4 | 2(-) | 149050.2 | 101393.23 | 1(+) |
| | | 0.1 | 0.001 | 37939 | 114416 | 114416 | 137665.33 | 6257.29 | 2(-) | **137942.4** | 28286.98 | 1(+) |
| | | | 0.01 | 101587 | 143082 | **145893** | 145231.46 | 6267.71 | 2(-) | 145816 | 99587.5 | 1(+) |
| | | | 0.1 | 148605 | **148613** | **148613** | 147931.4 | 6655.73 | 2(-) | 148450 | 100575.52 | 1(+) |
| uncorr 3 | 171819 | 0.01 | 0.001 | 69026 | **208530** | **208530** | 207914.33 | 6685.42 | 2(-) | 207955.3 | 114927.6 | 1(+) |
| | | | 0.01 | 69076 | **209489** | **209489** | 208724.56 | 6684.8 | 2(-) | 208737 | 114892.71 | 1(+) |
| | | | 0.1 | 69076 | **209734** | **209734** | 208963.6 | 6617.71 | 2(-) | 208980.9 | 114864.58 | 1(+) |
| | | 0.05 | 0.001 | 68118 | 141171 | 156856 | 203217.33 | 6648.96 | 2(-) | **203416** | 113179.69 | 1(+) |
| | | | 0.01 | 129569 | 207365 | **207985** | 207275.63 | 6617.71 | 2(-) | 207315.8 | 114427.6 | 1(+) |
| | | | 0.1 | 129871 | 209229 | **209255** | 208519.46 | 6657.29 | 2(-) | 208567 | 114638.02 | 1(+) |
| | | 0.1 | 0.001 | 66564 | 142764 | 154311 | 197404.69 | 6572.4 | 2(-) | **197631.8** | 111736.46 | 1(+) |
| | | | 0.01 | 68197 | 204818 | **206030** | 205424.63 | 6635.42 | 2(-) | 205511.3 | 113712.5 | 1(+) |
| | | | 0.1 | 115721 | 208706 | **208706** | 208053.79 | 6643.23 | 2(-) | 208019.7 | 114743.23 | 1(+) |

Table 4.6. Statistical results for the instance eil101 with 100 items based on Normal distribution and Chebyshev's inequality

| | Capacity | $\beta$ | $\alpha$ | ILP | | | Heuristic approach | | DP | | (1+1)EA | | GSEMO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 2mins | 6mins | 10mins | profit | time(ms) | profit | time(ms) | profit | time(ms) | profits | time(ms) |
| bou-s-c 1 | 11775 | 0.01 | 0.001 | **15635** | **15635** | **15635** | **15635** | 1074 | – | – | 15481.8 | 1227.083 | **15635** | 122546.3 |
| | | | 0.01 | **15946** | 59.92s | | **15946** | 1836 | – | – | 15759.4 | 1232.813 | **15946** | 119314.8 |
| | | | 0.1 | **15946** | 59.51s | | **15946** | 1211 | – | – | 15862.53 | 1223.958 | **15946** | 114167.2 |
| | | 0.05 | 0.001 | 15114 | 15114 | 15114 | **15128** | 1316 | – | – | 14971.97 | 1224.479 | **15128** | 95547.55 |
| | | | 0.01 | 15732 | 15732 | 15732 | **15736** | 1360 | – | – | 15631.53 | 1222.917 | **15736** | 98784.17 |
| | | | 0.1 | **15946** | 16.88s | | **15946** | 1546 | – | – | 15789.63 | 1225 | **15946** | 113056.9 |
| | | 0.1 | 0.001 | 14719 | 14719 | 14719 | **14739** | 971 | – | – | 14631.63 | 1224.479 | **14739** | 100566.2 |
| | | | 0.01 | 15630 | 15630 | 15630 | **15635** | 2152 | – | – | 15480.13 | 1226.042 | **15635** | 110594.3 |
| | | | 0.1 | **15946** | 13.94s | | **15946** | 1634 | – | – | 15758.67 | 1221.354 | **15946** | 112109.6 |
| bou-s-c 2 | 31027 | 0.01 | 0.001 | **37359** | **37359** | **37359** | **37359** | 22126 | – | – | 37159.3 | 1271.35 | 37374.47 | 326773.7 |
| | | | 0.01 | 37272 | 37272 | 37751 | **37752** | 28381 | – | – | 37586.77 | 1265.625 | 37751.9 | 265563.1 |
| | | | 0.1 | **37874** | 36.48s | | **37874** | 27277 | – | – | 37703.33 | 1268.75 | **37874** | 399498.2 |
| | | 0.05 | 0.001 | 36539 | 36539 | 36539 | **36606** | 20569 | – | – | 36425.97 | 1269.271 | 36605.83 | 446766.4 |
| | | | 0.01 | 37509 | 37515 | 37515 | **37537** | 22290 | – | – | 37345.97 | 1274.479 | 37536.07 | 541687.4 |
| | | | 0.1 | 37776 | 37791 | 37791 | **37809** | 27942 | – | – | 37622.23 | 1266.667 | 37808.97 | 523256 |
| | | 0.1 | 0.001 | 35982 | 36000 | 36000 | **36103** | 35383 | – | – | 35857.93 | 1281.771 | 36100.5 | 336191.5 |
| | | | 0.01 | 37272 | 37303 | 37303 | **37375** | 32159 | – | – | 37163.57 | 1286.979 | 37374.5 | 292882 |
| | | | 0.1 | 37752 | 37752 | 37752 | **37760** | 34877 | – | – | 37587.67 | 1266.667 | **37760** | 166735.9 |
| bou-s-c 3 | 58455 | 0.01 | 0.001 | 66883 | 66883 | 66889 | **66895** | 263962 | – | – | 66633 | 1310.93 | 66856.3 | 253407.8 |
| | | | 0.01 | 67402 | 67402 | 67402 | **67414** | 307935 | – | – | 67214.13 | 1410.93 | 67412.17 | 252614.6 |
| | | | 0.1 | 67581 | 67581 | 67581 | **67582** | 217621 | – | – | 67395.43 | 1314.583 | 67580.73 | 250042.7 |
| | | 0.05 | 0.001 | 65863 | 65865 | 65865 | **65871** | 178900 | – | – | 65674.87 | 1324.479 | 65868.4 | 244173.4 |
| | | | 0.01 | 67067 | 67085 | 67085 | **67119** | 226313 | – | – | 66903.47 | 1328.125 | 67104.73 | 250305.2 |
| | | | 0.1 | 67492 | 67492 | 67492 | **67493** | 226818 | – | – | 67284.2 | 1314.583 | 67491.37 | 250183.9 |
| | | 0.1 | 0.001 | 65181 | **65187** | **65187** | **65187** | 193779 | – | – | 64911.7 | 1317.188 | 65102.6 | 241117.2 |
| | | | 0.01 | 66883 | 66883 | 66883 | **66899** | 230572 | – | – | 66675.67 | 1311.458 | 66851.1 | 249634.4 |
| | | | 0.1 | 67425 | 67425 | 67425 | **67426** | 215087 | – | – | 67230.03 | 1327.083 | 67423.43 | 246294.3 |
| uncorr 1 | 7715 | 0.01 | 0.001 | **17151** | 155.74s | | **17151** | 170 | **17151** | 62462460 | 17099.73 | 1230.208 | **17151** | 7984.781 |
| | | | 0.01 | **17366** | 5.22s | | **17366** | 153 | **17366** | 74021365 | 17333.5 | 1243.75 | **17366** | 7432.983 |
| | | | 0.1 | **17499** | 1.92s | | **17499** | 342 | **17499** | 81548280 | 17445.2 | 1322.396 | **17499** | 6449.72 |
| | | 0.05 | 0.001 | **16638** | **16638** | **16638** | **16638** | 127 | **16638** | 46226151 | 16571.3 | 1227.604 | **16638** | 8861.319 |
| | | | 0.01 | **17247** | 15.46s | | **17247** | 109 | **17247** | 66880112 | 17167.67 | 1229.167 | **17247** | 6220.258 |
| | | | 0.1 | **17408** | 3.77s | | **17408** | 164 | **17408** | 71001731 | 17373.7 | 1199.688 | **17408** | 7471.837 |
| | | 0.1 | 0.001 | **16266** | 1.23s | | **16266** | 80 | **16266** | 44301629 | 16177.27 | 1229.167 | **16266** | 6806.991 |
| | | | 0.01 | **17151** | 58.59s | | **17151** | 672 | **17151** | 50929605 | 17090.43 | 1247.604 | **17151** | 6305.284 |
| | | | 0.1 | **17366** | 3.86s | | **17366** | 688 | **17366** | 66415177 | 17338.23 | 1242.083 | **17366** | 8335.682 |
| uncorr 2 | 19545 | 0.01 | 0.001 | 29248 | 29248 | 29248 | **29250** | 610 | – | – | 29159.13 | 1276.042 | **29250** | 20153.41 |
| | | | 0.01 | **29508** | 9.98s | | **29508** | 658 | – | – | 29401.57 | 1166.042 | **29508** | 19399.76 |
| | | | 0.1 | **29595** | 2.87s | | **29595** | 586 | – | – | 29512.8 | 1277.083 | **29595** | 20561.9 |
| | | 0.05 | 0.001 | **28766** | 2.45s | | **28766** | 691 | – | – | 28312.4 | 1276.563 | **28766** | 19775.83 |
| | | | 0.01 | **29415** | 5.78s | | **29415** | 584 | – | – | 29159.7 | 1278.125 | **29415** | 21976.75 |
| | | | 0.1 | **29585** | 4.6s | | **29585** | 1048 | – | – | 29446.37 | 1273.438 | **29585** | 20835.11 |
| | | 0.1 | 0.001 | **28472** | 5.18s | | **28472** | 1050 | – | – | 28379.33 | 1273.958 | **28472** | 23583.94 |
| | | | 0.01 | 29248 | 29248 | 29248 | **29250** | 516 | – | – | 29117.5 | 1276.563 | **29250** | 23508.19 |
| | | | 0.1 | **29508** | 8.01s | | **29508** | 581 | – | – | 29505.1 | 1276.042 | **29508** | 18264.97 |
| uncorr 3 | 36091 | 0.01 | 0.001 | 40858 | 105.48s | | 40858 | 1190 | – | – | 40785.7 | 1448.958 | 40858 | 59528.52 |
| | | | 0.01 | **41084** | 2.75s | | **41084** | 881 | – | – | 41022.8 | 1348.958 | **41084** | 70468.21 |
| | | | 0.1 | **41121** | 4.3s | | **41121** | 1721 | – | – | 41069.2 | 1346.875 | 41120.53 | 64887.42 |
| | | 0.05 | 0.001 | **40482** | **40482** | **40482** | **40482** | 2445 | – | – | 40124.27 | 1342.188 | **40482** | 67218.26 |
| | | | 0.01 | **40985** | 58.57s | | **40985** | 1562 | – | – | 40778.7 | 1346.354 | **40985** | 68263.79 |
| | | | 0.1 | **41090** | 4.91s | | **41090** | 1121 | – | – | 40995.8 | 1348.958 | **41090** | 60929.71 |
| | | 0.1 | 0.001 | **40150** | **40150** | **40150** | **40150** | 1597 | – | – | 40096.6 | 1341.667 | 40148.83 | 57693.78 |
| | | | 0.01 | **40858** | **40858** | **40858** | **40858** | 1422 | – | – | 40845.53 | 1346.875 | **40858** | 58638.7 |
| | | | 0.1 | **41084** | 5.77s | | **41084** | 1305 | – | – | 41059.5 | 1347.917 | **41084** | 58810.93 |

Table 4.7. Statistical results for the instance eil101 with 500 items based on Normal distribution and Chebyshev's inequality

| | Capacity | $\beta$ | $\alpha$ | ILP | | | (1+1)EA (1) | | | GSEMO (2) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 2mins | 6mins | 10mins | profit | time(ms) | stat | profits | time(ms) | stat |
| bou-s-c 1 | 61447 | 0.01 | 0.001 | **78649** | **78649** | **78649** | 75578.23 | 10723.1 | 2(-) | 77967.43 | 64191.74 | 1(+) |
| | | | 0.01 | 78649 | **78991** | **78991** | 77276.7 | 11453.57 | 2(-) | 78614.93 | 65536.54 | 1(+) |
| | | | 0.1 | **79471** | **79471** | **79471** | 77930.87 | 12345.55 | 2(-) | 78775.83 | 65576.01 | 1(+) |
| | | 0.05 | 0.001 | **77515** | **77515** | **77515** | 66902.87 | 11525.26 | 2(-) | 76868.6 | 64167.35 | 1(+) |
| | | | 0.01 | 78649 | 78823 | **78897** | 74202.93 | 11487.28 | 2(-) | 78222.2 | 64482.46 | 1(+) |
| | | | 0.1 | 79095 | 79275 | **79280** | 76840.73 | 8534.408 | 2(-) | 78664.87 | 65810.88 | 1(+) |
| | | 0.1 | 0.001 | 45784 | 73809 | **76045** | 58576.47 | 9455.142 | 2(-) | 76069.83 | 64195.89 | 1(+) |
| | | | 0.01 | **78649** | **78649** | **78649** | 70790.93 | 11302.55 | 2(-) | 77959.23 | 64669.14 | 1(+) |
| | | | 0.1 | 79024 | 79024 | **79108** | 75677.07 | 14396.65 | 2(-) | 78590.93 | 65864.06 | 1(+) |
| bou-s-c 2 | 162943 | 0.01 | 0.001 | 190595 | 190630 | **190992** | 185699.2 | 14038.99 | 2(-) | 189585.9 | 106708.7 | 1(+) |
| | | | 0.01 | **192267** | **192267** | **192267** | 189245.8 | 13191.94 | 2(-) | 190586.1 | 107000.5 | 1(+) |
| | | | 0.1 | 192091 | 192445 | **192587** | 190476.9 | 12174.11 | 2(-) | 190861.6 | 109597.3 | 1(+) |
| | | 0.05 | 0.001 | **189477** | **189477** | **189477** | 168475.9 | 12581.75 | 2(-) | 187893.3 | 106447.4 | 1(+) |
| | | | 0.01 | **191739** | **191739** | **191739** | 182869 | 12546.28 | 2(-) | 190051.8 | 107138.7 | 1(+) |
| | | | 0.1 | 192200 | 192340 | **192385** | 188430.7 | 12950.24 | 2(-) | 190718.1 | 109228.8 | 1(+) |
| | | 0.1 | 0.001 | 187042 | **187826** | **187826** | 151777.8 | 12721.25 | 2(-) | 186650.7 | 198824.9 | 1(+) |
| | | | 0.01 | 190595 | 190595 | **190606** | 175823.3 | 12746.82 | 2(-) | 189036.6 | 153722.1 | 1(+) |
| | | | 0.1 | **192252** | **192252** | **192252** | 185978 | 12273.07 | 2(-) | 190568.9 | 109722.5 | 1(+) |
| bou-s-c 3 | 307286 | 0.01 | 0.001 | 346237 | **346560** | **346560** | 338244.3 | 12581.56 | 2(-) | 343847.3 | 258392 | 1(+) |
| | | | 0.01 | 347893 | **348099** | **348099** | 343735.8 | 12276.5 | 2(-) | 345168.6 | 177581.7 | 1(+) |
| | | | 0.1 | 348242 | 348411 | **348417** | 345411 | 12445.5 | 2(-) | 345577.9 | 176647.7 | 1(+) |
| | | 0.05 | 0.001 | 200259 | 343015 | **344569** | 310428.7 | 12381.02 | 2(-) | 341663.5 | 161079.6 | 1(+) |
| | | | 0.01 | 346222 | **346473** | **346473** | 333825.5 | 12646.84 | 2(-) | 344489.5 | 156963.6 | 1(+) |
| | | | 0.1 | 347997 | **348128** | **348212** | 342423.9 | 8960.956 | 2(-) | 345352.1 | 152797.6 | 1(+) |
| | | 0.1 | 0.001 | 202187 | **341627** | **341627** | 283131.2 | 12028.49 | 2(-) | 339939.8 | 301736 | 1(+) |
| | | | 0.01 | 346271 | **346767** | **346767** | 322537.6 | 12357.88 | 2(-) | 343863.1 | 156330.3 | 1(+) |
| | | | 0.1 | **348020** | **348020** | **348020** | 338710.1 | 12106.73 | 2(-) | 345162 | 156225.4 | 1(+) |
| uncorr 1 | 37686 | 0.01 | 0.001 | 87489 | **87515** | **87515** | 85923.03 | 10130.34 | 2(-) | 87311.57 | 49110.62 | 1(+) |
| | | | 0.01 | **88169** | **88169** | **88169** | 86916.9 | 11220.51 | 2(-) | 87887.03 | 55903.9 | 1(+) |
| | | | 0.1 | **88331** | **88331** | **88331** | 87354.8 | 11674.73 | 2(-) | 88078.5 | 23058.93 | 1(+) |
| | | 0.05 | 0.001 | **86431** | **86431** | **86431** | 79797.07 | 11502.3 | 2(-) | 86271.57 | 53259.75 | 1(+) |
| | | | 0.01 | 87252 | 87729 | **87765** | 84955.23 | 11715.79 | 2(-) | 87578.73 | 51830.28 | 1(+) |
| | | | 0.1 | 88169 | **88236** | **88236** | 86697.47 | 12063.41 | 2(-) | 87977.4 | 23219.8 | 1(+) |
| | | 0.1 | 0.001 | 84636 | **85738** | **85738** | 73279.37 | 13354.03 | 2(-) | 85481.67 | 30391.65 | 1(+) |
| | | | 0.01 | 16687 | **87513** | **87513** | 82380.23 | 12894.84 | 2(-) | 87314.6 | 30685.76 | 1(+) |
| | | | 0.1 | **88169** | **88169** | **88169** | 85934.5 | 11955.7 | 2(-) | 87913.5 | 26798.67 | 1(+) |
| uncorr 2 | 93559 | 0.01 | 0.001 | **148834** | **148834** | **148834** | 146900.1 | 12483.14 | 2(-) | 148232.8 | 42380.86 | 1(+) |
| | | | 0.01 | 149445 | **149516** | **149516** | 148429.4 | 12386.6 | 2(-) | 148769.8 | 42206.33 | 1(+) |
| | | | 0.1 | 149726 | **149730** | **149730** | 148869 | 12523.58 | 2(-) | 148951.5 | 38688.04 | 1(+) |
| | | 0.05 | 0.001 | 37741 | 147694 | **147780** | 139095.4 | 12240.31 | 2(-) | 147102.9 | 41993.5 | 1(+) |
| | | | 0.01 | 148834 | **148934** | **148934** | 145768.8 | 11206.09 | 2(-) | 148419.6 | 42218.72 | 1(+) |
| | | | 0.1 | 148834 | **149625** | **149625** | 148125.6 | 14307.55 | 2(-) | 148839.1 | 39514.93 | 1(+) |
| | | 0.1 | 0.001 | 101587 | 146281 | **146888** | 130418.7 | 13508.59 | 2(-) | 146250.9 | 41917.14 | 1(+) |
| | | | 0.01 | 37741 | **148834** | **148834** | 142598.9 | 11929.05 | 2(-) | 148163.2 | 41993.42 | 1(+) |
| | | | 0.1 | 149398 | **149543** | **149543** | 147111.4 | 12399.31 | 2(-) | 148791 | 39720.63 | 1(+) |
| uncorr 3 | 171819 | 0.01 | 0.001 | 208811 | **209074** | **209074** | 207151.1 | 12162.4 | 2(-) | 206872.6 | 49345.98 | 1(+) |
| | | | 0.01 | 209630 | **209630** | **209630** | 208489.4 | 13234.64 | 2(-) | 207338.3 | 49400.53 | 1(+) |
| | | | 0.1 | 209790 | **209790** | **209790** | 208946.9 | 14540.8 | 2(-) | 207487.1 | 49227.39 | 1(+) |
| | | 0.05 | 0.001 | 141224 | 208033 | **208095** | 199039.4 | 12712.34 | 2(-) | 205892.9 | 49377.69 | 1(+) |
| | | | 0.01 | 209229 | **209289** | **209289** | 205913.6 | 12785.77 | 2(-) | 206973 | 49178.8 | 1(+) |
| | | | 0.1 | 209663 | **209694** | **209694** | 208144.3 | 12656.35 | 2(-) | 207608.4 | 49367.14 | 1(+) |
| | | 0.1 | 0.001 | 141356 | 144160 | **207294** | 188980.9 | 12458.81 | 2(-) | 205302.1 | 48909.96 | 1(+) |
| | | | 0.01 | 141561 | **209047** | **209047** | 202708.7 | 13160.75 | 2(-) | 206906.2 | 49180.67 | 1(+) |
| | | | 0.1 | 209630 | **209630** | **209630** | 207203.3 | 12552.59 | 2(-) | 207316.4 | 49094.54 | 1(+) |

Table 4.8. Statistical results of $(1 + 1)$ EA with Chernoff bound for instance eil101 with 500 items

| | capacity | delta | alpha | Standard $(1+1)$ EA (1) | | | $(1+1)$ EA with HT (2) | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Mean | Std | stat | Mean | Std | stat |
| bou-s-c 1 | 61447 | 25 | 0.001 | 77188.75 | 131.32 | 2(-) | 77354.80 | 137.75 | **1**(+) |
| | | | 0.01 | 77431.35 | 217.91 | 2(-) | 77682.50 | 142.07 | **1**(+) |
| | | | 0.1 | 77846.90 | 149.80 | 2(-) | 78046.45 | 101.43 | **1**(+) |
| | | 50 | 0.001 | 75625.30 | 114.08 | 2(-) | 75796.10 | 124.82 | **1**(+) |
| | | | 0.01 | 76189.05 | 168.81 | 2(-) | 76429.60 | 164.43 | **1**(+) |
| | | | 0.1 | 76990.85 | 130.88 | 2(-) | 77190.65 | 147.35 | **1**(+) |
| bou-s-c 2 | 162943 | 25 | 0.001 | 189768.50 | 176.14 | 2(-) | 190192.30 | 109.27 | **1**(+) |
| | | | 0.01 | 190136.65 | 146.38 | 2(-) | 190435.60 | 152.04 | **1**(+) |
| | | | 0.1 | 190668.95 | 164.61 | 2(-) | 190889.80 | 138.44 | **1**(+) |
| | | 50 | 0.001 | 187930.55 | 200.91 | 2(-) | 188244.65 | 87.04 | **1**(+) |
| | | | 0.01 | 188636.25 | 185.64 | 2(-) | 189002.70 | 157.02 | **1**(+) |
| | | | 0.1 | 189560.60 | 185.47 | 2(-) | 189882.45 | 134.64 | **1**(+) |
| uncorr 1 | 37686 | 25 | 0.001 | 85793.80 | 141.97 | 2(-) | 85905.00 | 125.82 | **1**(+) |
| | | | 0.01 | 86163.70 | 152.75 | 2(-) | 86323.45 | 103.78 | **1**(+) |
| | | | 0.1 | 86735.10 | 107.89 | 2(-) | 86887.80 | 87.51 | **1**(+) |
| | | 50 | 0.001 | 83617.85 | 175.42 | 2(-) | 83746.00 | 72.36 | **1**(+) |
| | | | 0.01 | 84400.05 | 131.08 | 2(-) | 84556.10 | 117.32 | **1**(+) |
| | | | 0.1 | 85514.45 | 170.14 | 2(-) | 85668.30 | 88.90 | **1**(+) |
| uncorr 2 | 93559 | 25 | 0.001 | 147538.15 | 105.60 | 2(-) | 147693.65 | 45.21 | **1**(+) |
| | | | 0.01 | 147931.80 | 164.88 | 2(-) | 148048.80 | 64.67 | **1**(+) |
| | | | 0.1 | 148371.20 | 101.71 | 2(-) | 148515.65 | 76.13 | **1**(+) |
| | | 50 | 0.001 | 145675.90 | 73.28 | 2(-) | 145767.40 | 88.57 | **1**(+) |
| | | | 0.01 | 146381.05 | 123.87 | 2(-) | 146478.65 | 65.55 | **1**(+) |
| | | | 0.1 | 147311.05 | 98.65 | 2(-) | 147450.10 | 78.27 | **1**(+) |

Table 4.9. Statistical results of (1+1) EA with Chebyshev's inequality for instance eil101 with 500 items

| | capacity | delta | alpha | Standard $(1+1)$ EA (1) | | | $(1+1)$ EA with HT (2) | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Mean | Std | stat | Mean | Std | stat |
| bou-s-c 1 | 61447 | 25 | 0.001 | 73845.45 | 154.22 | 2(-) | 74030.20 | 106.77 | **1**(+) |
| | | | 0.01 | 77118.95 | 177.20 | 2(-) | 77288.75 | 112.60 | **1**(+) |
| | | | 0.1 | 78184.80 | 166.21 | 2(-) | 78353.85 | 143.99 | **1**(+) |
| | | 50 | 0.001 | 69136.30 | 210.29 | 2(-) | 69394.60 | 121.15 | **1**(+) |
| | | | 0.01 | 75619.85 | 159.61 | 2(-) | 75829.30 | 129.03 | **1**(+) |
| | | | 0.1 | 77712.80 | 201.23 | 2(-) | 77987.45 | 124.92 | **1**(+) |
| bou-s-c 2 | 162943 | 25 | 0.001 | 185706.25 | 151.89 | 2(-) | 185969.05 | 136.42 | **1**(+) |
| | | | 0.01 | 189753.70 | 166.73 | 2(-) | 190109.25 | 133.08 | **1**(+) |
| | | | 0.1 | 191092.25 | 182.75 | 2(-) | 191410.15 | 122.83 | **1**(+) |
| | | 50 | 0.001 | 179824.75 | 167.43 | 2(-) | 180107.35 | 126.69 | **1**(+) |
| | | | 0.01 | 187914.05 | 134.66 | 2(-) | 188176.30 | 162.26 | **1**(+) |
| | | | 0.1 | 190566.85 | 159.93 | 2(-) | 190784.95 | 119.70 | **1**(+) |
| uncorr 1 | 37686 | 25 | 0.001 | 80931.05 | 206.08 | 2(-) | 81114.65 | 108.74 | **1**(+) |
| | | | 0.01 | 85710.55 | 181.84 | 2(-) | 85857.05 | 130.49 | **1**(+) |
| | | | 0.1 | 87306.40 | 174.52 | 2(-) | 87405.90 | 134.22 | **1**(+) |
| | | 50 | 0.001 | 74345.30 | 232.35 | 2(-) | 74483.10 | 152.61 | **1**(+) |
| | | | 0.01 | 83497.20 | 132.89 | 2(-) | 83685.30 | 117.64 | **1**(+) |
| | | | 0.1 | 86617.35 | 146.49 | 2(-) | 86761.15 | 83.79 | **1**(+) |
| uncorr 2 | 93559 | 25 | 0.001 | 143213.25 | 87.61 | 2(-) | 143359.95 | 62.21 | **1**(+) |
| | | | 0.01 | 147487.20 | 110.01 | 2(-) | 147597.30 | 65.75 | **1**(+) |
| | | | 0.1 | 148827.55 | 142.91 | 2(-) | 148962.75 | 67.57 | **1**(+) |
| | | 50 | 0.001 | 137111.55 | 102.85 | 2(-) | 137262.15 | 75.68 | **1**(+) |
| | | | 0.01 | 145514.50 | 142.63 | 2(-) | 145636.50 | 55.13 | **1**(+) |
| | | | 0.1 | 148259.75 | 116.00 | 2(-) | 148367.25 | 64.25 | **1**(+) |

Table 4.10. Statistical results of $(\mu+1)$ EA with Chernoff bound for instance eil101 with 500 items

| | capacity | delta | alpha | Standard $(\mu+1)$ EA (3) | | | $(\mu+1)$ EA with HT (4) | | | $(\mu+1)$ EA with HT and PS (5) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Mean | Std | stat | Mean | Std | stat | Mean | Std | stat |
| bou-s-c 1 | 61447 | 25 | 0.001 | 77112.27 | 182.05 | 4(-),5(-) | 77350.17 | 108.33 | 3(+),5(-) | 77518.20 | 104.44 | **3**(+), **4**(+) |
| | | | 0.01 | 77413.97 | 188.46 | 4(-),5(-) | 77646.67 | 119.40 | 3(+),5(-) | 77811.80 | 107.17 | **3**(+), **4**(+) |
| | | | 0.1 | 77787.77 | 138.06 | 4(-),5(-) | 78000.57 | 133.04 | 3(+),5(-) | 78197.40 | 142.42 | **3**(+), **4**(+) |
| | | 50 | 0.001 | 75562.23 | 201.47 | 4(-),5(-) | 75816.83 | 141.98 | 3(+),5(-) | 75934.03 | 102.39 | **3**(+), **4**(+) |
| | | | 0.01 | 76178.43 | 156.15 | 4(-),5(-) | 76455.20 | 113.76 | 3(+),5(-) | 76559.90 | 138.41 | **3**(+), **4**(+) |
| | | | 0.1 | 76948.67 | 177.82 | 4(-),5(-) | 77193.60 | 136.70 | 3(+),5(-) | 77296.53 | 148.22 | **3**(+), **4**(+) |
| bou-s-c 2 | 162943 | 25 | 0.001 | 189795.90 | 164.86 | 4(-),5(-) | 190130.60 | 142.84 | 3(+),5(-) | 190418.70 | 155.28 | **3**(+), **4**(+) |
| | | | 0.01 | 190179.80 | 150.85 | 4(-),5(-) | 190523.13 | 118.16 | 3(+),5(-) | 190814.40 | 130.31 | **3**(+), **4**(+) |
| | | | 0.1 | 190569.93 | 194.37 | 4(-),5(-) | 190917.27 | 122.95 | 3(+),5(-) | 191286.05 | 127.97 | **3**(+), **4**(+) |
| | | 50 | 0.001 | 188027.17 | 138.12 | 4(-),5(-) | 188308.87 | 129.41 | 3(+),5(-) | 188575.15 | 146.49 | **3**(+), **4**(+) |
| | | | 0.01 | 188690.86 | 139.27 | 4(-),5(-) | 189016.46 | 112.01 | 3(+),5(-) | 189287.95 | 108.62 | **3**(+), **4**(+) |
| | | | 0.1 | 189574.83 | 184.21 | 4(-),5(-) | 189862.87 | 118.61 | 3(+),5(-) | 190207.65 | 130.52 | **3**(+), **4**(+) |
| uncorr 1 | 37686 | 25 | 0.001 | 85788.63 | 108.84 | 4(-),5(-) | 85938.83 | 113.10 | 3(+),5(-) | 85968.47 | 82.01 | **3**(+), **4**(+) |
| | | | 0.01 | 86198.93 | 147.91 | 4(-),5(-) | 86331.03 | 109.54 | 3(+),5(-) | 86395.43 | 87.11 | **3**(+), **4**(+) |
| | | | 0.1 | 86784.87 | 133.35 | 4(-),5(-) | 86887.37 | 104.28 | 3(+),5(-) | 86925.50 | 88.78 | **3**(+), **4**(+) |
| | | 50 | 0.001 | 83586.80 | 204.58 | 4(-),5(-) | 83745.63 | 101.20 | 3(+),5(-) | 83819.10 | 91.67 | **3**(+), **4**(+) |
| | | | 0.01 | 84476.20 | 143.76 | 4(-),5(-) | 84572.10 | 104.82 | 3(+),5(-) | 84641.60 | 126.83 | **3**(+), **4**(+) |
| | | | 0.1 | 85521.30 | 166.74 | 4(-),5(-) | 85654.07 | 105.95 | 3(+),5(-) | 85717.10 | 97.85 | **3**(+), **4**(+) |
| uncorr 2 | 93559 | 25 | 0.001 | 147537.37 | 128.76 | 4(-),5(-) | 147683.03 | 70.76 | 3(+),5(-) | 147745.95 | 76.22 | **3**(+), **4**(+) |
| | | | 0.01 | 147923.97 | 98.21 | 4(-),5(-) | 148032.67 | 65.28 | 3(+),5(-) | 148126.05 | 61.35 | **3**(+), **4**(+) |
| | | | 0.1 | 148388.83 | 106.95 | 4(-),5(-) | 148513.17 | 74.87 | 3(+),5(-) | 148604.60 | 53.52 | **3**(+), **4**(+) |
| | | 50 | 0.001 | 145648.43 | 90.76 | 4(-),5(-) | 145770.03 | 65.78 | 3(+),5(-) | 145838.10 | 59.74 | **3**(+), **4**(+) |
| | | | 0.01 | 146379.80 | 86.76 | 4(-),5(-) | 146513.47 | 57.61 | 3(+),5(-) | 146572.60 | 47.19 | **3**(+), **4**(+) |
| | | | 0.1 | 147316.30 | 91.46 | 4(-),5(-) | 147414.76 | 85.65 | 3(+),5(-) | 147497.10 | 60.68 | **3**(+), **4**(+) |

Table 4.11. Statistical results of $(\mu+1)$ EA with Chebyshev's inequality for instance eil101 with 500 items

| | capacity | delta | alpha | Standard $(\mu+1)$ EA (3) | | | $(\mu+1)$ EA with HT (4) | | | $(\mu+1)$ EA with HT and PS (5) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Mean | Std | stat | Mean | Std | stat | Mean | Std | stat |
| bou-s-c 1 | 61447 | 25 | 0.001 | 73881.73 | 175.53 | 4(-),5(-) | 74005.90 | 137.56 | 3(+),5(-) | 74076.60 | 153.03 | **3**(+), **4**(+) |
| | | | 0.01 | 77141.50 | 161.02 | 4(-),5(-) | 77376.87 | 105.29 | 3(+),5(-) | 77515.70 | 106.62 | **3**(+), **4**(+) |
| | | | 0.1 | 78171.23 | 212.15 | 4(-),5(-) | 78000.57 | 166.67 | 3(+),5(-) | 78620.00 | 135.42 | **3**(+), **4**(+) |
| | | 50 | 0.001 | 69178.33 | 151.69 | 4(-),5(-) | 69393.30 | 103.48 | 3(+),5(-) | 69439.10 | 110.33 | **3**(+), **4**(+) |
| | | | 0.01 | 75661.87 | 138.77 | 4(-),5(-) | 76455.20 | 121.03 | 3(+),5(-) | 75968.90 | 100.49 | **3**(+), **4**(+) |
| | | | 0.1 | 77735.60 | 201.61 | 4(-),5(-) | 77982.27 | 136.70 | 3(+),5(-) | 78157.37 | 98.79 | **3**(+), **4**(+) |
| bou-s-c 2 | 162943 | 25 | 0.001 | 185726.67 | 121.77 | 4(-),5(-) | 186003.77 | 117.99 | 3(+),5(-) | 186232.25 | 111.68 | **3**(+), **4**(+) |
| | | | 0.01 | 189717.60 | 163.28 | 4(-),5(-) | 190063.07 | 152.09 | 3(+),5(-) | 190416.70 | 117.58 | **3**(+), **4**(+) |
| | | | 0.1 | 191088.17 | 145.18 | 4(-),5(-) | 191362.76 | 117.11 | 3(+),5(-) | 191808.15 | 109.23 | **3**(+), **4**(+) |
| | | 50 | 0.001 | 179826.70 | 163.99 | 4(-),5(-) | 180083.90 | 118.26 | 3(+),5(-) | 180218.70 | 106.71 | **3**(+), **4**(+) |
| | | | 0.01 | 187871.70 | 153.35 | 4(-),5(-) | 188232.13 | 120.62 | 3(+),5(-) | 188546.80 | 139.90 | **3**(+), **4**(+) |
| | | | 0.1 | 190485.00 | 166.26 | 4(-),5(-) | 190831.97 | 131.35 | 3(+),5(-) | 191183.70 | 160.72 | **3**(+), **4**(+) |
| uncorr 1 | 37686 | 25 | 0.001 | 81049.37 | 140.15 | 4(-),5(-) | 81109.53 | 131.12 | 3(+),5(-) | 81184.30 | 109.41 | **3**(+), **4**(+) |
| | | | 0.01 | 85798.93 | 121.88 | 4(-),5(-) | 85898.80 | 97.51 | 3(+),5(-) | 85959.10 | 84.36 | **3**(+), **4**(+) |
| | | | 0.1 | 87322.67 | 142.69 | 4(-),5(-) | 87449.93 | 97.28 | 3(+),5(-) | 87486.60 | 82.56 | **3**(+), **4**(+) |
| | | 50 | 0.001 | 74378.33 | 174.08 | 4(-),5(-) | 74498.87 | 119.64 | 3(+),5(-) | 74597.45 | 120.22 | **3**(+), **4**(+) |
| | | | 0.01 | 83554.13 | 153.41 | 4(-),5(-) | 83665.77 | 104.76 | 3(+),5(-) | 83723.30 | 97.05 | **3**(+), **4**(+) |
| | | | 0.1 | 86634.27 | 150.50 | 4(-),5(-) | 86801.60 | 78.42 | 3(+),5(-) | 86872.40 | 88.62 | **3**(+), **4**(+) |
| uncorr 2 | 93559 | 25 | 0.001 | 143256.20 | 97.24 | 4(-),5(-) | 143350.47 | 72.07 | 3(+),5(-) | 143420.45 | 58.82 | **3**(+), **4**(+) |
| | | | 0.01 | 147521.47 | 81.86 | 4(-),5(-) | 148032.67 | 75.09 | 3(+),5(-) | 147703.15 | 53.83 | **3**(+), **4**(+) |
| | | | 0.1 | 148900.87 | 88.43 | 4(-),5(-) | 149009.90 | 81.65 | 3(+),5(-) | 149083.10 | 51.14 | **3**(+), **4**(+) |
| | | 50 | 0.001 | 137174.83 | 102.32 | 4(-),5(-) | 137303.53 | 64.42 | 3(+),5(-) | 137317.90 | 53.63 | **3**(+), **4**(+) |
| | | | 0.01 | 145525.53 | 111.41 | 4(-),5(-) | 145645.90 | 81.14 | 3(+),5(-) | 145719.65 | 37.79 | **3**(+), **4**(+) |
| | | | 0.1 | 148298.47 | 119.97 | 4(-),5(-) | 148409.80 | 62.97 | 3(+),5(-) | 148491.60 | 40.45 | **3**(+), **4**(+) |

Table 4.12. Statistical results of GSEMO with Chernoff bound for the instance eil101 with 500 items

| | capacity | delta | alpha | GSEMO with old model (6) | | | GSEMO with new model (7) | | | GSEMO with new model and PS (8) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Mean | Std | stat | Mean | Std | stat | Mean | Std | stat |
| bou-s-c 1 | 61447 | 25 | 0.001 | 77907.23 | 28.13 | 7(-),8(-) | 77921.57 | 30.63 | 6(+),8(-) | 77955.00 | 0.00 | 6(+),7(+) |
| | | | 0.01 | 78242.20 | 1.86 | | 78242.47 | 0.81 | | 78243.00 | 0.00 | |
| | | | 0.1 | 78592.00 | 15.55 | 8(-) | 78596.80 | 20.83 | 8(-) | 78649.00 | 0.00 | 6(+),7(+) |
| | | 50 | 0.001 | 76341.33 | 6.83 | | 76344.73 | 0.69 | | 76345.00 | 0.00 | |
| | | | 0.01 | 76955.33 | 9.48 | 7(-),8(-) | 76960.47 | 1.17 | 6(+) | 76961.00 | 0.00 | 6(+) |
| | | | 0.1 | 77721.87 | 2.45 | | 77722.80 | 0.81 | | 77723.00 | 0.00 | |
| bou-s-c 2 | 162943 | 25 | 0.001 | 190899.33 | 16.90 | 8(-) | 190901.23 | 8.68 | 8(-) | 190909.00 | 0.79 | 6(+),7(+) |
| | | | 0.01 | 191244.80 | 3.25 | | 191246.00 | 2.00 | | 191245.60 | 0.55 | |
| | | | 0.1 | 191682.97 | 24.79 | 8(-) | 191674.40 | 1.14 | 8(-) | 191759.00 | 0.00 | 6(+),7(+) |
| | | 50 | 0.001 | 188977.60 | 25.98 | 7(-),8(-) | 188985.40 | 9.24 | 6(+),8(-) | 188996.29 | 2.45 | 6(+),7(+) |
| | | | 0.01 | 189687.60 | 25.64 | 8(-) | 189687.80 | 24.96 | 8(-) | 189748.27 | 7.43 | 6(+),7(+) |
| | | | 0.1 | 190604.00 | 3.39 | 7(-),8(-) | 190609.80 | 8.64 | 6(+),8(-) | 190686.27 | 5.35 | 6(+),7(+) |
| uncorr 1 | 37686 | 25 | 0.001 | 86246.17 | 6.98 | | 86247.00 | 5.00 | | 86248.00 | 0.00 | |
| | | | 0.01 | 86653.47 | 1.66 | | 86654.00 | 0.00 | | 86654.00 | 0.00 | |
| | | | 0.1 | 87213.89 | 2.36 | | 87214.00 | 0.00 | | 87214.00 | 0.00 | |
| | | 50 | 0.001 | 84065.07 | 8.17 | | 84069.70 | 0.47 | | 84070.00 | 0.00 | |
| | | | 0.01 | 84909.97 | 10.33 | 8(-) | 84914.07 | 7.65 | | 84917.00 | 0.00 | 6(+) |
| | | | 0.1 | 85965.83 | 6.30 | | 85966.47 | 2.37 | | 85967.00 | 0.00 | |
| uncorr 2 | 93559 | 25 | 0.001 | 147885.00 | 3.36 | | 147888.60 | 0.89 | | 147889.00 | 0.00 | |
| | | | 0.01 | 148252.40 | 4.75 | | 148256.20 | 0.84 | | 148257.00 | 0.00 | |
| | | | 0.1 | 148720.55 | 3.83 | 8(-) | 148724.60 | 2.51 | | 148726.60 | 0.89 | 6(+) |
| | | 50 | 0.001 | 145964.00 | 5.00 | 8(-) | 145965.60 | 1.34 | 8(-) | 145973.58 | 6.89 | 6(+),7(+) |
| | | | 0.01 | 146696.00 | 8.80 | 8(-) | 146694.00 | 10.12 | 8(-) | 146710.00 | 0.00 | 6(+),7(+) |
| | | | 0.1 | 147623.80 | 10.06 | 7(-),8(-) | 147632.20 | 7.60 | 6(+),8(-) | 147642.00 | 0.00 | 6(+),7(+) |

TABLE 4.13. Statistical results of GSEMO with Chebyshev's inequality for the instance eil101 with 500 items

| | capacity | delta | alpha | GSEMO with old model (6) | | | GSEMO with new model (7) | | | GSEMO with new model and PS (8) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Mean | Std | stat | Mean | Std | stat | Mean | Std | stat |
| bou-s-c 1 | 61447 | 25 | 0.001 | 74505.00 | 11.76 | 7(-),8(-) | 74514.80 | 0.81 | 6(+) | 74515.00 | 0.00 | 6(+) |
| | | | 0.01 | 77882.23 | 5.67 | 8(-) | 77885.23 | 5.41 | 8(-) | 77953.00 | 0.00 | 6(+),7(+) |
| | | | 0.1 | 79026.23 | 3.73 | | 79028.10 | 1.06 | | 79029.00 | 0.00 | |
| | | 50 | 0.001 | 69852.37 | 29.36 | 7(-),8(-) | 69900.60 | 8.69 | 6(+),8(-) | 69925.00 | 0.00 | 6(+),7(+) |
| | | | 0.01 | 76327.47 | 13.59 | 7(-),8(-) | 76336.13 | 2.01 | 6(+) | 76337.00 | 0.00 | 6(+) |
| | | | 0.1 | 78517.57 | 3.83 | 8(-) | 78519.60 | 4.61 | 8(-) | 78525.00 | 0.00 | 6(+),7(+) |
| bou-s-c 2 | 162943 | 25 | 0.001 | 186609.20 | 21.00 | 7(-),8(-) | 186651.20 | 18.32 | 6(+),8(-) | 186666.00 | 5.10 | 6(+),7(+) |
| | | | 0.01 | 190795.60 | 27.49 | 7(-),8(-) | 190800.80 | 32.41 | 6(+),8(-) | 190866.00 | 8.78 | 6(+),7(+) |
| | | | 0.1 | 192177.97 | 35.88 | 7(-),8(-) | 192202.00 | 3.74 | 6(+),8(-) | 192207.20 | 0.45 | 6(+),7(+) |
| | | 50 | 0.001 | 180661.00 | 36.85 | 7(-),8(-) | 180696.80 | 10.55 | 6(+),8(-) | 180765.80 | 8.23 | 6(+),7(+) |
| | | | 0.01 | 188857.40 | 39.80 | 7(-),8(-) | 188888.40 | 27.14 | 6(+),8(-) | 188902.00 | 8.17 | 6(+),7(+) |
| | | | 0.1 | 191575.00 | 4.58 | 8(-) | 191576.00 | 1.87 | 8(-) | 191581.33 | 1.65 | 6(+),7(+) |
| uncorr 1 | 37686 | 25 | 0.001 | 81479.37 | 3.39 | | 81480.73 | 1.38 | | 81480.20 | 1.79 | |
| | | | 0.01 | 86206.30 | 1.66 | | 86208.77 | 4.48 | | 86210.00 | 0.00 | |
| | | | 0.1 | 87728.77 | 4.36 | | 87730.17 | 2.09 | | 87732.00 | 0.00 | |
| | | 50 | 0.001 | 74869.00 | 3.81 | | 74868.80 | 4.63 | | 74870.00 | 0.00 | |
| | | | 0.01 | 84013.50 | 6.40 | 7(-),8(-) | 84019.90 | 4.30 | 6(+) | 84022.00 | 0.00 | 6(+) |
| | | | 0.1 | 87066.73 | 5.30 | | 87068.70 | 0.47 | | 87069.00 | 0.00 | |
| uncorr 2 | 93559 | 25 | 0.001 | 143559.20 | 7.91 | 8(-) | 143557.60 | 6.58 | 8(-) | 143568.00 | 13.11 | 6(+),7(+) |
| | | | 0.01 | 147816.35 | 7.26 | 8(-) | 147821.20 | 2.77 | | 147825.40 | 1.34 | 6(+) |
| | | | 0.1 | 149211.15 | 5.85 | 8(-) | 149211.80 | 5.85 | 8(-) | 149218.00 | 0.00 | 6(+),7(+) |
| | | 50 | 0.001 | 137517.60 | 0.89 | | 137515.60 | 2.19 | | 137517.78 | 8.05 | |
| | | | 0.01 | 145843.40 | 2.97 | 7(-),8(-) | 145849.60 | 3.78 | 6(+),8(-) | 145855.14 | 22.98 | 6(+),7(+) |
| | | | 0.1 | 148603.60 | 3.29 | 7(-),8(-) | 148612.80 | 3.42 | 6(+) | 148614.90 | 2.78 | 6(+) |

TABLE 4.14. Statistical results of NSGA-II with Chernoff bound for instance eil101 with 500 items

| | capacity | delta | alpha | NSGA-II with old and uniform (9) | | | NSGA-II with old and PS (10) | | | NSGA-II with new and uniform (11) | | | NSGA-II with new and PS (12) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Mean | Std | stat | Mean | Std | stat | Mean | Std | stat | Mean | Std | stat |
| bou-s-c 1 | 61447 | 25 | 0.001 | 77504.93 | 148.92 | 10(-),11(-),12(-) | 77836.33 | 59.83 | 9(+),11(+),12(-) | 77724.13 | 97.36 | 9(+),10(-),12(-) | 77914.33 | 40.90 | 9(+),10(+),11(+) |
| | | | 0.01 | 77878.60 | 145.69 | 10(-),11(-),12(-) | 78154.63 | 60.99 | 9(+),11(+),12(-) | 78078.10 | 61.79 | 9(+),10(-),12(-) | 78220.37 | 30.01 | 9(+),10(+),11(+) |
| | | | 0.1 | 78267.03 | 120.50 | 10(-),11(-),12(-) | 78558.20 | 49.06 | 9(+),11(+),12(-) | 78464.67 | 85.56 | 9(+),10(-),12(-) | 78595.33 | 30.28 | 9(+),10(+),11(+) |
| | | 50 | 0.001 | 75911.17 | 145.99 | 10(-),11(-),12(-) | 76252.93 | 67.80 | 9(+),11(-),12(-) | 76698.10 | 89.40 | 9(+),10(+),12(-) | 76932.93 | 23.63 | 9(+),10(+),11(+) |
| | | | 0.01 | 76581.50 | 126.37 | 10(-),11(-),12(-) | 76883.90 | 61.50 | 9(+),11(+),12(-) | 76698.10 | 89.40 | 9(+),10(-),12(-) | 76932.93 | 23.63 | 9(+),10(+),11(+) |
| | | | 0.1 | 77439.80 | 101.89 | 10(-),11(-),12(-) | 77699.27 | 29.82 | 9(+),11(+),12(-) | 77534.43 | 72.53 | 9(+),10(-),12(-) | 77712.27 | 8.31 | 9(+),10(+),11(+) |
| bou-s-c 2 | 162943 | 25 | 0.001 | 190285.93 | 190.97 | 10(-),11(-),12(-) | 190841.83 | 54.12 | 9(+),11(+),12(-) | 190532.70 | 119.81 | 9(+),10(-),12(-) | 190888.73 | 12.46 | 9(+),10(+),11(+) |
| | | | 0.01 | 190686.43 | 182.27 | 10(-),11(-),12(-) | 191209.57 | 42.33 | 9(+),11(+),12(-) | 190937.13 | 88.80 | 9(+),10(-),12(-) | 191227.20 | 19.37 | 9(+),10(+),11(+) |
| | | | 0.1 | 191149.50 | 138.73 | 10(-),11(-),12(-) | 191549.50 | 18.71 | 9(+),11(+),12(-) | 191398.70 | 81.26 | 9(+),10(-),12(-) | 191693.53 | 42.38 | 9(+),10(+),11(+) |
| | | 50 | 0.001 | 188328.37 | 193.81 | 10(-),11(+),12(-) | 188952.87 | 41.35 | 9(+),11(+) | 188260.07 | 146.80 | 9(-),10(-),12(-) | 188953.47 | 12.58 | 9(+),10(+) |
| | | | 0.01 | 189111.80 | 175.58 | 10(-),11(+),12(-) | 189686.80 | 49.97 | 9(+),11(+),12(-) | 189076.40 | 140.02 | 9(-),10(-),12(-) | 189710.07 | 26.35 | 9(+),10(+),11(+) |
| | | | 0.1 | 190071.53 | 147.13 | 10(-),11(+),12(-) | 190623.40 | 41.52 | 9(+),11(+),12(-) | 190023.73 | 130.68 | 9(-),10(-),12(-) | 190718.33 | 42.59 | 9(+),10(+),11(+) |
| uncorr 1 | 37686 | 25 | 0.001 | 86132.83 | 65.96 | 10(-),11(-),12(-) | 86214.63 | 22.27 | 9(+),11(+) | 86158.40 | 41.90 | 9(+),10(-),12(-) | 86215.40 | 15.00 | 9(+),10(+) |
| | | | 0.01 | 86573.93 | 73.75 | 10(-),11(-),12(-) | 86645.23 | 14.75 | 9(+),11(+),12(-) | 86614.20 | 27.57 | 9(+),10(-),12(-) | 86651.50 | 9.52 | 9(+),10(+),11(+) |
| | | | 0.1 | 87128.00 | 53.90 | 10(-),11(-),12(-) | 87212.10 | 10.41 | 9(+),11(+) | 87162.83 | 29.00 | 9(+),10(-),12(-) | 87214.00 | 0.00 | 9(+),10(+) |
| | | 50 | 0.001 | 84039.10 | 31.64 | 10(-),11(-),12(-) | 84059.03 | 13.01 | 9(+),11(-),12(-) | 84627.23 | 150.14 | 9(+),10(+),12(-) | 84737.87 | 8.97 | 9(+),10(+),11(+) |
| | | | 0.01 | 84863.40 | 27.71 | 10(-),11(+),12(-) | 84892.63 | 7.00 | 9(+),11(+) | 84692.00 | 86.52 | 9(-),10(-),12(-) | 84895.63 | 3.30 | 9(+),10(+) |
| | | | 0.1 | 85929.30 | 29.21 | 10(-),11(+),12(-) | 85958.03 | 15.21 | 9(+),11(+),12(-) | 85859.10 | 56.61 | 9(-),10(-),12(-) | 85964.77 | 6.08 | 9(+),10(+),11(+) |
| uncorr 2 | 93559 | 25 | 0.001 | 147795.97 | 41.29 | 10(-),11(-),12(-) | 147878.17 | 8.42 | 9(+),11(+) | 147810.67 | 29.52 | 9(+),10(-),12(-) | 147875.37 | 9.02 | 9(+),10(+) |
| | | | 0.01 | 148168.07 | 41.88 | 10(-),11(-),12(-) | 148243.57 | 9.34 | 9(+),11(+),12(-) | 148183.73 | 31.21 | 9(+),10(-),12(-) | 148247.07 | 8.31 | 9(+),10(+),11(+) |
| | | | 0.1 | 148640.40 | 39.29 | 10(-),11(-),12(-) | 148714.30 | 9.19 | 9(+),11(+) | 148674.13 | 21.49 | 9(+),10(-),12(-) | 148713.33 | 7.75 | 9(+),10(+) |
| | | 50 | 0.001 | 145879.27 | 37.37 | 10(-),11(+),12(-) | 145959.83 | 9.18 | 9(+),11(+),12(-) | 145660.67 | 77.82 | 9(-),10(-),12(-) | 145975.67 | 13.97 | 9(+),10(+),11(+) |
| | | | 0.01 | 146609.63 | 51.45 | 10(-),11(+),12(-) | 146695.10 | 10.20 | 9(+),11(+) | 146508.33 | 43.19 | 9(-),10(-),12(-) | 146694.47 | 13.75 | 9(+),10(+) |
| | | | 0.1 | 147563.63 | 34.45 | 10(-),11(+),12(-) | 147623.67 | 9.87 | 9(+),11(+),12(-) | 147529.50 | 39.26 | 9(-),10(-),12(-) | 147628.07 | 5.29 | 9(+),10(+),11(+) |

Table 4.15. Statistical results of NSGA-II with Chebyshev's inequality for the instance eil101 with 500 items

| | capacity | delta | alpha | NSGA-II with old and uniform (9) | | | NSGA-II with old and PS (10) | | | NSGA-II with new and uniform (11) | | | NSGA-II with new and PS (12) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Mean | Std | stat | Mean | Std | stat | Mean | Std | stat | Mean | Std | stat |
| bou-s-c 1 | 61447 | 25 | 0.001 | 73833.80 | 142.46 | 10(-),11(+),12(-) | 74461.90 | 22.06 | 9(+),11(+),12(+) | 73566.07 | 214.57 | 9(-),10(-),12(-) | 74440.57 | 45.01 | 9(+),10(-),11(+) |
| | | | 0.01 | 77611.13 | 85.06 | 10(-),11(+),12(-) | 77909.53 | 44.31 | 9(+),11(+),12(+) | 77293.37 | 159.75 | 9(-),10(-),12(-) | 77875.37 | 47.50 | 9(+),10(-),11(+) |
| | | | 0.1 | 78663.90 | 145.51 | 10(-),11(+),12(-) | 79019.50 | 19.01 | 9(+),11(+),12(+) | 78390.00 | 173.59 | 9(-),10(-),12(-) | 78998.17 | 40.33 | 9(+),10(-),11(+) |
| | | 50 | 0.001 | 69066.73 | 163.28 | 10(-),11(+),12(-) | 69671.93 | 61.49 | 9(+),11(+) | 68813.60 | 167.88 | 9(-),10(-),12(-) | 69673.10 | 32.07 | 9(+),11(+) |
| | | | 0.01 | 75892.57 | 96.68 | 10(-),12(-) | 76304.20 | 29.36 | 9(+),11(+),12(+) | 75883.23 | 129.05 | 10(-),12(-) | 76275.37 | 19.69 | 9(+),10(-),11(+) |
| | | | 0.1 | 78247.07 | 97.79 | 10(-),11(+),12(-) | 78507.90 | 4.27 | 9(+),11(+) | 78178.97 | 174.67 | 9(-),10(-),12(-) | 78506.53 | 5.20 | 9(+),11(+) |
| bou-s-c 2 | 162943 | 25 | 0.001 | 185031.00 | 256.80 | 10(-),11(+),12(-) | 186597.97 | 40.87 | 9(+),11(+),12(+) | 184857.03 | 229.71 | 9(-),10(-),12(-) | 186464.03 | 91.39 | 9(+),10(-),11(+) |
| | | | 0.01 | 189960.17 | 161.64 | 10(-),11(+),12(-) | 190779.23 | 41.14 | 9(+),11(+),12(+) | 189654.13 | 182.06 | 9(-),10(-),12(-) | 190732.63 | 58.43 | 9(+),10(-),11(+) |
| | | | 0.1 | 191284.70 | 173.66 | 10(-),11(+),12(-) | 192124.63 | 73.61 | 9(+),11(+),12(+) | 190988.60 | 243.98 | 9(-),10(-),12(-) | 192108.20 | 66.57 | 9(+),10(-),11(+) |
| | | 50 | 0.001 | 178525.63 | 357.61 | 10(-),11(-),12(-) | 180523.90 | 45.63 | 9(+),11(+) | 178875.80 | 248.68 | 9(+),10(-),12(-) | 180519.93 | 46.64 | 9(+),11(+) |
| | | | 0.01 | 187701.30 | 165.20 | 10(-),11(-),12(-) | 188848.53 | 51.18 | 9(+),11(+) | 187879.77 | 169.79 | 9(+),10(-),12(-) | 188852.90 | 41.61 | 9(+),11(+) |
| | | | 0.1 | 190806.63 | 160.38 | 10(-),11(+),12(-) | 191553.40 | 30.92 | 9(+),11(+) | 190794.07 | 155.19 | 9(-),10(-),12(-) | 191554.83 | 22.00 | 9(+),11(+) |
| uncorr 1 | 37686 | 25 | 0.001 | 81097.67 | 120.79 | 10(-),11(+),12(-) | 81438.43 | 29.77 | 9(+),11(+),12(-) | 79826.57 | 314.75 | 9(-),10(-),12(-) | 81457.00 | 0.00 | 9(+),10(-),11(+) |
| | | | 0.01 | 86056.53 | 59.24 | 10(-),11(+),12(-) | 86172.97 | 18.39 | 9(+),11(+),12(-) | 85932.27 | 120.74 | 9(-),10(-),12(-) | 86178.23 | 26.97 | 9(+),10(-),11(+) |
| | | | 0.1 | 87692.13 | 32.70 | 10(-),11(+),12(-) | 87718.70 | 11.17 | 9(+),11(+),12(-) | 87666.30 | 59.59 | 9(-),10(-),12(-) | 87725.37 | 8.55 | 9(+),10(-),11(+) |
| | | 50 | 0.001 | 74116.47 | 201.70 | 10(-),11(+),12(-) | 74763.10 | 45.28 | 9(+),11(+),12(+) | 71446.10 | 514.91 | 9(-),10(-),12(-) | 74759.37 | 10.02 | 9(+),10(+),11(+) |
| | | | 0.01 | 83765.50 | 102.91 | 10(-),11(+),12(-) | 83983.30 | 23.59 | 9(+),11(+) | 83404.57 | 514.91 | 9(-),10(-),12(-) | 83987.43 | 22.76 | 9(+),11(+) |
| | | | 0.1 | 86993.97 | 43.19 | 10(-),11(-),12(-) | 87057.27 | 17.19 | 9(+),11(+),12(-) | 87024.87 | 35.61 | 9(+),10(-),12(-) | 87062.53 | 5.07 | 9(+),10(+),11(+) |
| uncorr 2 | 93559 | 25 | 0.001 | 142887.10 | 161.18 | 10(-),11(+),12(-) | 143493.53 | 33.13 | 9(+),11(+),12(-) | 141780.10 | 281.11 | 9(-),10(-),12(-) | 143521.00 | 17.66 | 9(+),10(+),11(+) |
| | | | 0.01 | 147612.27 | 78.26 | 10(-),11(-),12(-) | 147799.43 | 16.74 | 9(+),11(+) | 147602.77 | 89.57 | 9(+),10(-),12(-) | 147802.23 | 13.51 | 9(+),11(+) |
| | | | 0.1 | 149125.07 | 33.53 | 10(-),11(+),12(-) | 149197.40 | 16.73 | 9(+),11(+) | 149102.27 | 45.91 | 9(-),10(-),12(-) | 149200.67 | 14.26 | 9(+),11(+) |
| | | 50 | 0.001 | 136249.50 | 232.14 | 10(-),11(+),12(-) | 137380.00 | 53.81 | 9(+),11(+),12(-) | 133715.83 | 452.04 | 9(-),10(-),12(-) | 137442.47 | 26.81 | 9(+),10(+),11(+) |
| | | | 0.01 | 144655.87 | 33.53 | 10(-),11(-),12(-) | 145808.87 | 18.50 | 9(+),11(+),12(-) | 145025.17 | 193.57 | 9(+),10(-),12(-) | 145822.27 | 19.88 | 9(+),10(+),11(+) |
| | | | 0.1 | 148484.37 | 45.83 | 10(-),11(-),12(-) | 148588.10 | 18.61 | 9(+),11(+),12(-) | 148517.47 | 38.88 | 9(+),10(-),12(-) | 148609.63 | 8.25 | 9(+),10(+),11(+) |

# Chapter 5

# Runtime Analysis for the Chance-Constrained Knapsack Problem with Correlated Uniform Weight

## 5.1 Introduction

In chapter 4, we have seen that evolutionary algorithms show efficiency when solving the chance-constrained knapsack problem (CCKP). In this chapter, we present the theoretical understanding of two bio-inspired algorithms: $\text{RLS}_2$ and (1+1) EA, for chance-constrained knapsack problem with correlated uniform weight, which can be expressed as a specific version of the formal CCKP.

While evolutionary algorithms for solving dynamic and stochastic combinatorial optimisation problems have been theoretically analysed in many articles (He, Mitavskiy, and Zhou, 2014; Lissovoi and Witt, 2017; Neumann, Pourhassan, and Roostapour, 2020; Roostapour et al., 2019), the literature on bio-indpired algorithms for chance-constraint optimisation problems is not that large. During recent years, chance-constrained optimisation problems have gained much attention due to their crucial role in situations where critical stochastic components are involved.

The first runtime analysis of evolutionary algorithms for CCKP is introduced by Neumann and Sutton (2019). They analysed different settings for the CCKP and argued that an optimal solution with a minimum probability of violating the chance constraint is more robust than other optimal solutions with the same fitness value. Firstly, they studied the cases in which all the profits are equal, but weights are chosen according to different distributions. They proved that, for instance, with uniform profit and iid weight, (1+1) EA found the optimal solution within the expected time $O(n \log n)$. Then they proved that (1+1) EA found the optimal solution for the instance with uniform profit and two variance class weights within the expected time $O(n^3)$. Secondly, they theoretically analyse the (1+1) EA on cases where profits and weight distribution differ between items. For the instances in which items are divided into two groups, items in the first group have a profit of 1, and weights are chosen uniformly at random from interval $[1/2, 3/2]$, items in the second group have a profit of 2, and weights are chosen uniformly at random from interval $[3/2, 5/2]$. They proved that the expected time of the (1+1) EA to find the optimal solution of this type of instance is $O(n^4)$. Moreover, they also pointed that a tiny change to the value of profit can result in local optimal that cannot be escaped in expected polynomial time for the (1+1) EA.

This chapter considers the CCKP with correlated uniform weight, this variant partitions the set of items into groups, and pairs of items within the same group have correlated weights. We prove bounds on both the time to find a feasible solution and the time to obtain the optimal solution. In this chapter, we argue that the optimal solution has both maximal profit and minimal probability of violating the chance constraint. In particular, we first prove that RLS$_2$ can find a feasible solution in time bounded by $O(n \log n)$ and by the (1+1) EA in time bounded by $O(n^2 \log n)$. Then, we investigate the optimisation time for these algorithms when the profit values are uniform, which has been studying in the deterministic constrained optimisation problems (Friedrich et al., 2020). However, the items in our case are divided into different groups, and it becomes necessary to take the number of chosen items from each group into account. Therefore, the optimisation time bound for RLS$_2$ becomes $O(n^3)$ and $O(n^3 \log n)$ for the (1+1) EA. After that, we consider the more general and complicated case in which profits may be arbitrary as long as each group has the same set of profit values. We show that an upper bound of $O(n^3)$ holds for RLS$_2$ and $O(n^3(\log n + \log p_{max}))$ holds for the (1+1) EA where $p_{max}$ denotes the maximal profit among all items.

The work of this chapter is based on a conference paper presented in GECCO 2021 (Xie et al., 2021).

The rest of the chapter is organized as follows. Section 5.2 presents the detailed definition of the considered CCKP with correlated uniform weights. In Section 5.3, we present the expected time of different algorithms needed to produce a feasible solution in Section 5.3.1. Moreover, we investigate the problem theoretically for uniform profits and arbitrary profits in Section 5.3.2 and 5.3.3, respectively. Our experimental analyses are presented in Section 5.4, followed by a conclusion in Section 5.5.

## 5.2 Preliminaries

In this section, we present the version of CCKP that is considered in this chapter and the details of the algorithms that we analyse.

### 5.2.1 Problem Definition

We consider a version of CCKP which is slightly different from the one introduced in Section 4. In this version, we are given $n$ items partitioned to $K$ groups and $m$ items in each group. Let $e_{ij}$ denote the $j$-th item in group $i$, with an associated stochastic weight $w_{ij}$, expected weight $a_{ij}$ and variance of weight $\sigma_{ij}^2 = d$, and each item has profit $p_{ij}$. We assume that the weights of items in different groups are independent to each other, but the weights of items in a same group are correlated with each other via a shared covariance $c > 0$, i.e. $cov(e_{ij}, e_{kl}) = c$ iff $i = k$, and $cov(e_{ij}, e_{kl}) = 0$ iff $i \neq k$.

The chance-constrained knapsack problem with correlated uniform weights can be formulated as follows:

$$\max \quad P(x) = \sum_{i=1}^{K}\sum_{j=1}^{m} p_{ij}x_{ij} \tag{5.1}$$

$$\text{s.t.} \quad P_r(W(x) > C) \leq \alpha. \tag{5.2}$$

The objective of this problem is to select a set of items that maximises profit subject to the chance constraint, which requires that the solution violates the constraint bound $C$ only with probability at most $\alpha$.

A solution is characterized as a vector of binary decision variables $x = (x_{11}, x_{12}, \dots, x_{1m}, \dots, x_{Km}) \in \{0,1\}^n$. When $x_{ij} = 1$, the $j$-th item of the $i$-th group is selected. The weight of a solution $x$ is the random variable

$$W(x) = \sum_{i=1}^{K}\sum_{j=1}^{m} w_{ij}x_{ij}, \tag{5.3}$$

with expectation

$$E[W(x)] = \sum_{i=1}^{K}\sum_{j=1}^{m} x_{ij}a_{ij}, \tag{5.4}$$

and variance

$$Var(x) = d\sum_{i=1}^{K}\sum_{j=1}^{m} x_{ij} + 2c\sum_{i=1}^{K}\sum_{1\leq j_1<j_2\leq m}(x_{ij_1}x_{ij_2}). \tag{5.5}$$

We define two specific types of solutions for any instance among the same number of selected items. Solutions that match the type *balanced solutions* shall have the minimal covariance value, and solutions that match the type *most unbalance solution* have the maximal covariance value.

**Definition 5.2.1.** Among all solutions with exactly $\ell$ items, we call a search point $x; |x|_1 = \ell$ a **balanced solution**, denoted by $\ell^b$ if it selects $\lfloor \frac{\ell}{K} \rfloor$ items from $K - \left(\ell - \lfloor \frac{\ell}{K} \rfloor \cdot K\right)$ groups and $\lfloor \frac{\ell}{K} \rfloor + 1$ items from the remaining $\ell - \lfloor \frac{\ell}{K} \rfloor \cdot K$ groups. This solution has covariance

$$s_\ell^b = c\left\{\left[K - \left(\ell - \lfloor \tfrac{\ell}{K} \rfloor\right)\right] \lfloor \tfrac{\ell}{K} \rfloor \left(\lfloor \tfrac{\ell}{K} \rfloor - 1\right) + \left(\ell - \lfloor \tfrac{\ell}{K} \rfloor\right)\left(\lfloor \tfrac{\ell}{K} \rfloor + 1\right)\lfloor \tfrac{\ell}{K} \rfloor\right\}.$$

Solutions with exactly $\ell$ bits that are not balanced solutions are called **unbalanced solutions**.

Among all unbalanced solutions, we call the following one the **most unbalanced solution** denoted by $\ell^{ub}$, which selects exactly $m$ items from $\lfloor \frac{\ell}{m} \rfloor$ groups and $\left(\ell - \lfloor \frac{\ell}{m} \rfloor \cdot m\right)$ items from another group. Since $m$ is the maximal number of items in each group, in the most unbalanced solution, there are $\lfloor \frac{\ell}{m} \rfloor$ full groups and one other group containing the remaining items. This solution has covariance

$$s_\ell^{ub} = c\left[\left\lfloor \frac{\ell}{m} \right\rfloor m(m-1) + \left(\ell - \left\lfloor \frac{\ell}{m} \right\rfloor m\right)\left(i - \left\lfloor \frac{\ell}{m} \right\rfloor m - 1\right)\right].$$

We calculate the upper bound of the covariance of acceptable solutions according to the one-side Chebyshev's inequality (3.9) for all solutions with $\ell$ itmes. The covariance of the solution $X$ is denoted by

$$s_\ell = 2c \sum_{i=1}^{K} \sum_{1 \leq j_1 < j_2 \leq m} (x_{ij_1} x_{ij_2}),$$

and $\ell$ denotes the number of one bits in solutions. By Equation (5.5), $Var[W(X)] = \ell d + s_\ell$. Therefore, by assuming the expected weights are all equal and denoted by $a$, the bound according to Chebyshev's inequality yields

$$\begin{aligned} &\frac{\ell d + s_\ell}{\ell d + s_\ell + (C - a\ell)^2} \leq \alpha \\ \Longleftrightarrow &\ell d + s_\ell \leq \alpha(\ell d + s_\ell + (C - a\ell)^2) \\ \Longleftrightarrow &s_\ell \leq \frac{(C - a\ell)^2 \alpha}{1 - \alpha} - \ell d. \end{aligned} \tag{5.6}$$

Therefore, the covariance of feasible solutions with exactly $\ell$ items is bounded above by $\frac{(C - a\ell)^2 \alpha}{1 - \alpha} - \ell d$.

In this chapter, we assume the weights of items are correlated uniformly and that it is intractable to calculate the exact probability of violating the chance constraint. Similar to the work in Chapter 4, we use the one-sided Chebyshev's inequality to construct a usable surrogate of the chance constraint (5.2).

In particular, we define a surrogate function $\beta$ over decision vectors as

$$\beta(x) = \frac{Var(x)}{Var(x) + (C - E[W(x)])^2}. \tag{5.7}$$

It is clear that $P_r(W(x) \geq C) \leq \beta(x)$, and therefore every $x$ such that $\beta(x) \leq \alpha$ is also feasible.

### 5.2.2 Algorithms

We analyse the runtime of two algorithms. The first one, described in Algorithm 4, named RLS$_2$, which can flip a bit or two bits at once. Another algorithm is (1+1) EA (Algorithm 6). This algorithm, as introduced in Section 3.3.2 flips each bit of the current solution with the probability of $1/n$ in each mutation step. Both algorithms replace the current solution with the generated offspring if it is at least as good as its parents.

Since RLS$_2$ and (1+1) EA are single-objective algorithms, the comparisons between solutions in these algorithms are based on the fitness function

$$f(x) := (P'(x), \beta'(x)), \tag{5.8}$$

where $P'(x) = -1$ iff $\beta'(x) > \alpha$ and $P'(x) = P(x)$ otherwise, $\beta'(x) = \beta(x)$ iff $E[W(x)] < C$ and $\beta'(x) = 1 + E[W(x)] - C$ otherwise. We optimize $f$ in lexicographic order where the goal is to maximize $P'(x)$ and minimize $\beta(x)$, i.e. we have

$$\begin{aligned} & f(x) \succeq f(y) \\ \iff \quad & P'(x) > P'(y) \\ & \text{or} \quad \left(P'(x) = P'(y) \wedge \beta(x) \leq \beta(y)\right). \end{aligned} \tag{5.9}$$

Since selection is monotone, once a feasible solution is located, neither algorithm will subsequently accept an infeasible solution. Therefore, the algorithm can separate the process of finding an optimal solution into two parts, and the algorithm may first need to find a feasible solution in the first part. In the second part, it must find the highest profit among all feasible solutions.

## 5.3 Theoretical Analysis

In this section, we theoretically investigate the performance of the RLS$_2$ and (1+1) EA on different versions of the CCKP with correlated uniform weights by using the running time analysis. We analyse the behaviour of algorithms when obtaining feasible solutions and assume all items have the same expected weights. Moreover, we pay special attention to the case in which all the profits are equal. After that, we consider cases in where profits are arbitrary and mirrored by each group.

### 5.3.1 Obtaining feasible solutions

In this section, we analyze the expected time for RLS$_2$ and (1+1) EA to find feasible solutions.

**Lemma 5.3.1.** *Starting with an arbitrary initial solution, the expected time until RLS$_2$ obtains a feasible solution is $O(n \log n)$.*

*Proof.* Adding a new item to the selected set will increase both the total expected weight and the probability of violating the chance constraint. Since all items have the same expected weight $a$, the sum of expected weight can be represented by the number of ones in the solution.

The fitness function is defined so that the total expected weight of a solution will never increase as long as no feasible solution has been obtained. It implies that the RLS$_2$ never accepts mutations that increase the number of ones and only accept a decrease in the number of ones. RLS$_2$ cannot accept any single bit flips that flip a one to zero or 2-bit flips that flip two one-bits to zeros.

Therefore, at any solution $x; |x|_1 = \ell$, there are $\ell$ itmes to decrease, and the probability of decreasing the number of ones is at least $\frac{\ell}{2n}$. Hence, the expected waiting time until RLS$_2$ constructs a feasible solution is bounded above by $2n\left(1 + \cdots + \frac{1}{n}\right) = O(n \log n)$. □

**Lemma 5.3.2.** *Starting with an arbitrary initial solution, the expected time until the (1+1) EA obtains a feasible solution is $O(n^2 \log n)$.*

*Proof.* According to the definition of the fitness function (5.8), before finding a solution with an expected weight less than $C$, the (1+1) EA never accepts a solution that increases the number of items. Therefore, before producing such a solution, the algorithm only takes mutations that reduce the number of items and thus behave identically to optimise the classical OneMax problem. The expected time for the (1+1) EA to find a solution $x$ with $E[W(x)] < C$ is thus bounded by $O(n \log n)$, i.e., it's expected running time on OneMax (Mühlenbein, 1992).

According to Chebyshev's inequality, after finding a solution with an expected weight less than $C$, the (1+1) EA always accepts a solution with a more negligible constraint-violation probability. We construct a potential function $h : \{0,1\}^n$ as the sum of the variance and covariance of a solution,

$$h(x) = d\ell + 2c \sum_{i=1}^{K} \sum_{1 \le j_1 < j_2 \le m} (x_{ij_i} x_{ij_2}), \tag{5.10}$$

where $\ell$ denotes the number of items selected by solution $x$, $|x|_1 = \ell$ and $E[W(x)] < C$.

For a solution $x$ with $\ell$ one bits, the (1+1) EA can reduce the potential $h(x)$ and the violation probability when flipping any one of the $\ell$ one bits to zero. Let the solution $y; |y| = \ell - 1$ be an offspring generated from $x$ by flipping a one bit to zero. Then, we have $h(y) < h(x)$ and $\beta(y) < \beta(x)$. Let $x'$ be the next possible acceptable solution for the (1+1) EA with $\ell$ one bits. Then solution $x'$ should be better than solution $y$ according to the fitness function. We have $\beta(x') \le \beta(x) < \beta(x)$ and $h(x') \le h(y) < h(x)$ according to the Chebyshev's inequality.

Given solution $x$, we consider all steps that flipping a 1-bit the algorithm generates solution $y$ after finding the solution $x'$ which reduces the value of the potential function. Let $\{r_1, \ldots, r_K\}$, where $0 \le r_i \le m$ denotes the number of items in group $i$ selected by $x$. Assume $x'$ is generated from $x$ by flipping a one bit from group $i$ to zero. Then, the reduction in potential for this mutation is the sum of the variance of this item and the difference of the covariance between $r_i$ elements and $r_i - 1$ elements. That is,

$$d + c(r_i(r_i - 1) - (r_i - 1)(r_i - 2)) = d + 2c(r_i - 1).$$

In group $i$, there are $r_i$ single bit flips that achieve this reduction, so the total contribution for group $i$ is

$$dr_i + 2cr_i(r_i - 1) \ge dr_i + 2c\frac{r_i(r_i - 1)}{2}.$$

Summing over all groups we have

$$\sum_{i=1}^{K} dr_i + 2cr_i(r_i - 1) \ge h(x). \tag{5.11}$$

Therefore, after producing all single bit flips where each one bit of $\ell$ bits in $x$ has been flipped to zero once, the sum of gains concerning the potential function should be as least as large as $h(x)$.

For all $t \in \mathbb{N}$, let $x_t$ be the search point of the (1+1) EA for the problem at time $t$ and $X_t = h(x_t)$. Then

$$X_t - X_{t+1} = h(x_t) - h(x_{t+1}).$$

Let $x \in \{0,1\}^n$ be a fixed nonempty solution, and let the points $y_1, \ldots, y_\ell$ be the $\ell$ different search points in $\{0,1\}^n$ generated from $x$ by first flipping one of the different $\ell$ one bit to zero. Thus, we have by $h(y_i) \le h(y)$ for all $i \in \{1, \ldots, \ell\}$ and inequality

(5.11) that

$$\sum_{i=1}^{\ell} (h(x) - h(y_i)) \geq h(x). \tag{5.12}$$

Since each $y_i$ is generated from $x$ by a single bit flip, we have

$$\Pr(x_{t+1} = y_i \mid x_t = x) = \left(\left(1 - \frac{1}{n}\right)^{(n-1)} \left(\frac{1}{n}\right)\right) \geq \frac{1}{en} \tag{5.13}$$

for all $i \in \{1, \ldots, \ell\}$. Furthermore

$$E[X_t - X_{t+1} \mid x_t = x, x_{t+1} = y_i] = h(x) - h(y_i) \tag{5.14}$$

holds for all $i \in \{1, \ldots, \ell\}$.

The (1+1) EA never increases the current $h$-value of a search point, that is, $X_t - X_{t+1}$ is non-negative. Therefore, we have

$$E[X_t - X_{t+1} \mid x_t = x] \geq \sum_{i=1}^{\ell} (h(x) - h(y_i)) \frac{1}{en} \tag{5.15}$$

and therefore, we have for all $X \in \{0,1\}^n$ that

$$E[X_t - X_{t+1} \mid x_t = x] \geq \frac{h(x)}{en} = \frac{X_t}{en}.$$

Therefore, the drift on $X_t$ is at least $\frac{h(x)}{en}$, and since the algorithm starts with $h(x) \leq h_\ell^{ub}$ where $h_\ell^{ub}$ denotes the sum of variance and covariance of the most unbalanced solution from the same level of $x$, and the minimum value of $h$ before reaching $h = 0$ is 1, by multiplicative drift analysis (Theorem 3.4.6), we find the expected time of at most

$$\frac{1 + \log(h_\ell^{ub})}{\frac{1}{en}} = O(n \log n) \tag{5.16}$$

to reach a solution with the number of one bit less than the starting search point. Let $\ell$ denote the number of one bits for the starting point, and the probability value of this solution is better than the best probability value for any solution with $\ell$ ones.

Furthermore, there is at most $n$ levels in the search space, and it takes $O(n \log n)$ steps for the (1+1) EA to produce all possible solutions in each level. Altogether, the expected time of (1+1) EA to find a feasible solution is at most $O(n^2 \log n)$. $\square$

In the following sections, we assume the algorithms have produced a feasible solution and analyse the expected time that the RLS$_2$ and (1+1) EA require to find the optimal solutions for two different settings of the CCKP with correlated uniform weight.

### 5.3.2 Uniform Profits

We consider case that the deterministic profits are uniform. Since the actual value of profits does not affect the analysis, it is convenient to use unit profits. We calculate the performance of algorithms for this type of problem.

**Instance 5.3.3.** *Given $K$ groups, each group has $m$ items. There are $n = K \cdot m$ items in total, the capacity of the knapsack is bounded by $C$. For $1 \leq i \leq K$, $1 \leq j \leq m$, let $p_{ij} = 1$, $a_{ij} = a$, $\sigma_{ij}^2 = d$, where $d > 0$ is a constant. The covariance of items within any group is $c$, i.e., we have $cov(e_{ij}, e_{kl}) = c$ iff $i = k$ and $cov(e_{ij}, e_{kl}) = 0$ otherwise.*

**Definition 5.3.1.** Let $r = \max\{|x|_1 \mid x \in \{0,1\}^n \text{ with } \beta(x) \leq \alpha\}$ and partition the feasible search space by $L_0, L_1, \ldots, L_r$ such that

$$L_i = \{x \in \{0,1\}^n \mid |x|_1 = i \ \text{ with } \beta(x) \leq \alpha\}. \tag{5.17}$$

We further bi-partition each partition $L_i$ into two sets $S_{i\gamma}$ and $S_{i\zeta}$ such that $S_{i\gamma} \cup S_{i\zeta} = L_i$ and $S_{i\gamma} \cap S_{i\zeta} = \emptyset$ as follows.

The set $S_{i\gamma} \subseteq L_i$ contains all feasible solutions where no extra item can be added without violating the chance constraint and $S_{i\zeta} \subseteq L_i$ is the set containing all feasible solutions where at least one extra item can be added to obtain a feasible solution with at least $i + 1$ ones.

**Lemma 5.3.4.** *Starting with an arbitrary initial solution, the expected optimization time of RLS$_2$ on the chance-constrained knapsack problem with correlated uniform weight is $O(n^3)$.*

*Proof.* Due to Lemma 5.3.1, RLS$_2$ finds a feasible solution in expected time $O(n \log n)$. Also, since all feasible solutions dominate any infeasible solution, the algorithm does not return to the infeasible region.

Let $x \in L_\ell$. If $x \in S_{\ell\zeta}$, then there is at least one additional item that can be feasibly selected in the mutation process. This selection occurs with a probability of $1/2n$. Otherwise, only a 2-bit flip changing a zero to one and a one to zero is accepted if it reduces the covariance of the solution without changing the profit until the algorithm produces a balanced solution on the same level.

According to Definition 5.2.1, the balanced solution in each level has the smallest covariance and number of items selected from each group. Let $l_i(x)$, $1 \leq i \leq K$ be the number of elements chosen by $x$ from group $i$. Assume, without loss of generality, that the groups are sorted in increasing order with respect to the $l_i(x)$. Furthermore, let

$$\begin{aligned} s(x) = &\sum_{i=1}^{K-(\ell-\lfloor \frac{\ell}{K} \rfloor \cdot K)} \max\left\{0, \left\lfloor \frac{\ell}{K} \right\rfloor - l_i(x)\right\} \\ &+ \sum_{K-(\ell-\lfloor \frac{\ell}{K} \rfloor \cdot K)+1}^{K} \max\left\{0, \left\lfloor \frac{\ell}{K} \right\rfloor + 1 - l_i(x)\right\} \end{aligned} \tag{5.18}$$

be the number of items that belong to an arbitrary balanced solution, but not chosen by $x$, and let

$$\begin{aligned} t(x) = &\sum_{i=K-(\ell-\lfloor \frac{\ell}{K} \rfloor \cdot K)+1}^{K} \max\left\{0, l_i(x) - \left(\left\lfloor \frac{\ell}{K} \right\rfloor + 1\right)\right\} \\ &+ \sum_{i=1}^{K-(\ell-\lfloor \frac{\ell}{K} \rfloor \cdot K)} \max\left\{0, l_i(x) - \left\lfloor \frac{\ell}{K} \right\rfloor\right\} \end{aligned} \tag{5.19}$$

be the number of items chosen by $x$, but do not belong to a balanced solution. Note that $s(x)$ should be equal to $t(x)$ for any feasible solution in $L_\ell$. Let $g = s(x) = t(x)$.

As there are exactly $\ell$ 1-bit in solution $x$, and $s(x)$ is a fixed value, this implies that there are at least $g$ 1-bits which can be swapped with an arbitrary 0-bit of the missing $g$ elements to reduce the covariance of $x$. Hence, the probability of such swapping is at least $g^2/2n^2$. Since $g$ cannot increase and $g \leq \ell$, it suffices to sum up these expected waiting times, and the expected time until reaching $g = 0$ is

$$\sum_{g=1}^{\ell}(2n^2/g^2) = O\left(n^2(1-1/\ell)\right).$$

There are at most $n$ levels of $L_\ell$ which implies that the expected time until an optimal solution has been achieved is

$$\sum_{\ell=1}^{n}(n^2 - n^2/\ell) = O(n^3 - n^2\log n) = O(n^3),$$

which completes the proof. □

**Lemma 5.3.5.** *Let $x \in S_{\ell\gamma}$, then there exists some $q \in \{1, \dots, n-1\}$ different accepted 2-bit flips, and each of those $q$ 2-bit flip can reduce the covariance of the solution. The expected one-step change of the (1+1) EA is $\frac{X_t}{en^2}$.*

*Proof.* Let solution $x \in S_{\ell\gamma}, |x|_1 = \ell$ and let $s_x$ denote the covariance of $x$. Then according to the inequality (5.6), $s_x$ is bounded by $\frac{(C-a\ell)^2\alpha}{1-\alpha} - d\ell$. Let $x' \in S_{i\zeta}, |x'|_1 = \ell$ be the balanced solution which takes $\left\lceil\frac{\ell}{K}\right\rceil$ elements from the first $\ell - \left\lfloor\frac{\ell}{K}\right\rfloor K$ groups and takes $\left\lfloor\frac{\ell}{K}\right\rfloor$ elements from the last $K - \ell - \left\lfloor\frac{\ell}{K}\right\rfloor K$ groups.

Now, we have the reduction of covariance that flip $i \in I$ to zero denoted by $2c(r_i - 1)$, where $r_i$ is the number of selected items of the group that $i$ belong to. There are $r_i - \left\lceil\frac{\ell}{K}\right\rceil$ one bits need to be flipped in this group that achieve this reduction to attend balance, and $r_i > \left\lceil\frac{\ell}{K}\right\rceil$, so the total contribution for this group is

$$2c(r_i-1)\left(r_i - \left\lceil\frac{\ell}{K}\right\rceil\right) \geq c(r_i-1)r_i - c\left(\left\lceil\frac{\ell}{K}\right\rceil - 1\right)\left\lceil\frac{\ell}{K}\right\rceil. \tag{5.20}$$

Similarly, we have the increase of covariance when flip a bit $j \in J$ denoted by $2cr_j$, where $r_j$ is the number of selected items of $x$ in the group that $j$ belong to. There are $\left\lfloor\frac{\ell}{K}\right\rfloor - r'_j$ zero bits flip in this group to attend balance and the total contribution for group $k'$ is

$$2c(r'_j)\left(\left\lfloor\frac{\ell}{K}\right\rfloor - r'_j\right) \leq c\left(\left\lfloor\frac{\ell}{K}\right\rfloor - 1\right)\left\lfloor\frac{\ell}{K}\right\rfloor - c(r'_j-1)r'_j. \tag{5.21}$$

Define the total reduction of covariance from $x$ to $x'$ by inequalities (5.20) and (5.21) as

$$\begin{aligned}&\sum_{i=1}^{q} 2c(r_i-1) - \sum_{j=1}^{q} 2cr'_j\\ \geq& c(r_i-1)r_i - c\left(\left\lfloor \frac{\ell}{K}\right\rfloor - 1\right)\left\lfloor \frac{\ell}{K}\right\rfloor - \left(c\left(\left\lfloor \frac{\ell}{K}\right\rfloor - 1\right)\left\lfloor \frac{\ell}{K}\right\rfloor - c(r'_j-1)r'_j\right)\\ =& s_x - s^b_\ell. \end{aligned} \tag{5.22}$$

Therefore performing all $q$ 2-bit flips simultaneously changes $x$ into $x'$ and leads to a covariance decrease at least as large as $s_x - s^b_\ell$, where $s^b_\ell$ denotes the covariance of the balanced solution with exactly $i$ items.

For all $t \in \mathbb{N}$, let $X_t \in L_i$ be a fixed, non-empty solution generated at time $t$ by the (1+1) EA, and let $X_t = s_{X_t} - s^b_i$. Then

$$X_t - X_{t+1} = s_{X_t} - s_{X_{t+1}}. \tag{5.23}$$

Let $Y = \{y_1, \ldots, y_q\}$ with $q \in \{1, \ldots, n\}$ be the set of $q$ different search points that on the same level of $x$ in the search space generated from $x$ by one of the $q$ acceptable different 2-bit flips. We have $s_{y_i} \leq s_x$ for all $i \in \{1, \ldots, q\}$ and

$$\sum_{i=1}^{q} (s_x - s_{y_i}) \geq s_x - s^b_i. \tag{5.24}$$

Since each $y_i$ is generated from $x$ by one of the $q$ 2-bit flips,

$$\Pr[X_{t+1} \in Y | X_t = x] = q\left(1 - \frac{1}{n}\right)^{n-2}\left(\frac{1}{n}\right)^2 \geq \frac{q}{en^2} \tag{5.25}$$

of the (1+1) EA. Furthermore,

$$E[X_t - X_{t+1} | X_t = x, X_{t+1} \in Y] = \frac{s_x - s^b_i}{q} = \frac{X_t}{q}. \tag{5.26}$$

The algorithm cannot accept an offspring on the same level that increases the covariance. That is, $X_t - X_{t+1}$ is non-negative. Thus, we have by (5.25) and (5.26) that

$$E[X_t - X_{t+1} | X_t = x] \geq \frac{X_t}{en^2}. \tag{5.27}$$

□

**Lemma 5.3.6.** *The expected time for the (1+1) EA to transform a solution in $S_{i\gamma}$ to a solution in $S_{i\zeta} \cup L_j$ where $j > i$ is bounded by $O(n^2 \log n)$.*

*Proof.* According to Lemma 5.3.5, the drift on $X_t$ is at least $\frac{X_t}{en^2}$ for the (1+1) EA. Therefore, since both algorithms start with $X_t \leq s^i = O(n^2)$ and the minimum value of $X_t$ before reaching $X_t = 0$ is 1, by multiplicative drift analysis, the expected time is at most $O(n^2 \log n)$ to reach a solution in $S_{i\zeta}$. Then, if $i < r$, it is possible for the (1+1) EA to generate a feasible in $L_{i+1}$ with probability $1/en$. The total expected time of the (1+1) EA until an solution in $S_{i\zeta} \cup L_j$ is generated is thus bounded by $O(n^2 \log n)$.

□

**Theorem 5.3.7.** *The expected time until the (1+1) EA working on the fitness function (5.8) constructs the optimal solution to Instance 5.3.3 is bounded by $O(n^3 \log n)$.*

*Proof.* By Lemmas 5.3.2 and 5.3.6, for all $i < r$, it is sufficient to investigate the search process after having found a feasible solution $x \in S_{i\zeta}$, and after that, the algorithms can only accept an offspring with a more significant number of one bit. It is possible for the (1+1) EA to generate a feasible solution in $L_{i+1}$ by mutating exactly one zero bit to one. This event occurs with probability at least $\frac{1}{en}$ for the (1+1) EA.

Therefore, it will take $O(n^2 \log n + en)$ steps to produce a feasible solution in level $L_{r+1}$ when started from a random feasible solution in $L_r$. Altogether, the expected optimization time is bounded by

$$O(n^2 \log n) + \sum_{i=0}^{r-1} (n^2 \log n + en) = O(n^3 \log n), \tag{5.28}$$

where $r < n$. □

## 5.3.3 Arbitrary profits mirrored by each group

We now turn our attention to the more complicated case where a single group has arbitrary profits, but this set of profits is the same for each of the $K$ groups. This case resembles the case of general linear functions, but the chosen function is shared by all groups.

**Instance 5.3.8.** *Given $K$ groups, each group has $m$ items. There are $n = K \cdot m$ items in total, the capacity of knapsack is bounded by $C$. For $1 \le i \le K$, $1 \le j \le m$, let $a_{ij} = a$, $\sigma_{ij}^2 = d$ are constants, and let $p_{i1} \ge p_{i2} \ge \ldots \ge p_{im}$ for $i \in \{1, \ldots, K\}$ and $p_{i\ell} = p_{k\ell}$ for each $i, k \in \{1, \ldots, K\}$, $1 \le \ell \le m$. The covariance of items in groups is $c$, i.e. we have $cov(e_{ij}, e_{kl}) = c$ iff $i = k$ and $cov(e_{ij}, e_{kl}) = 0$ otherwise.*

**Theorem 5.3.9.** *Starting with an arbitrary initial solution, the expected optimization time of RLS$_2$ on the chance-constrained knapsack problem with correlated uniform weights is $O(n^3)$ on Instance 5.3.8.*

*Proof.* By Lemma 5.3.1, RLS$_2$ finds a feasible solution in expected time $O(n \log n)$. Also, since all feasible solutions dominate infeasible solutions, the algorithm does not switch back to the infeasible region again. By the definition of Instance 5.3.8 that items in a group have different profit and the same weight, it is possible to have more than one balanced solution in each level of this case, but only one balanced solution with maximum profit, where we ignore the order of groups. We order all items regarding to their profit as $p_{11} = p_{21} = \ldots = p_{K1} \ge p_{12} = p_{22} = \ldots = p_{K2} \ge \ldots \ge p_{1m} = p_{2m} = \ldots = p_{Km}$.

For a given solution $x$, we call the multi-set $P(x) = \{p_i \mid x_i = 1\}$ the *profit profile* of $x$, i.e., the multi-set of profit values selected by $x$. We say that a profit profile $P$ is contained in $P(x)$ if $P \subseteq P(x)$. Let $x$ be a feasible solution whose profit profile contains $P_j = \{p_1, \ldots, p_j\}$ (but which does not contain $P_{j+1}$). We claim that RLS$_2$ does not accept a solution whose profit profile does not contain $P_j$. An operation flipping a single 1-bit that flips a 1 to 0 is not accepted, as it reduces the profit and therefore cannot lead to a solution not containing $P_j$. A 2-bit flip is only accepted if it does not decrease the profit, and thus also cannot create a solution whose profit profile does not contain $P_j$, as $P_j$ includes the $j$-largest profits of the given input.

We analyze the time to transform a solution $x$ containing profit profile $P_j = \{p_i \mid 1 \leq i \leq j\}$ into a solution $x'$ containing profit profile $P_{j+1}$. Consider the profit $p_{j+1}$ in the group with the smallest number of elements whose bit $x_i$ is set to 0. Flipping $x_i$ adds the profit $p_{j+1}$ to the profile $P_j$. Assume that bit $x_i$ belongs to group $r \in \{1, \ldots, K\}$, i.e., $x_i = x_{rs}$. If there is another item selected in group $r$ (selected by $x_{rs'} = 1$) whose profit is less than $p_j$, then flipping both $x_{rs}$ and $x_{rs'}$ leads to an accepted solution $x'$ with $P_{j+1} \subseteq P(x')$. This happens with probability $1/2n^2$. Assume now that there is no such item in group $r$. Then $p_{j+1}$ is the largest non selected profit in group $r$.

Let $S$ be the set of groups with the largest number of items selected and $\hat{p}$ the smallest selected profit from these groups. Assume that $x_i$ is not in one of the groups in $S$. Then flipping $x_i$ to 1 and setting the bit corresponding to $\hat{p}$ to 0 is accepted and leads to a solution containing profit profile $P_{j+1}$. If $x_i$ is from one of the groups in $S$, then there should has another item selected in $S$ with a profit smaller than $p_{j+1}$ or the solution $x$ is already optimal.

Altogether, to produce a solution $x'$ containing $P_{j+1}$ from a solution with $P_j$, RLS$_2$ needs at most $O(n^2)$ steps, and since there are at most $n$ items in any solution, the expected optimization time of RLS$_2$ is $O(n^3)$. □

Let $p_{\max} = p_{i1}, i \in \{1, \ldots K\}$ be the maximal profit of the given input.

**Theorem 5.3.10.** *Starting with an arbitrary initial solution, the expected optimization time of the (1+1) EA on the chance-constrained knapsack problem with correlated uniform weight is $O(n^3(\log n + \log p_{\max}))$ on Instance 5.3.8.*

*Proof.* According to Lemma 5.3.2, the expected time to reach a feasible solution is $O(n^2 \log n)$. Therefore it is sufficient to start the analysis with a feasible solution, after which the (1+1) EA will never sample an infeasible solution during the remainder of the optimization process.

For our analysis, we consider the set of all solutions $L_j = \{x \mid |x|_1 = j; \beta(x) \leq \alpha\}$ containing exactly $j$ 1-bits. For each $j$ we show that the expected number of offspring created from an individual in $L_j$ is $O(n^2(\log n + \log p_{\max}))$. After this many iterations, either the optimal solution (contained in $L_j$) has been created, or a feasible solution $y$ with $p(y) > \max_{X \in L_j} p(x)$ has been produced, which implies that the algorithm will not accept any solution in $L_j$ afterwards.

We now show that the expected number of offspring created from solutions in $L_j$ is $O(n^2(\log n + \log p_{\max}))$. Let $x \in L_j$ be the current solution, and let $x^{j,\max} = \arg\max_{x \in L_j} p(x)$ be an arbitrary feasible solution in $L_j$ with the largest possible profit. Denote the *loss* by

$$l(x) = \sum_{i=1}^{n} p_i X_i^{j,\max}(1 - x_i),$$

that is, the sum of the profits chosen by $x^{j,\max}$ but not chosen by $x$. Denote the *surplus* by

$$s(x) = \sum_{i=1}^{n} p_i(1 - x_i^{j,\max})x_i,$$

that is, the sum of the profits chosen by $x$ but not chosen by $x^{\max,j}$. Define the total increase in profit from $x$ to $x^{j,\max}$ as

$$g(x) = p(x^{j,\max}) - p(x) = l(x) - s(x).$$

Let $r = \sum_{i=1}^{n} x_i^{j,\max}(1 - x_i)$ be the number of indices set to 1 by $x^{j,\max}$ and 0 by $x$. We give a set of $r$ accepted 2-bit flips where the sum of the increases in profit is $g(x)$.

We consider the $K$ groups and w.l.o.g. assume that they are sorted in increasing order with respect to the number of elements chosen by $x = (x_{11}, x_{12}, \ldots, x_{1m}, \ldots, x_{Km})$. Let $\ell_i(x)$, $1 \leq i \leq K$ be the number of elements chosen by $x$ in group $i$. We have $\ell_1(x) \leq \ldots \leq \ell_K(x)$. We consider the solution $\hat{x}^{j,\max}$ of maximal profit in $L_j$ for which $\ell_1(\hat{x}^{j,\max}) \leq \ldots \leq \ell_K(\hat{x}^{j,\max})$ and $\ell_K(\hat{x}^{j,\max}) \leq \ell_1(\hat{x}^{j,\max}) + 1$ holds. This implies that $\hat{x}^{j,\max}$ is a balanced solution having the smallest variance in $L_j$. Note that such a solution exists as we may reorder the groups as each group contains the same (multi-)set of profits.

We have

$$\sum_{i=1}^{k} \ell_i(x) \leq \sum_{i=1}^{k} \ell_i(\hat{x}^{j,\max}), 1 \leq k \leq K \tag{5.29}$$

as both solutions contain $j$ elements and the groups are sorted in increasing order of the number of elements chosen by $x$.

This implies

$$\sum_{i=1}^{k} \ell_i(x - \hat{x}^{j,\max}) \leq \sum_{i=1}^{k} \ell_i(\hat{x}^{j,\max} - x), 1 \leq k \leq K \tag{5.30}$$

as the intersection of $\hat{x}^{j,\max}$ and $X$ contributes the same to each summand. Here $x - y = \max\{x - y, 0\}$ denotes the set different of the elements chosen by $x$ but not by $y$.

Therefore, the number of elements chosen by $\hat{x}^{j,\max}$ but not by $x$ is greater than or equal to the number of elements chosen by $x$ but not $\hat{x}^{j,\max}$ for each of the first $k$ groups. We then define our set of $r$ 2-bit flips. The $i$th 2-bit flip flips the $i$th 0 bit of $X$ (in the order given by the bit string $x = (x_{11}, x_{12}, \ldots, x_{1m}, \ldots, x_{Km})$ set to 1 in $\hat{x}^{j,\max} - x$ and the $i$th 1 bit in $x - \hat{x}^{j,\max}$.

Consider operation $i$ and let $p'_i$ be the profit introduced and $p''_i$ be the profit to be removed. As per construction we have $\sum_{i=1}^{r} p'_i = l(x)$ and $\sum_{i=1}^{r} p''_i = s(x)$ which implies that the total gain of the set of $r$ 2-bit flips is $g(x) = l(x) - s(x)$. It remains to show that each of these $r$ 2-bit flips is accepted by the algorithm. Consider the $i$th operation. We show that $\beta(x)$ does not increase. Let $r'$ be the group that $p'_i$ belongs to and $r''$ be the group that $p''_i$ belongs to. We have $r' \leq r''$ due to Equation (5.30) and therefore $\ell_{r'}(x) \leq \ell_{r''}(x)$. This implies that the 2-bit flip leads to a solution $y$ with $\beta(Y) \leq \beta(x)$. We also have $p'_i \geq p''_i$ as otherwise we could improve the profit of $\hat{x}^{j,\max}$ which contradicts that $\hat{x}^{j,\max}$ is a feasible solution of maximal profit in $L_j$.

Given the set of $r$ accepted 2-bit flips, the expected increase in profit is at least

$$r/(en^2) \cdot g(x)/r = g(x)/(en^2),$$

as the probability of the (1+1) EA to produce such a 2-bit flip is

$$r \cdot \frac{1}{n} \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-2} = \frac{r}{n^2}\left(1 - \frac{1}{n}\right)^{n-2} = O(\frac{r}{en^2}),$$

and the average gain of this flip is $g(x)/r$.

For any non-maximal solution $x \in L_j$, we have $1 \leq g(x) \leq j \cdot p_{\max}$. Using multiplicative drift analysis, the expected number of offspring created from a solution $x \in L_j$ before having obtained a feasible solution $x^*$ with $p(x^*) \geq p(x^{j,\max})$ is therefore $O(n^2(\log n + \log p_{\max}))$. Moreover, $x^*$ is as good as the best solution in $L_j$, $x^*$ contains the top $j$ elements regarding to the profits of items, this implies that $X^*$ has the same construction as $x^{j,\max}$ and is a balanced solution that has smallest probability in $L_j$.

If $x^*$ is not optimal, then there exists a 1-bit flip adding element and strictly improving the profit. There are at most $n$ levels $L_j$ which implies that the expected time until an optimal solution has been achieved is $O(n^3(\log n + \log p_{\max}))$. □

## 5.4 Experiments

In this section, we study the performance of the algorithm through experiments. The goal is to supplement the theoretical analysis to further understand the behaviour of the algorithm during the optimisation process. In addition, by analysing the running time of our algorithms on design examples of different scales, we can understand their performance more clearly.

### 5.4.1 Benchmarking and Experimental Setting

To compare the actual performance of different algorithms in different types of CCKP with correlated uniform weights, we used 30 different instances, each of which consists of $n$ items. By selecting the number of groups $K$, the number of items corresponding to each group $m$, the variance $d$ and the item covariance $c$ to generate each instance of the CCKP size $n$ with relevant uniform weights, we set $C = n/2$, $\alpha = 0.001$. For the uniform profit example, we set the expected weight and profit of the project to 1. For the example of arbitrary profit, we set the expected weight of the project to 1 and assign a group of projects to $p_1, \ldots, p_m$ in descending order.

In the first experiment, we considered the expected time for the algorithm to achieve a feasible solution. We run each algorithm on six sizes of instances $n = \{200, 500, 1000, 2000, 3000, 4000\}$. For each size $n$, we design thirty different instances. We record the average number of evaluations for each algorithm to find the first feasible solution as the expected time for all instances with same $n$. Therefore, for each algorithm, we have six points to estimate the performance (Figure 5.1).

The second experiment shows how the algorithms converge to the optimal solution for instances with uniform profits and arbitrary profits. We set $n = \{200, 500, 1000, 2000, 3000, 4000, 5000, 6000\}$. For each $n$, we design and run 30 instances of each algorithm. We record the average number of evaluations to find the optimal solution for 30

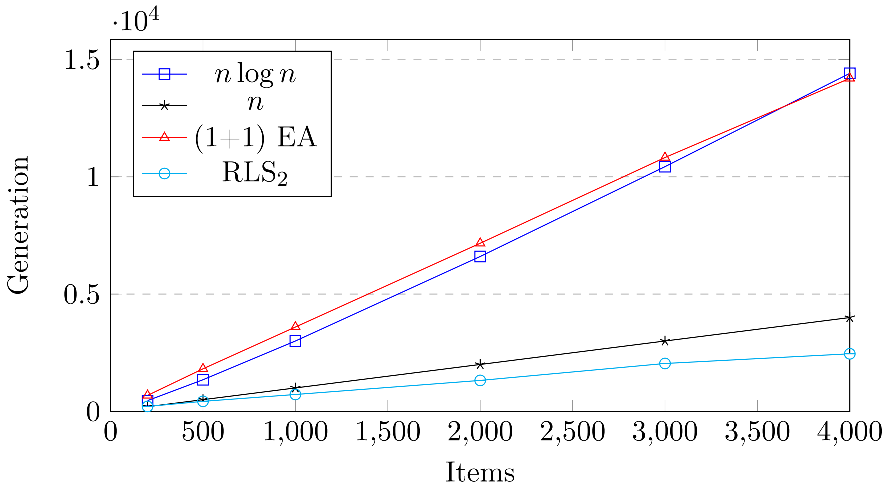


FIGURE 5.1. Expected time of the algorithms to achieve a feasible solution

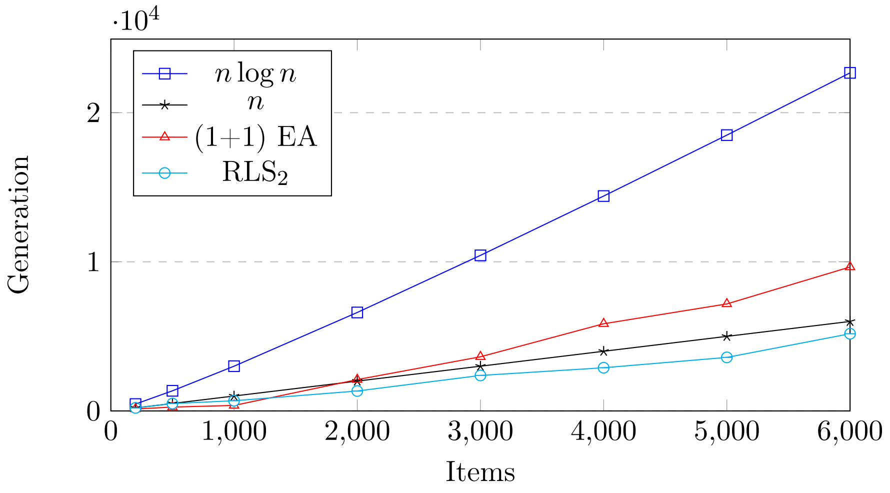


FIGURE 5.2. Running time of the algorithms with uniform profit.

instances as its expected time. Therefore, for each algorithm, we have eight points to estimate the performance in both cases and show the results in Figure 5.2 and 5.3.

Moreover, in the last experiment, we show how the number of groups and the number of items in each group affects the generation required to obtain the optimal solution for the same total number of items. We run each algorithm on five instances of different sizes, create with the same constants as the previous one, with six different group numbers for each size. We record the average number of evaluations for each group number to find the best solution for 30 runs as its optimisation time. We only do this analysis on (1+1) EA (Figures 5.4 and 5.5).

### 5.4.2 Analysis

In this section, we analyse the performance of the algorithm according to the experimental results. Figure 5.1 illustrates how the algorithm finds a feasible solution for all instances on average. This data can be used to gain insight into the expected time of each algorithm. Figure 5.1 shows the expression data of $n$ and $n \log n$, indicating that (1+1) EA running time is close to $O(n \log n)$ and $RLS_2$ close $O(n)$ Random instance.

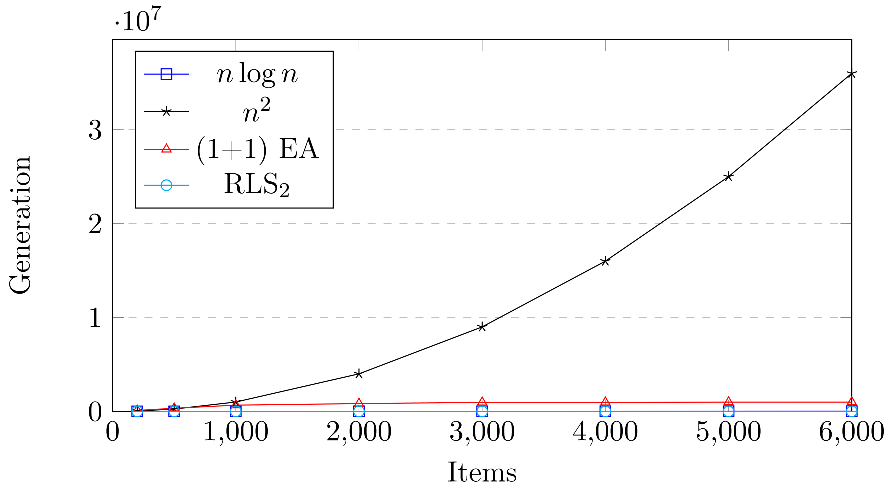


FIGURE 5.3. Running time of the algorithms with arbitrary profit.

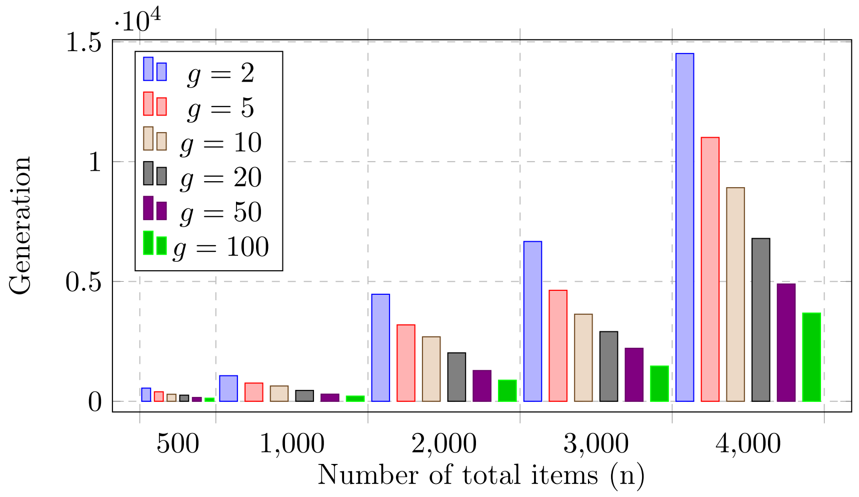


FIGURE 5.4. Comparison between the running time of the (1+1) EA according to the number of groups on uniform cases.

The results of the second experiment are shown in Figure 5.2 and Figure 5.3. We run each algorithm against eight different sizes, and for each specific size, we have thirty different instances. Figure 5.2 shows the average require generation of both algorithms in the case of uniform profit. The line graph shows that the expected optimal time of the two algorithms is close to $n$ or $n \log n$. Figure 5.4 and 5.5 present the average of expected running times for two cases with different number of groups but same $n$. As shown in these figures, in the same $n$, with the increase of $g$, the expected running time decrease. This data can give an insight into the expected optimisation time for a different number of groups within the same size of instances.

The experimental results expressed in Figure 5.2 and 5.3 indicate that the bounds presented in this chapter are a gross overestimate of the actual convergence of the methods. This finding was expected and suggested that the analytical bounds are weak and could potentially be improved significantly. Therefore, future research should investigate the run-time bounds of two algorithms on solving different types of instances guided by these experimental results.

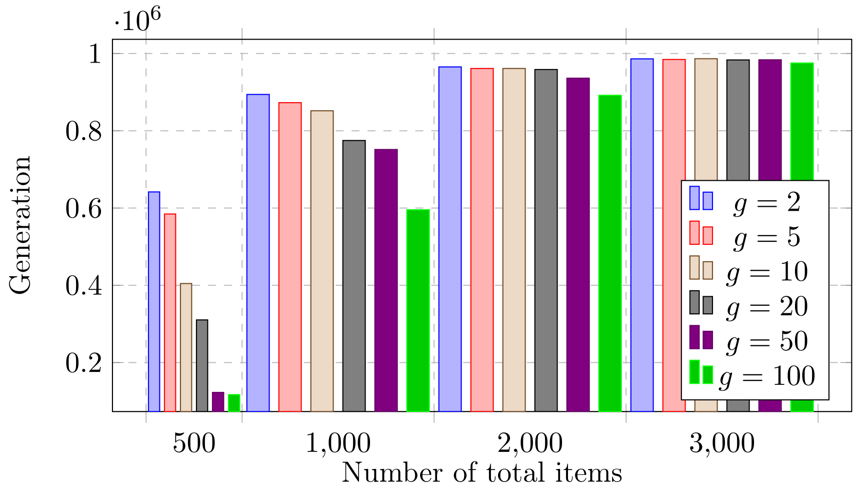


FIGURE 5.5. Comparison between the running time of the (1+1) EA according to the number of groups on arbitrary cases.

## 5.5 Conclusions

The chance-constrained knapsack problem with correlated uniform weights plays a key role in situations where dependent stochastic components are involved. We have carried out a theoretical analysis on the expected optimization time of RLS$_2$ and the (1+1) EA on the chance-constrained knapsack problem with correlated uniform weights in this chapter. We are interested in minimizing the probability that our solution will violate the constraint. We prove the bounds for both algorithms for producing a feasible solution. Then we carried out analyses of two settings for the problem, the one with uniform profits and the groups in the second case has the same profits profile. Our proofs are designed to provide insight into the structure of these problems and to reveal new challenges in deriving runtime bounds in the chance-constrained setting with the general type of stochastic variables.

# Chapter 6

# The Large-Scale Stockpile Blending Problem

## 6.1 Introduction

In this chapter, we concentrate on a real-world application problem: the stockpile blending problem, which plays a significant role in the open-pit mine production scheduling (OPMPS) problem in mining engineering.

The OPMPS problem has received a great deal of attention in recent years, both in the academic literature and in the mining industry (Lamghari and Dimitrakopoulos, 2012; Moreno et al., 2017; Sotoudeh et al., 2020). The OPMPS problem is a decision problem, and the aim of which is to maximise net present value by determining the extraction time for each block of ore and the destination to which each block is sent. As an important component of the OPMPS problem, the stockpile blending problem involves determining the rate and quality of production (which are associated with large cash flows), and it takes the constraints of mining scheduling (upstream), process limitation and customer requirements (downstream) into account. The stockpile blending problem can be formulated as a decision problem that entails calculating the volume of ore that should be claimed in each period (within stockpile limitations), and determining the parcel to which the ore should be sent. In solving this problem, the aim is to maximise the volume of valuable materials that are recovered, subject to several constraints.

In this chapter, we study the large-scale stockpile blending problem. As can be inferred from its name, this problem is characterised by challenges relating to scale. There are many parcels under a plan in this problem, and each parcel can claim material from several stockpiles. Many research about long-term or short-term planning for mines has been dome in the operations research and mining engineering communities Blom, Pearce, and Stuckey (2019). However, it is the first time for the large-scale stockpile blending problem be studied independently and published in the evolutionary computation literature. We introduce a realistic model of the stockpile blending problem and describe the related input parameters of production processes in a real-world situation. Subsequently, we introduce an approach based on the differential evolution (DE) algorithm for the large-scale problem and investigate the performance of the approach by examining instances involving real data.

The content of this chapter is based on a conference paper (CEC 2021) (Xie, Neumann, and Neumann, 2021a).

### 6.1.1 Related Work

To solve the OPMPS problem, a plan must be generated to guide the sequence of mining blocks of the ore body. This plan ensures the delivery of specified weights and grades of the mineral raw material to the mill within the period under consideration. The OPMPS problem was first described by Johnson (1968) as a mixed-integer linear model without considering a stockpile. This paper leads to the research that formulated the OPMPS as a mixed-integer program (MIP) with binary variables (Johnson, 1968; Bley et al., 2012; Topal and Ramazan, 2012). To address the challenge of the large-scale OPMPS problem, Osanloo, Gholamnejad, and Karimi (2008) reviewed different models and algorithms in the context of long-term OPMPS. They discussed the advantages and disadvantages of the deterministic and uncertainty-based approaches to solving the long-term production planning problem. However, when the research on OPMPS becoming more complex and more realistic, it becomes difficult for MIP to solve the problem. Lipovetzky et al. (2014) introduced a combined MIP for a mine planning problem, which devises a heuristic objective function in the MIP and can improve the resulting search space for the planner. Samavati et al. (2017) proposed a heuristic approach that combines local branching with a new adaptive branching scheme to tackle the OPMPS problem.

In real-world applications, stockpiles play an indispensable role in OPMPS, and are used to store the materials of different grades and increase the efficiency of the mill. Jupp, Howard, and Everett (2013) proposed four reasons for stockpiling before processing: buffering, blending, storing and separating material into different grades. Robinson (2004) concluded that blending material in a stockpile can lead to a reduction in grade variation. Some researchers have introduced approaches which represent open-pit mine production scheduling with stockpiling (OPMPS+S) problems as nonlinear-integer models, with the assumption that the material mixes homogeneously in the stockpile. Such problems are difficult to solve. Akaike and Dagdelen (1999) proposed a model for mine planning with using a stockpile; however, within the bounds of their model, there was no blending in the stockpile and the material grade in the stockpile was the same as that of the block. Moreno et al. (2017) introduced a linear integer model for OPMPS+S and demonstrated the superiority of their model to other models by comparing their respective objective function values. More recently, Rezakhah and Moreno (2019) used a linear-integer model to approximate the OPMPS+S problem; this model required that the stockpile to have an average grade above a specific limit.

The rest of the chapter is organised as follows. Section 6.2 presents a model of the stockpile blending problem and introduces two repair operators for the tight constraints. Next, heuristic search approaches for small-scale and large-scale stockpile blending problems are introduced in Section 6.3. Section 6.4 presents experimental results, followed by some concluding remarks.

## 6.2 Mathematical Model of Stockpile Blending Problem

In the mining industry, a blending strategy is affected by the corresponding mining plan, which determines the volume and quality of material hauled from the mine to the stockpiles. A blending strategy also leads to changes in a stockpile's grade level. The created parcels have to respond to the market plan, which stipulates certain requirements, such as tonnes concentrate in the final product and the sum of duration of all parcels from downstream process and customers' demands.

Table 6.1. Notation of large-scale stockpile blending problem

| **Indices and sets** | |
|---|---|
| Name | Description |
| $s$ | stockpiles; $s \in \{1, \ldots, S\}$ |
| $m$ | month; $m \in \{1, \ldots, M\}$ |
| $P^m$ | number of parcel in month $m$ |
| $p^m$ | the $p$-th parcel in month $m$; $1, \ldots, P^m$ |
| $o$ | material; $o \in \{Cu, Ag, Fe, Au, U, F, S\}$ |
| **Variables** | |
| Name | Description |
| $x_{ps}^m$ | proportion of the tonnage of material that is provided by stockpile $s$ to $p$-th parcel in month $m$ |
| $t_p^m$ | production time (duration) for $p$-th parcel in month $m$; could be several days |
| $w_p^m$ | tonnage of $p$-th parcel in month $m$ |
| $\theta_{ps}^m$ | available material in stockpile $s$ for providing $p$-th parcel in month $m$ |
| $Cu_p^m$ | tonnage of Cu in the final production of $p$-th parcel in month $m$ |
| $g_p^{om}$ | grade of material $o$ of $p$-th parcel in month $m$ |
| $\tilde{g}_{ps}^{om}$ | grade of material $o$ in stockpile $s$ for providing $p$-th parcel in month $m$ |
| $k_p^m$ | tonne concentrate of final production of $p$-th parcel in month $m$ |
| $r_p^{Cu,m}$ | Cu recovery rate of $p$-th parcel in month $m$ |
| $r_p^{F,m}$ | F recovery rate of $p$-th parcel in month $m$ |
| $r_p^{U,m}$ | U recovery rate of $p$-th parcel in month $m$ |
| **Parameters** | |
| Name | Description |
| $\delta$ | discount factor for time period |
| $D^m$ | available duration of month $m$; different month has different available days |
| $H_s^m$ | tonnage of material hauled from mine to stockpile $s$ in month $m$ |
| $G_s^{om}$ | grade of material $o$ that shipping to the stockpile $s$ in month $m$ |
| $K_p^m$ | expected tonne concentrate of parcel $p$ in month $m$ |
| $B^F$ | upper bound of F recovery rate |
| $B^U$ | upper bound of U recovery rate |
| $B^{Cu}$ | lower bound of Cu grade |
| $D^{Cu}$ | bound of the difference between Cu grades of parcels within the same month |

We now present a mathematical model of the stockpile blending problem. In order to ensure that the model of the problem matches real-world situations, we have to formulate the problem without losing any information, as well as describing the production processes with their corresponding input variables and parameters. **Yue: To avoid disclosing confidential business information, $f_1$ to $f_5$ are used to denote a series of non-linear complexity calculation processes specified separately. The details of processes are provided by our industrial partner, and we are not able to publish them here. However, we point out the inputs of $f_1$ to $f_5$ and give some connection between those parameters and decision variables.**

Table 6.1 presents the relevant notation. We use the term "material" to refer to ore, i.e., rock that contains a quality of sufficient minerals (including metals) that can be economically extracted. The non-linear model of the stockpile blending problem is presented as follows:

$$Obj. \max \sum_{m=1}^{M} \sum_{p=1}^{P^m} Cu_p^m = \sum_{m=1}^{M} \sum_{p=1}^{P^m} f_1\left(t_p^m, g_p^{om}\right) \tag{6.1}$$

$$\begin{aligned}
s.t. \sum_{1 \leq p \leq P^m} t_p^m &\leq D^m && \forall m \in \{1, \dots, M\} && (6.2)\\
\sum_{s \in \mathcal{S}} x_{ps}^m &= 1 && \forall p \in \{1, \dots, P^m\}, \forall m \in \{1, \dots, M\} && (6.3)\\
g_p^{om} &= \sum_{s=1}^{S} x_{ps}^m \tilde{g}_{ps}^{om} && \forall p \in \{1, \dots, P^m\}, \forall m \in \{1, \dots, M\} && (6.4)\\
g_p^{Cu,m} &\geq B^{Cu} && \forall p \in \{1, \dots, P^m\}, \forall m \in \{1, \dots, M\} && (6.5)\\
|g_p^{Cu,m} - g_{p'}^{Cu,m'}| &\leq D^{Cu} && \forall p, p' \quad \text{from any month} && (6.6)\\
w_p^m &= f_2(t_p^m, g_p^{om}) && \forall p \in \{1, \dots, P^m\}, \forall m \in \{1, \dots, M\} && (6.7)\\
k_p^m &= f_3(t_p^m, g_p^{om}) && \forall p \in \{1, \dots, P^m\}, \forall m \in \{1, \dots, M\} && (6.8)\\
(k_p^m - 1) \leq K_p^m &\leq (k_p^m + 1) && \forall p \in \{1, \dots, P^m\}, \forall m \in \{1, \dots, M\} && (6.9)\\
r_p^{Fl,m} &= f_4(g_p^{Fl,m}) && \forall p \in \{1, \dots, P^m\}, \forall m \in \{1, \dots, M\} && (6.10)\\
r_p^{Fl,m} &\leq B^F && \forall p \in \{1, \dots, P^m\}, \forall m \in \{1, \dots, M\} && (6.11)\\
r_p^{U,m} &= f_5(g_p^{U,m}) && \forall p \in \{1, \dots, P^m\}, \forall m \in \{1, \dots, M\} && (6.12)\\
r_p^{U,m} &\leq B^U && \forall p \in \{1, \dots, P^m\}, \forall m \in \{1, \dots, M\} && (6.13)
\end{aligned}$$

$$\tilde{g}_{ps}^{om} = \begin{cases} \frac{H_s^m \cdot G_s^{om} + \tilde{g}_{P^{m-1}s}^{o(m-1)} \cdot (\theta_{P^{m-1}s}^{m-1} - x_{P^{m-1}s}^{m-1} \cdot w_{P^{m-1}}^{m-1})}{\theta_{ps}^m} & \textit{if } p = 1 \\ \tilde{g}_{(p-1)s}^{om} & \textit{otherwise} \end{cases} \quad (6.14)$$

$$\theta_{ps}^m = \begin{cases} \theta_{P^{m-1}s}^{m-1} + H_s^m - x_{P^{m-1}s}^{m-1} \cdot w_{P^m}^{m-1} & \textit{if } p = 1 \\ \theta_{(p-1)s}^m - x_{(p-1)s}^m \cdot w_{p-1}^m & \textit{otherwise.} \end{cases} \quad (6.15)$$

As shown in the model, there are two kinds of decision variables in the problem. The first variable is a vector, $X = \{x_{11}^1, \dots x_{P^1S}^1, \dots, x_{P^MS}^M\}$, which consists of $S \times \sum_{m=1}^{M} P^m$ decision variables. $x_{ps}^m$ is a continuous variable which takes the values in $(0, 1)$, and it denotes the proportion of the tonnage of material that is provided by stockpile $s$ to $p$-th parcel in month $m$. The other variables are encoded as a vector of continuous decision variables $T = \{t_p^1, \dots, t_{P1}^i, \dots, t_{PM}^M\}$, where $t_p^m$ denotes the working duration of $p$-th parcel in month $m$.

The objective function (6.1) is the sum of the tonnage of Cu in the final production of all parcels, in which the tonnage of copper is inextricably intertwined with material grades and the duration of parcel.

Constraint (6.2) ensures that the sum of duration in each month is less than or equal to the available working days of this month. Constraint (6.3) forces the sum of the decision variables of a parcel to equal one. Equation (6.4) calculates the material grades of each parcel which are related to the material grades in corresponding stockpiles and the decision variables of the parcel. Constraint (6.5) guarantees that the Cu grade of each parcel should be at least a given bound. $f_2$ in function (6.7) denotes the manner in which the tonnage of parcels is calculated. $f_3$ in function (6.8) denotes the calculation process of the amount of final production of each parcel, and the value of $k_p^m$ is limited by constraint (6.9), which is a tight constraint in the model. Functions (6.10) and (6.12) denote the calculation process of the F recovery rate and the U

```
Algorithm 13: Decision variables normalised approach
Input: Decision vector $X_j = \{x_{1j}, x_{2j}.., x_{Ij}\}$
$a = \sum_{i \in I} x_{ij}$;
for $i = 1$ to $I$ do
    $x_{ij} = \frac{x_{ij}}{a}$;
return the normalised decision variables.
```

recovery rate, respectively. Constraints (6.11) and (6.13) ensure the $r_p^{Fl,m}$ and $r_p^{U,m}$ of parcels are less than or equal to the given thresholds. Constraints (6.14) and (6.15) enforce the balance of material grades and inventory for stockpiles with respect to the provision of material to parcels and thr quantity of ore that is hauled. Since the ore is hauled from the mine to the stockpiles at the beginning of every month, the stockpile material grades are updated once at the beginning of each month, as well as being updated constantly for all parcels in that month. The available material in each stockpile denoted by $\theta_{ps}^m$ should be non-negative.

### 6.2.1 Repaired Operators

The stockpile blending problem is a non-linear optimisation problem in continuous search space, and it is difficult to solve the problem with respect to the problem's constraints. **Yue: Regarding the implementation process of evolutionary algorithms, we found two constraints in the model that lead to the difficulty of algorithms to reach a feasible solution. These two constraints are constraint (6.3) and (6.9). The first one force the sum of the decision variables of each parcel equal to one, and the second one force the value of real tonne concentrate of each parcel to be no more or less one than the given bound. Here, we present two repaired operators to tackle the complexity constraints separately.**

To tackle constraint (6.3), we introduce a decision-variables normalised approach (cf. Algorithm 13) to force the solution to match the constraint. This approach first entails calculating the sum of the variables of each parcel separately, dividing each variable of the same parcel by the corresponding sum.

For the tight constraint (6.9), it is hard to generate a feasible solution that forces the real amount of tonnes concentrate of a parcel is no more or less one than the given threshold. As shown in equation (6.8), the amount of tonnes concentrate of each parcel is related to the duration of this parcel and the material grades of that particular parcel. Meanwhile, the metal grades of the parcel are directly calculated by decision variables according to equation (6.4)). Therefore, based on the determined decision variables of a parcel, the real tonnes concentrate of the parcel are affected by the duration of the parcel. Consider the constraint (6.9) is the tightest constraint in the model and it is hard for evolutionary algorithms to achieve a solution that satisfies this constraint in reasonable iterations, we present a duration repair operator (cf. Algorithm 14) which uses a binary search process to convert an infeasible solution into a solution without violating constraints (6.9).

The time complexity of the binary search is $\log n$ where $n$ denotes the length of the search space in the beginning. In the stockpile blending problem, the duration of each parcel cannot exceed the total available duration of the month. Therefore, the run-time of the duration repair operator for one parcel is $\log d$ in the worst case where $d$ denotes the total available duration of the current month.

**Algorithm 14:** Duration repair operator

```
Input: $X \in (0,1)^{I \cdot J}$, $i \in \{1,..,I\}$, $j \in \{1,..,J\}$; parameter $\zeta$; available duration
       $D$
Output: parcel duration: $d \in \{0, D\}$
initialization: $\underline{d} = 0$, $\overline{d} = D$, $d \in \{0, D\}$ , $k = \zeta \cdot d$
while $d \in \{0, D\}$ and $k \notin \{K-1, K+1\}$ do
    if $k > K+1$ then
        $d := (d + \underline{d})/2$;
        $k := \zeta \cdot d$;
        if $k > K+1$ then
            $\overline{d} = d$;
        else
            $\underline{d} = d$;
    else if $k < K-1$ then
        $d := (d + \overline{d})/2$
    $k := \zeta \cdot d$;
    if $k > K+1$ then
        $\overline{d} := d$;
    else
        $\underline{d} := d$
return the duration corresponding to solution $X$
```

## 6.3 Heuristic Search Approach

In this section, we investigate the differential evolution (DE) algorithm on solving the stockpile blending problem. As introduced in Section 3.3.4, the DE algorithm is a well-known evolutionary computation approach which was developed to solve the global optimisation problems in continuous search space where the objective function can be nonlinear (Das and Suganthan, 2010). However, the large-scale stockpile blending problem is difficult to solve due to its significant number of constraints. Therefore, we introduce a strategy to address this challenge in the rest of this section.

We start by designing a fitness function that can be used in the heuristic approach. The fitness function $f$ takes into account all constraints. We define the fitness function of a solution $X$ as follows:

$$f(X) = (u(X), v(X), w(X), p(X), q(X), g(X), h(X), C(X)) \tag{6.16}$$

where

$$
\begin{aligned}
u(X) &= \sum_{m=1}^{M}\sum_{p=1}^{P^m}\max\{\left|K_p^m - k_p^m\right|, 1\}\\
v(X) &= \sum_{m=1}^{M}\max\{\sum_{p=1}^{P^m} t_p^m - D^m, 0\}\\
w(X) &= \sum_{m=1}^{M}\sum_{p=1}^{P^m}\sum_{s=1}^{S}\min\{\theta_{ps}^m, 0\}\\
p(X) &= \sum_{m=1}^{M}\sum_{p=1}^{P^m}\max\{r_p^{U,m} - B^U, 0\}\\
q(X) &= \sum_{m=1}^{M}\sum_{p=1}^{P^m}\max\{r_p^{Fl,m} - B^{Fl}, 0\}\\
g(X) &= \sum_{m=1}^{M}\sum_{p=1}^{P^m}\max\{B^{Cu} - g_p^{Cu,m}, 0\}\\
C(X) &= \sum_{m=1}^{M}\sum_{p=1}^{P^m} Cu_p^m.
\end{aligned}
$$

In this fitness function, $u(X)$, $v(X)$, $p(X)$, $q(X)$ and $g(X)$ need to be minimised while $w(X)$ and $C(X)$ need to be maximised. We optimise $f$ in lexicographic order (Fishburn, 1974), and the function takes into account all constraints. According to the fitness function, an infeasible solution can at least violate at least one of the above constraints. Among solutions that meet all constraints, we aim to maximise the combined tonnage of copper across all parcels'.

Formally, we have

$$
\begin{aligned}
& f(X) \succeq f(Y)\\
\textbf{iff } & u(X) < u(Y) \textit{ or}\\
& u(X) = u(Y) \land v(X) < v(Y) \textit{ or}\\
& \{u, v\}\text{are equal} \land w(X) > w(Y) \textit{ or}\\
& \{u, v, w\}\text{are equal} \land p(X) < p(Y) \textit{ or}\\
& \{u, v, w, p\}\text{are equal} \land q(X) < q(Y) \textit{ or}\\
& \{u, v, w, p, q\}\text{are equal} \land g(X) < g(Y) \textit{or}\\
& \{u, v, w, p, q, g\} \text{ are equal} \land C(X) > C(Y).
\end{aligned}
$$

When comparing an infeasible and a feasible solution, the feasible solution is preferred. Between two infeasible solutions violating the same constraint, the solution with the lower degree of constraint violation is preferred. The fitness function (6.16) can be applied in any heuristic approach. In this section, we investigate the performance of the classical DE algorithm (see Algorithm 15).

As discussed above, it is difficult to obtain a feasible solution for a large-scale stockpile blending problem, let along to reach the optimal solution. However, for a stockpile blending problem that only contains one month ($M = 1$), that is, one in which no

**Algorithm 15:** Differential evolution algorithm

```
Generate initial population of size $NP$ ;
while stopping criterion not met do
    for each individual $t$ in the population do
        Generate three random integers, $r_1, r_2, r_3 \in (1, NP)$, with
          $r_1 \neq r_2 \neq r_3 \neq t$  Generate a random integer $m_{rand} \in (1, n)$; for each
          parameter $i$ of the individual do
            Generate mutant vector $V_t$ and trial vector $U_t$;
        Replace $X_t$ with $U_t$, if $U_t$ is better according to the fitness function;
return the best solution in the final population according to the fitness function.
```

**Algorithm 16:** DE approach for one-month problem

```
Initialization: initial population of size NP with applying Algorithm (13) to
  normalize decision variables of all parcels in this month;
while stopping criterion not met do
    for each individual in the population do
        Generate the trial vector by applying the DE algorithm (Alg. 15);
        Normalize the trial vector by using solution fixed operator (Alg. 13);
        Search the duration of parcels by applying operator (Alg. 14);
        Compare the new solution to the chosen individual via fitness function
          (6.16) with setting $M = 1$;
        Keep the best one for next population.
return the solution in the final population with maximum $C(X)$.
```

more than four parcels are under consideration ($P^m = 4$), DE can solve the problem effectively. Therefore, we define the one-month stockpile blending problem as a unit problem of the large-scale stockpile blending problem. We propose a DE-based approach (see Algorithm 16) that combines the classical DE algorithm and the decision variables normalised operator and the duration-repaired operator for a unit problem.

Furthermore, by observing Equations (6.14) and (6.15), we found that, the available material grades and the tonnage of ore for the first parcel in each month are affected by the blending strategy of the previous month and the ore hauled from mines. Here, we propose to optimise the large-scale stockpile blending problem month by month.

To solve the problem, it is necessary to maximise the sum of $Cu_p^m$, and we observe that parcels with a higher grade of Cu have a higher value of tonnage of Cu. Therefore, the DE-based approach always obtains a feasible solution which has the highest objective value and leads to the highest copper grade. The approach preferentially blends material for those stockpiles with high copper grades. However, constraint (6.5) necessitates that the maximum difference between the Cu grades of parcels be lower than a given threshold. It becomes extremely difficult to maintain the consistency of Cu grades among all parcels when optimising the problem month by month.

To tackle the problem of maintaining the consistency of Cu grades across parcels, we introduce a second objective function for the one-month stockpile blending problem. The second objective function satisfies the predefined maximum difference of copper grades. The developed fitness function $f'(X)$ of a solution is given as:

$$f'(X) = [u(X), v(X), w(X), p(X), q(X), g(X), h(X), (C(X), C^*(X))] \tag{6.17}$$

**Algorithm 17:** DE approach for long-term problem

```
Initialize the feasible solution set $Z$.
for $m$ in $\{1, ..M\}$ do
    Copy all element from $Z$ to $Z'$;
    Clean $Z$;
    for element $z$ in $Z'$ do
        Update $S_{ij}$ and $L^o_{ij}$ by considering ore hauled of this month and the
          solution of $z$;
        Apply the DE-based Algorithm 16 for this month by using fitness function
          (6.17);
        Add the feasible solutions in last population to $Z$;
return The best solution in $Z$.
```

where $C^*(X) = \sum_{p=1}^{Pm} x_{(ps^*)}$ and $x_{(ps^*)}$, and $s^*$ denotes the stockpile which has the highest copper grade in each month. This specific stockpile might be different in each month and is only chosen according to the grade of the Cu. The use of the multi-component in the fitness function make it possible to cater to the need to maintain a consistent Cu grade. In the bi-objective optimisation of the large-scale blending optimisation problem, the goal is to maximise $C(X)$ and minimise $C^*(X)$ while satisfying all constraints. Here, we have

$$f'(X) \succeq f'(Y) \text{ iff } C(X) \geq C(Y) \wedge C^*(X) \leq C^*(Y)$$

for the dominance relation of bi-objective optimisation for two solutions $X$ and $Y$.

Solving the large-scale stockpile blending problem is a challenge due to its significant number of constraints and the fact that its scale can be very large. For example, if an instance has ten parcels and can claim material from six different stockpiles, then the search space of this instance will be a sixty-dimension space for decision vectors combined with a ten-dimension space for the duration of parcels. Although DE has become a popular and effective algorithm for continuous optimisation problems, most reported studies on DE involve the solution of small-scale problems. It becomes difficult for the DE algorithm to solve large-scale stockpile blending problems within an acceptable amount of computation time.

Here, we propose an approach (cf. Algorithm 17) which involves optimising the problem month by month using the fitness function (6.17). The approach treats every one-month problem as a unit-problem and obtains a set of feasible solutions for each month using Algorithm 16. A such, the processes involved in this approach are as follows. (1) For every month, the approach adopts a set of feasible solutions $Z$ of the problem for the previous month; (2) for every solution in the set $Z$, it updates the parameters ($\tilde{g}^{om}_{ps}$ and $\theta^m_{ps}$) of stockpiles for this month; (3) it applies Algorithm 16 to obtain a set of feasible solutions for this month and adds them to a set $Z'$; and (4) it loops all feasible solutions from the previous month and selects a fixed number from the set $Z'$ randomly. The steps of this approach are repeated until the problem has been solved for each month. In this section, we set the number of feasible solutions in each month such that is equal to the population of the DE algorithm and randomly select them from $Z$.

TABLE 6.2. Results for one-month problem

| **Index** | **Parcels** | **Stockpiles** | **Org** | **DE-based** | | | |
|---|---|---|---|---|---|---|---|
| | | | | Max | Min | Mean | Std. |
| 1 | 2 | $\{6, 6\}$ | 5777.62 | **6167.46** | 5815.87 | 5989.31 | 99.40 |
| 2 | 2 | $\{6, 7\}$ | 4180.72 | **4383.60** | 4222.38 | 4298.08 | 36.47 |
| 3 | 2 | $\{7, 7\}$ | 5371.08 | **5611.94** | 5373.62 | 5472.12 | 65.12 |
| 4 | 3 | $\{7, 7, 7\}$ | 5124.92 | **5251.86** | 5129.03 | 5164.69 | 26.65 |
| 5 | 2 | $\{7, 7\}$ | 5484.07 | **5600.56** | 5488.43 | 5536.35 | 27.79 |
| 6 | 2 | $\{7, 7\}$ | 4334.78 | **4438.87** | 4340.60 | 4376.00 | 28.86 |
| 7 | 2 | $\{7, 7\}$ | 5243.46 | **5351.31** | 5250.24 | 5287.49 | 26.46 |
| 8 | 3 | $\{7, 7, 7\}$ | 5257.26 | **5411.48** | 5265.74 | 5342.09 | 32.35 |

## 6.4 Experimental Investigation

This section evaluates the efficiency of the proposed DE-based heuristic search approaches to the one-month stockpile blending problem and the large scale problem. We first compare the performance of the DE-based approach and the strategy used in real-world situations with respect to one-month instances. Our industrial partner provided the parameters values and the solutions for those instances. Then, the results obtained using the approach 17 were compared to the real-world results on large-scale stockpile blending instances. All experiments were performed using Java of version 11.0.1 and carried out on a MacBook with a 2.3GHz Intel Core i5 CPU.

We first estimate the performance of the DE approach with respect to one-month instances. The setup of the experiments and the results obtained through the different approaches are summarised in Table 6.2. For example, the instance with index 2 contains two parcels, and the first parcel can claim material from six different stockpiles, while the second parcel can claim material from the six stockpiles and another stockpile which was available during this period. In the implementation, each approach runs for $10^5$ fitness evaluations, and the DE approach runs the following parameters: $NP = 10$, $F = 1.2$ and $CR = 0.5$. Table 6.2 reports the average, maximum, minimum and standard deviation of the results of the instances obtained through the approach for 30 independent runs. The column *Parcels* refers to the number of parcels contained in the instances, and the column 'Stockpiles' presents the number of available stockpiles for each parcel. The list of results in the column 'Org' presents the real-world results, as provided by our industrial partner. We set the original solution of instances as the initial solution of the approaches.

As can be seen from Table 6.2, the results obtained through the DE approach were significantly better than the original results across all instances. Even the minimum results of all instances had higher values than the original results, indicating that the DE-based approach guaranteed better results for each of the 30 runs. Solving the stockpile blending problem is important, in that is an important component of OPMPS, a process which determines the quality of production and involves large cash flows. Therefore, the ability of a company to solve stockpile blending problems can make a difference of hundreds of millions of dollars. These results show that, for the one-month stockpile blending problem, the DE algorithm (combined with our proposed repair operators) is able to achieve higher Cu tonnage than the real-world

TABLE 6.3. Results for long-term problem

| **Index** | **Month** | **Parcels** | **Length** | **Org** | **DE-based** | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Max | Min | Mean | Std |
| 1 | 5 | 11 | 76 | 25938.41 | **26085.17** | 25988.38 | 26008.62 | 72.70 |
| 2 | 5 | 11 | 76 | 24495.58 | **24732.95** | 24497.95 | 24584.56 | 57.08 |
| 3 | 5 | 11 | 77 | 25558.31 | **25704.27** | 25565.20 | 25653.88 | 39.61 |
| 4 | 6 | 13 | 88 | 30273.20 | **30434.64** | 30289.02 | 30328.28 | 78.76 |
| 5 | 6 | 13 | 90 | 29739.03 | **29965.60** | 29761.57 | 29850.17 | 64.47 |
| 6 | 6 | 14 | 91 | 30815.57 | **31069.36** | 30841.72 | 30926.58 | 63.11 |
| 7 | 7 | 15 | 102 | 35516.65 | **35660.19** | 35530.73 | 35566.41 | 63.19 |
| 8 | 7 | 16 | 111 | 34996.29 | **35321.51** | 35009.17 | 35113.01 | 62.81 |
| 9 | 7 | 16 | 104 | 36025.85 | **36150.97** | 36029.17 | 36077.91 | 36.23 |
| 10 | 8 | 18 | 123 | 40773.92 | **41011.07** | 40782.25 | 40852.73 | 80.31 |
| 11 | 8 | 18 | 124 | 40206.57 | **40390.00** | 40213.26 | 40253.22 | 36.84 |
| 12 | 8 | 18 | 118 | 41297.38 | **41405.50** | 41308.73 | 41343.51 | 33.02 |

results, which can lead to benefits exceeding hundreds of millions of dollars in real-world situations.

We now analyse large-scale instances and compare the results obtained through DE and the real-world results. Table 6.3 lists average, maximum, minimum of the results of the instances obtained through the approach for 30 independent runs. The table also lists the results used in the real-world situation for all instances.

As shown in Table 6.3, the maximum result for each instance has higher values than the original results, indicating that the DE-based approach is outperforming the original results and leading to a relative profit exceeding hundreds of millions of dollars. Moreover, even the minimum result of each instance has higher values than the original results. This shows that, in every run, the DE-based approach can reach a better solution in term of objective value than the original result. Therefore, the results obtained by the DE approach for large-scale stockpile blending instances are significantly better than the real-world results.

## 6.5 Conclusions

This chapter studied the large-scale stockpile blending problem which subjects to a set of constraints dictated by the mine schedule and the demands of downstream customers. It is a challenge to solve the problem concerning the problem's constraints. We presented two repaired operators to improve the efficiency of finding a feasible solution in the mentioned approaches. Moreover, we divided the large-scale problem into several one-month problems and treated a one-month problem as a unit problem. Two different fitness functions are presented for the one-month stockpile blending problem and large-scale stockpile blending problem separately. The fitness function for the large-scale problem is a multi-component function, and it measures the usage of high-quality stockpiles for the large-scale stockpile blending problem. We presented a DE approach for a large-scale stockpile blending problem that optimises the problem month by month. This approach guarantees the quality of each unit solution and the balance of used material between stockpiles. In the experiment section, we first investigated the DE algorithm associate with the two repaired operators for one-month stockpiles blending problems. Then, we investigated the performance of the proposed approach for the large-scale stockpiles blending problem by testing real-world instances. The results show that the DE approach obtains significantly better results than the results used in real-world situations for all instances. For future research on the large-scale stockpile blending problem, it would be interesting to compare the performance of DE and other methods. Moreover, instead of selecting a random set of feasible solutions of each month carried forward to the next month, performing a study on selection strategies would be another exciting research direction.

Chapter 7

# The Stockpile Blending Problem with Chance Constraints

## 7.1 Introduction

As discussed in the previous chapter, an improvement in the results of the stockpile blending problem can lead to cash flows that can add several hundred million dollars in magnitude. It becomes critical to solving the stockpile blending problem in mining optimally. Moreover, in real-world situations, the blending strategies are affected by the uncertainty of materials from the available resource in mines. This uncertainty is acknowledged in the related technical literature to be the primary reason for not meeting production expectations (Baker and Giacomo, 1998; Asad and Dimitrakopoulos, 2012). Given its substantial impact on the financial outcome of mining operations, this chapter focuses on dealing with the uncertainty in metal content within a mineral deposit being mined.

Motivated by the uncertainty in the geologic input data and the uncertainty that can affect optimisation, we consider the stockpile blending problem with uncertainty in material grades and denote them as stochastic variables. To tackle those stochastic variables, we apply the chance-constrained programming introduced in Section 2.3. The objective function of the stockpile blending problem with chance constraints is the same as that of the large-scale stockpile blending problem. However, the stockpile blending problem with chance constraints is subject to the constraints that the probability of inequality constraints that violated a given bound is less than a given threshold.

Furthermore, we investigate the differential evolution (DE) algorithm for the stockpile blending problem with chance constraints. In the experiment section, we compare the performance of the DE algorithm with the deterministic model and three chance constraint models by using a synthetic benchmark. We also evaluate the effectiveness of different chance constraints.

The content of this chapter is based on a conference paper (GECCO 2021) (Xie, Neumann, and Neumann, 2021b).

The rest of the chapter is organised as follows. Section 7.2 presents a model of the stockpile blending problem. The model in this chapter is different to the one in the previous chapter to contain uncertainty into the problem. Next, a chance constraints model and surrogate functions of the chance constraints are presented in Section 7.3. Afterwards, approaches and fitness functions for the stockpile blending problem with chance constraints are introduced in Section 7.4. We set up experiments and

TABLE 7.1. Notation of simplified version of the stockpile blending problem

| | **Indices and sets** |
|---|---|
| Name | Description |
| $s \in \mathcal{S}$ | stockpiles; $1, \dots, S$ |
| $p \in \mathcal{P}$ | parcels; $1, \dots, P$ |
| $o$ | material; material; $\{Cu, Ag, Fe, Au, U, F, S\}$ |
| | **Variables** |
| Name | Description |
| $x_{ps}$ | fraction of the material that parcel $p$ claimed from stockpile $s$ |
| $t_p$ | produce time (duration) for parcel $p$ |
| $w_p$: | tonnage of parcel $p$ |
| $\theta_{ps}$: | tonnage stores in stockpile $s$ when providing material to parcel $p$ |
| $c_p$: | tonnage of Cu in the final production of parcel $p$ |
| $g_p^o$: | grade of material $o$ in parcel $p$ |
| $\tilde{g}_{ps}^o$: | grade of material $o$ in stockpile $s$ when proving parcel $p$ |
| $k_p$: | tonne concentrate in final production of parcel $p$ |
| $r_p^{Cu}$: | Cu recovery rate of parcel $p$ |
| $r_p^F$: | F recovery rate of parcel $p$ |
| | **Parameters** |
| Name | Description |
| $\delta$: | discount factor for time period |
| $\tilde{\phi}$: | factor in chemical processing stage |
| $\phi^{Au}$: | factor of Au in chemical processing stage |
| $\phi^U$: | factor of U in chemical processing stage |
| $\phi^{Fe}$: | factor of Fe in chemical processing stage |
| $\phi^{Cu}$: | factor of Cu in chemical processing stage |
| $(\gamma_1, \gamma_2)$ : | factor of Cu percentage within the produced Cu concentrate |
| $\mu^F$: | factor of F recovery |
| $\mu^U$ : | factor of U recovery |
| $(\mu_1^{Cu}, \mu_2^{Cu})$ : | factor of Cu recovery |
| $D$: | total available duration |
| $H_s$: | available material in stockpile $s$ before providing material to the first parcel |
| $G_s^o$: | grade of material $o$ in stockpile $s$ before providing material to the first parcel |
| $K_p$: | expected tonne concentrate of parcel $p$ |
| $R_p^F$ : | upper threshold of F recovery rate of parcel $p$ |
| $Cu_p$ : | lower threshold of Cu grade of parcel $p$ |

investigate the performance of the different fitness functions in Section 7.5, followed by some concluding remarks in Section 7.6.

## 7.2 Deterministic Model

We now present a nonlinear model of the stockpiles blending problem in a deterministic setting. In reality, some processes, such as the chemical process in the concentrate production progress, are highly complex to model because they depend on many factors, including the mineralogy of the ore, the particle size of milled material, temperature, and chemical reactants available in the process. Since we do not know the information of those variables, we use $f_1$ to $f_5$ to represent a series of calculation processes of the value of some variables in the previous chapter.

However, in this chapter, to investigate the uncertainty in the input data, instead of using $f_1$ to $f_5$, we introduce a simplified version of the stockpile blending problem and only discuss the small-scale problem here (one-month duration). Moreover, the recovery rate factors of all materials from the chemical processing stage and the copper percentage within the produced copper concentrate is assumed to be constant throughout the stockpiles blending and production schedule.

Table 7.1 presents the relevant notation. We use the term "material" to include ore, i.e., rock that contains a sufficient quantity of minerals. Then, we introduce the nonlinear formulations of the one-month stockpiles blending problem.

$$
\begin{aligned}
Obj: \quad & \max \sum_{p \in \mathcal{P}} c_p = max \sum_{p \in \mathcal{P}} \left( w_p {g_p^{Cu}} {r_p^{Cu}} \right) && && (7.1) \\
s.t. \quad & \sum_{p \in \mathcal{P}} t_p \leq D^m && && (7.2) \\
& \sum_{s \in \mathcal{S}} x_{ps} = 1 && \forall p \in \mathcal{P} && (7.3) \\
& r_p^{Cu} = \mu_1^{Cu} \frac{g_p^{Cu}}{g_p^S} + \mu_2^{Cu} && \forall p \in \mathcal{P} && (7.4) \\
& g_p^o = \sum_{s \in \mathcal{S}} x_{ps} \tilde{g}_{ps}^o && \forall p \in \mathcal{P} && (7.5) \\
& w_p = \delta t_p [\tilde{\phi} + (\phi^{Au} \log g_p^{Au}) + (\phi^U \log g_p^U) \\
& \quad - (\phi^{Fe} \log g_p^{Fe}) + (\phi^{Cu} \log g_p^{Cu})] && && (7.6) \\
& k_p = \frac{c_p}{\gamma_1 \frac{g_p^{Cu}}{g_p^S} + \gamma_2} && \forall p \in \mathcal{P} && (7.7) \\
& \tilde{g}_{ps}^o = G_s^o && \forall p \in \mathcal{P} && (7.8) \\
& g_p^{Cu} \geq Cu_p && \forall p \in \mathcal{P} && (7.9) \\
& (K_p - 1) \leq k_p \leq (K_p + 1) && \forall p \in \mathcal{P} && (7.10) \\
& \mu^F g_p^F \leq R_p^F && \forall p \in \mathcal{P} && (7.11) \\
& 0 \leq \theta_{ps} = \begin{cases} H_s & \mathit{if}\ p = 1 \\ \theta_{(p-1)s} - x_{(p-1)s} \cdot w_{p-1} & otherwise \end{cases} && && (7.12)
\end{aligned}
$$

The objective function (7.1) is the sum of the tonnage of Cu in the final production of all parcels, in which the tonnage of Cu in each parcel is calculated by the tonnage of parcel multiply the Cu grade of the parcel and multiple the Cu recovery rate of the parcel. Constraint (7.2) ensures that the sum of duration in each month is less than or equal to the available working days of this month.

Constraint (7.3) forces the sum of the decision variables of a parcel to equal one. Equation (7.4) calculates the Cu recovery rate of each parcel, and Equation (7.5) calculates the material grades of each parcel. Equation (7.6) express the simplified calculation of parcel tonne which is a component in objection function. Equation (7.7) shows the simplified version of how to calculate the tonne concentrate of each parcel. Constraint (7.12) enforces inventory balance when providing material to parcels that are not the first one, ensuring that the amount of material store in each stockpile is equal to that of the last parcel minus anything sent to the last parcel from the stockpile. Constraint (7.9) forces the Cu grade of each parcel to be less than or equal to the pre-given lower threshold of Cu grade of each parcel. Constraint (7.10) forces the value of tonne concentrate of each parcel to be not more or less than the expected tonne concentrate by one. Constraint (7.11) ensures the F recovery rate of each parcel is less than the given threshold.

In this chapter, to tackle the complexity and tight constraints in the model of the one-month stockpile blending problem, we apply the two repaired operators introduced in Section 6.2.1 to constraints (7.3) and (7.10).

## 7.3 Model with Chance Constraints

This section discusses the effeteness of stochastic material grades on the objective value of the stockpile blending problem. Due to the complexity of the problem, we reformulated the constraints (7.9) and (7.11) to chance constraints respectively. Chance constrained programming is a competitive tool for solving optimization problems under uncertainty. Its main feature is that the resulting decision ensures the probability of complying with constraints, i.e. the probability of being feasible. Thus, using chance-constrained programming, the relationship between profitability and reliability can be quantified.

### 7.3.1 The Formulation of Chance Constraints

We first define additional notations $\alpha_{Cu}$ and $\alpha_F$ as confidence level of a solution satisfy the two constraints separately. The values of $\alpha_{Cu}$ and $\alpha_F$ more close to 1 indicate the more confidence of a solution match the constraints. Then, we present the chance constraints as follows:

$$P_r(g_p^{Cu} \geq Cu_P) \geq \alpha_{Cu}, \tag{7.13}$$

$$P_r(\mu^F g_p^F \leq R_p^F) \geq \alpha_F. \tag{7.14}$$

Constraints (7.13), (7.14) force the probability of guarantying the given bound are greater than or equal to the corresponding given threshold.

Similar to chance-constrained knapsack problem introduced in Section 4, we use Cantelli-Chebyshev inequality presented in Section 3.4.1 to construct the available surrogate that translates to a guarantee on the feasibility of the chance constraint imposed by the inequalities.

We assume the stochastic material grades are estimated with given expected values and corresponding variances, and the material grades of stockpiles are independent of each other. The material grade corresponds to expected value $a_s^o$ and standard deviation $\sigma_s^{om}$. Let $g_p^{Cu} = \sum_{s\in\mathcal{S}} x_{ps}\tilde{g}_{ps}^{Cu}$ be the Cu grade of parcel $p$ with a given solution $X = \{x_{p1}, .., x_{ps}, .., x_{p\mathbf{S}}\}$, and

$$E[g_p^{Cu}] = \sum_{s\in\mathcal{S}} x_{ps} E(\tilde{g}_{ps}^{Cu})$$

denotes the expected Cu grade of parcel $p$ of the solution derived by linearity of expectation,

$$Var[g_p^{Cu}] = \sum_{s\in\mathcal{S}} (x_{ps})^2 Var(\tilde{g}_{ps}^{Cu})$$

denotes the variance of Cu grade of parcel $p$. To match the expression of the Cantelli's inequality (3.10), we set

$$Cu_p = E[g_p^{Cu}] - \lambda\sqrt{Var[g_p^{Cu}]}$$

and have

$$\lambda = \frac{E[g_p^{Cu}] - Cu_p}{\sqrt{Var[g_p^{Cu}]}}$$

for each parcel, then we have a formulation to calculate the upper bound of the chance constraint (7.13) as follows:

$$P_r(g_p^{Cu} \le Cu_p) \le \frac{Var[g_p^{Cu}]}{Var[g_p^{Cu}] + (E[g_p^{Cu}] - Cu_p)^2} \le (1 - \alpha_{Cu}). \tag{7.15}$$

Furthermore, let $r_p^F = \mu^F \sum_{s\in\mathcal{S}} x_{ps}\tilde{g}_{ps}^F$ be the F recovery rate of parcel $p$. Let

$$E[r_p^F] = \mu^F \sum_{s\in\mathcal{S}} x_{ps} E(\tilde{g}_{ps}^F)$$

denotes the expectation of F recovery rate, and

$$Var[r_p^F] = \sum_{s\in\mathcal{S}} (\mu^F x_{ps})^2 Var(\tilde{g}_{ps}^F),$$

is the variance of F recovery rate of parcel $p$ with solution $X = \{x_{p1}, .., x_{ps}, .., x_{p\mathcal{S}}\}$.

To match the expression of the Cantelli's inequality (3.9), we set

$$R_p^F = \mu^F E[r_p^F] + \lambda\sqrt{Var[r_p^F]},$$

then have

$$\lambda = \frac{R_p^F - \mu^F E[g_p^F]}{\sqrt{Var[g_p^F]}}$$

for each parcel. We have a formulation to calculate the upper bound of the chance constraint (7.14) as follows:

$$P_r(\mu^F g_p^F \ge R_p^F) \le \frac{Var[g_p^F]}{Var[g_p^F] + (R_p^F - \mu^F E[g_p^F])^2} \le (1 - \alpha_F). \tag{7.16}$$

Now, we have obtained the surrogate functions of the chance constraints in the stockpile blending problem and we introduce a approach for solving the problem in the next section.

## 7.4 Approaches for the Stockpile Blending Problem with Chance Constraints

In this section, we investigate the DE algorithm introduced in Section 3.3.4, and we use $DE/current-to-best/1$ mutation strategy and uniform crossover operator. In the following subsections, we present the fitness functions that respond to different settings of the problem.

### 7.4.1 Fitness Function for Deterministic Setting

We start by designing a fitness function of the deterministic model. The fitness function $f$ takes all constraints into account and is similar to the fitness function (6.16) presented in Section 6.3. The fitness function of a solution $X$ is defined as follows:

$$f(X) = (u(X), v(X), w(X), q(X), g(X), z(X)) \tag{7.17}$$

$$\begin{aligned}
u(X) &= \sum_{p\in\mathcal{P}} \max\{|K_p - k_p|, 1\}\\
v(X) &= \max\{\sum_{p\in\mathcal{P}} t_p - D, 0\}\\
w(X) &= \min\{\sum_{p\in\mathcal{P}}\sum_{s\in\mathcal{S}} \theta_{ps}, 0\}\\
q(X) &= \sum_{p\in\mathcal{P}} \max\{Cu_p - g_p^{Cu}, 0\}\\
g(X) &= \sum_{p\in\mathcal{P}} \max\{r_p^F - R_p^F, 0\}\\
z(X) &= \sum_{p\in\mathcal{P}} c_p.
\end{aligned}$$

In this fitness function, the components $u, v, q, g$ need to be minimise while $w$ and $O$ need to be maximised, and we optimise $f$ in lexicographic order. For the stockpile blending problem, any infeasible solution can violate at least one of the above constraints. Then, among solutions that meet all constraints, we aim to maximise the objective function. Formally, we have

$$\begin{aligned}
& f(X) \succeq f(Y)\\
\Longleftrightarrow \text{ iff } & u(X) < u(Y) \text{ or}\\
& u(X) = u(Y) \wedge v(X) < v(Y) \text{ or}\\
& \{u, v\} \quad \text{are equal} \quad \wedge w(X) > w(Y) \text{ or}\\
& \{u, v, w\} \quad \text{are equal} \quad \wedge q(X) < q(Y) \text{ or}\\
& \{u, v, w, q\} \quad \text{are equal} \quad \wedge g(X) < g(Y) or\\
& \{u, v, w, q, g\} \quad \text{are equal} \quad \wedge z(X) > z(Y).
\end{aligned}$$

When comparing an infeasible solution and a feasible solution, the feasible solution is preferred. Between two infeasible solutions violating the same constraint, the solution with the lower degree of constraint violation is preferred.

### 7.4.2 Fitness Functions for Chance Constraints

Now, we design the fitness functions for the stockpile blending problem with chance constraints. We first reformulate the components $q$ and $g$ from the fitness function (7.17) by applying chance-constrained programming as follows:

$$q'(X) = \sum_{p\in\mathcal{P}} max\left\{P_r\{g_p^{Cu} \le Cu_p\} - (1 - \alpha_{Cu}), 0\right\} \tag{7.18}$$

$$g'(X) = \sum_{p\in\mathcal{P}} max\left\{P_r\{\mu^F g_p^F \ge R_p^F\} - (1 - \alpha_F), 0\right\}, \tag{7.19}$$

in which $q'$ and $g'$ need to be minimised.

To distinguish the repercussion of each chance constraint, we design three different fitness functions. Fitness function (7.20) only considers the chance constraint (7.13), and fitness function (7.21) only considers chance constraint (7.14), and function (7.22)

Table 7.2. General information about the ore and processing parameters

| Description | Values or Value range |
|---|---|
| Number of parcel | stockpiles; $\{3,4,5\}$ |
| Number of Stockpile | 7 |
| Duration of month | $29, 30, 31$ |
| Discount factor for time period ($\delta$) | 0.98 |
| Factor in chemical processing stage ($\tilde{\phi}$) | $[1000, 2000]$ |
| Factor of Au in chemical processing stage ($\phi^{Au}$) | $[200, 300]$ |
| Factor of U in chemical processing stage ($\phi^{U}$) | $[300, 400]$ |
| Factor of Fe in chemical processing stage ($\phi^{Fe}$) | $[560000, 570000]$ |
| Factor of Cu in chemical processing stage ($\phi^{Cu}$) | $[6000000, 7000000]$ |
| Factor of copper percentage within the produced copper concentrate ($\gamma_1, \gamma_2$) | $([5, 10], [30, 40])$ |
| Factor of F recovery ($\mu^F$) | $[0.05, 0.15]$ |
| Factor of U recovery ($\mu^U$) | $[0.5, 0.9]$ |
| Factor of copper recovery ($\mu_1^{Cu}, \mu_2^{Cu}$) | $([1.5, 3.5], [0, 10])$ |
| Tonnage of material hauled to stockpile | $[5000, 1000000]$ |
| Cu grade | $[0.05, 2.5]$ |
| Ag grade | $[1.0, 4.0]$ |
| Fe grade | $[10.0, 30.0]$ |
| Au grade | $[0.3, 2.0]$ |
| U grade | $[30.0, 400.0]$ |
| F grade | $[1200, 4500]$ |
| S grade | $[0.15, 1.0]$ |
| Expected tonne concentrate of parcel ($K_p$) | $[10000, \infty]$ |
| Threshold of F recovery of parcel ( $R_p^F$) | $[1300, 1500]$ |
| Threshold of Copper grade of parcel ($Cu_p$) | $[0.5, 1.5]$ |

considers two chance constraints together.

$$f'(X) = \big(u(X), v(X), w(X), q'(X), g(X), z(X)\big) \tag{7.20}$$
$$f''(X) = \big(u(X), v(X), w(X), q(X), g'(X), z(X)\big) \tag{7.21}$$
$$f'''(X) = \big(u(X), v(X), w(X), q'(X), g'(X), z(X)\big) \tag{7.22}$$

## 7.5 Experimental Investigation

In this section, we examine the solution quality associated with different fitness functions. As mentioned in the previous chapter, we are not able to compare the performance of the DE algorithm in a real-data instance. Therefore, we first design the instances used in this chapter. Then, we compare the results obtained by applying different fitness functions and examine the performance of the algorithm.

### 7.5.1 Experimental Setup

Table 7.2 lists the possible intervals of the tonnage of ore and material grades of the ore shipping from mine to stockpiles used for performance analysis and some of the process parameters.

We design three different instances by randomly chose the values of parameters, and those parameters are listed in Table 7.3, 7.4 and 7.5. In the instances, we assume that the material grades are chosen according to the Normal distribution, the expected values of grades are randomly selected from the value range, and the deviation of material grades are set equal to 0.01 times the expected value. Let $\alpha_{Cu} = \{0.999, 0.99, 0.9\}$ and $\alpha_F = \{0.999, 0.99, 0.9\}$. Base on this arrangement, we compare the performance of the DE algorithm with fitness functions (7.17), (7.20), (7.21) and (7.22) on the presented instances. We investigate the performance of the DE algorithms with different fitness functions and provide the results from 30 independent runs with 10000 generation and populationm size of 10 for all instances. For a closer look, we list the average, best and worst solutions obtained by the algorithm in corresponding columns. We

TABLE 7.3. Global parameters of instances

| | Instance 1 | Instance 2 | Instance 3 |
|---|---|---|---|
| Number of parcels | 3 | 3 | 4 |
| Number of stockpiles | 7 | 7 | 7 |
| Total duration | 30 | 28 | 31 |
| $\delta$ | 0.98 | 0.98 | 0.98 |
| $\tilde{\phi}$ | 1100 | 1100 | 1100 |
| $\phi^{Au}$ | 270 | 270 | 270 |
| $\phi^{U}$ | 340 | 340 | 340 |
| $\phi^{Fe}$ | 564000 | 564000 | 56400 |
| $\phi^{Cu}$ | 6050000 | 6050000 | 6050000 |
| $(\gamma_1, \gamma_2)$ | (7,36) | (7,36) | (7,36) |
| $\mu^{F}$ | 0.11 | 0.11 | 0.11 |
| $\mu^{U}$ | 0.79 | 0.79 | 0.79 |
| $(\mu_1^{Cu}, \mu_2^{Cu})$ | (2.5,0) | (2.5,0) | (2.5,0) |
| $R_p^F$ | 500 | 400 | 400 |
| $Cu_p$ | 0.9 | 1 | 1 |

TABLE 7.4. Ore shipping parameters of instances

| | Stockpiles | Tonnage of ore | Cu grade | Ag grade | Fe grade | Au grade | U grade | F grade | S grade |
|---|---|---|---|---|---|---|---|---|---|
| Instance 1 | 1 | 480000 | 1.08 | 1.33 | 14.08 | 1.56 | 32.73 | 1263 | 0.21 |
| | 2 | 220000 | 1.86 | 3.81 | 25.9 | 0.47 | 70.69 | 2568 | 0.8 |
| | 3 | 970000 | 1.79 | 3.41 | 28.97 | 0.5 | 127.83 | 4500 | 0.74 |
| | 4 | 400000 | 0.96 | 2.49 | 25 | 0.49 | 400 | 7500 | 0.8 |
| | 5 | 3550000 | 1.37 | 2.02 | 14.21 | 0.31 | 44 | 3000 | 0.5 |
| | 6 | 1130500 | 0.93 | 2.13 | 23.76 | 1.25 | 26.73 | 1560 | 0.26 |
| | 7 | 5377000 | 1.61 | 2.22 | 16.5 | 0.61 | 31 | 2780 | 0.15 |
| Instance 2 | 1 | 480000 | 1.78 | 1.33 | 14.08 | 1.56 | 32.73 | 1263 | 0.21 |
| | 2 | 220000 | 1.86 | 3.81 | 25.9 | 0.47 | 70.69 | 2568 | 0.8 |
| | 3 | 970000 | 1.79 | 3.41 | 28.97 | 0.5 | 127.83 | 4500 | 0.74 |
| | 4 | 400000 | 1.16 | 2.49 | 25 | 0.49 | 400 | 7500 | 0.8 |
| | 5 | 3550000 | 0.77 | 2.02 | 14.21 | 0.31 | 44 | 3000 | 0.5 |
| | 6 | 1130500 | 1.23 | 2.13 | 23.76 | 1.25 | 26.73 | 1560 | 0.26 |
| | 7 | 53770 | 1.81 | 2.22 | 16.5 | 0.61 | 31 | 2780 | 0.15 |
| Instance 3 | 1 | 5000000 | 1.58 | 1.33 | 14.08 | 1.56 | 32.73 | 1263 | 0.21 |
| | 2 | 4200000 | 1.86 | 3.81 | 25.9 | 0.47 | 70.69 | 2568 | 0.8 |
| | 3 | 9700000 | 1.79 | 3.41 | 28.97 | 0.5 | 127.83 | 4500 | 0.74 |
| | 4 | 4000000 | 1.16 | 2.49 | 25 | 0.49 | 400 | 7500 | 0.8 |
| | 5 | 3550000 | 1.37 | 2.02 | 14.21 | 0.31 | 44 | 3000 | 0.5 |
| | 6 | 1130500 | 1.13 | 2.13 | 23.76 | 1.25 | 26.73 | 1560 | 0.26 |
| | 7 | 5377000 | 1.91 | 2.22 | 16.5 | 0.61 | 31 | 2780 | 0.15 |

TABLE 7.5. Customer requirements of instances

| Parcels | Instances 1 | Instance 2 | Instance 3 |
|---|---|---|---|
| 1 | 750000 | 300000 | 137000 |
| 2 | 600000 | 460000 | 94000 |
| 3 | 420000 | 330000 | 92000 |
| 4 | | | 111000 |

TABLE 7.6. Fitness values obtained with single chance constraint

| | | Deterministic | Cu Chance constraint ($\alpha_{Cu}$) | | | F Chance constraint ($\alpha_F$) | | |
|---|---|---|---|---|---|---|---|---|
| Instance | | | 0.999 | 0.99 | 0.9 | 0.999 | 0.99 | 0.9 |
| 1 | Mean | 103603035.94 | 99319128.52 | 102724900.09 | 103206748.35 | 103117715.98 | 103340755.20 | 102753876.59 |
| | Best | 110830487.20 | 100434268.90 | 110489368.20 | 111221777.51 | 108460913.10 | 110593860.12 | 106976825.01 |
| | Worst | 100025173.10 | 98404158.16 | 99426947.62 | 99935646.96 | 99979453.63 | 98951340.30 | 99444063.80 |
| | Success rate | | 0.166666667 | 1 | 1 | 1 | 1 | 1 |
| 2 | Mean | 66339866.34 | 64280794.53 | 65346088.50 | 65440690.91 | 65741114.11 | 66128957.43 | 64999603.10 |
| | Best | 69691652.87 | 66062865.43 | 70504938.14 | 69307609.02 | 70846603.30 | 69265095.80 | 67416065.20 |
| | Worst | 63401302.85 | 62822049.78 | 61409848.38 | 62043167.43 | 62885220.60 | 63139531.31 | 62369471.32 |
| | Success rate | | 0.3 | 1 | 1 | 1 | 1 | 1 |
| 3 | Mean | 25706739.82 | 25172345.85 | 25484058.55 | 25667396.95 | 25414602.50 | 25487090.60 | 25737554.80 |
| | Best | 26714591.61 | 25780112.93 | 27501228.19 | 26652385.99 | 26542657.00 | 26440088.30 | 27420748.21 |
| | Worst | 25042412.92 | 24939884.30 | 24517912.31 | 24338065.15 | 24301347.20 | 24502516.90 | 24675079.20 |
| | Success rate | | 0.166666667 | 0.733333333 | 0.8 | 0.83333333 | 0.7 | 1 |

also evaluate the algorithm by success rate, which is the percentage of success for the algorithm in getting reasonable solutions out of 30 runs.

### 7.5.2 Experimental Results

We benchmark our approach with the combinations from the experimental setting described above. All experiments were performed using Java of version 11.0.1 and carried out on a MacBook with a 2.3GHz Intel Core i5 CPU.

Table 7.6 lists the results for the three instances with using fitness function (7.17, 7.20) and (7.21) separately, where they consider one of the chance constraints. Figure 7.1 shows the how the one of the chance-constrained bound $(\alpha_{Cu}, \alpha_F)$ affects the solution quality. The bars are corresponding to the solutions of instances combining with the probability of chance constraint, respectively, and the three bars in each group corresponding to the value of uncertainty $\{0.999, 0.99, 0.9\}$. Among others, we observe that results obtained by applying the fitness function (7.20) are significantly affected by the value of uncertainty denoted by $\alpha_{Cu}$. The results show an increasing trend as the value of $\alpha_{Cu}$ decreases. However, by observing the bars in *F chance constraint* group, the value of $\alpha_F$ does not affect the result when using the F chance constraint in the fitness function.

As can be seen from Table 7.6, the success rate shows significantly different results between using Cu chance constraint and F chance constraint, for instance 1 and 2. When the probability of Cu grade chance constraint is tight, such as 0.999, the DE algorithm can not generate a pure feasible population in the last generation. At the same time, the probability of F recovery chance constraint does not move the success rate of the algorithm. However, for instance, 3, which has four parcels into consideration and is the most complex instance in our study, the DE algorithm fails to obtain a feasible population in the last generation when the probability of F recovery chance constraint is higher than 0.999.

Table 7.7 lists the results obtained by considering two chance constraints, the fitness function (7.22). For each instance, we investigate different parameters setting together with the different requirements on the chance constraints determined by $\alpha_{Cu}$ and $\alpha_F$. The results list in the columns with the same $\alpha_{Cu}$ shows that there is no significant difference between the solutions obtained by applying difference $\alpha_F$. Moreover, with the same $\alpha_F$, the object value increase while the $\alpha_{Cu}$ decrease.

Now, we compare the results obtained by using different fitness functions of instances. However, there is no significant difference between the result obtained by using Cu grade chance constraint and combine chance constraints separately for the same instance with the same level of $\alpha_{Cu}$. By comparing the solutions list in the column *F Chance constraint* in Table 7.6 against that of the combined chance constraint in the same value of $\alpha_F$, we find that for the same instance, in most cases, the results obtained by a single chance constraint are better than the multi chance constraints.

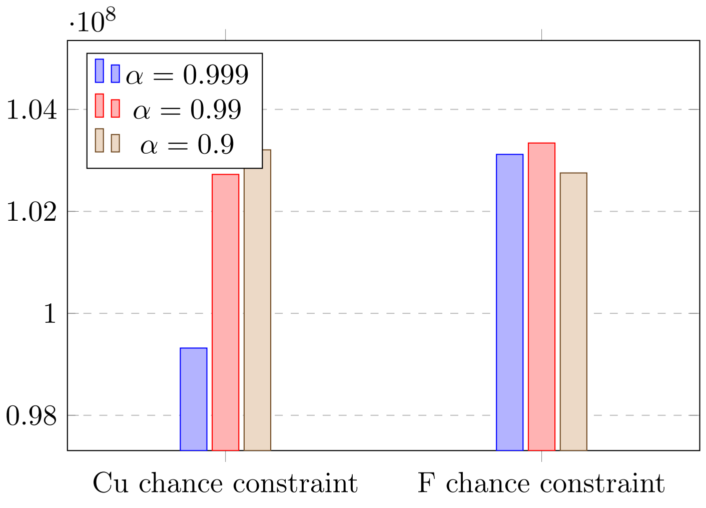
One interesting finding is that the value of $\alpha_F$ does not show significant effects on the results in the experiments. A possible explanation for this might be that the parameters of the instances are not reliable or match the real-world situation, which can indicate the malfunction of the constraint.

## 7.6 Conclusion

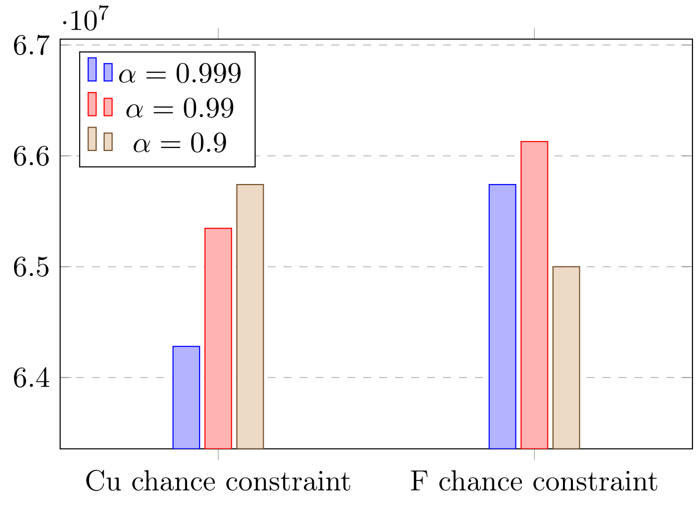
In this chapter, we studied the stockpile blending problem under uncertainty in the geologic data. We modelled the stockpile blending problem as a nonlinear optimization problem and introduced the chance constraints to tackle the uncertainty assuming that the material grades are stochastic variables. We showed how to incorporate the well-known probability tail, Chebyshev's inequality, into presenting the surrogate functions of the chance constraints. In addition, we designed fitness functions that responded to different chance constraints. In our experiments, which have covered various instances according to the parameters, we have observed that the probability of the Cu chance constraint affects the results obtained by using the fitness function only consider the Cu chance constraint and the fitness function with considering multi chance constraints. However, the threshold of the F chance constraint didn't show a significant influence on the results. It would be interesting to develop benchmarks for the stockpile blending problem with chance constraints as there are no available open access data-set. The study of stockpile blending problems in a stochastic environment is a potentially rich area for future research. A further study focusing on formulating the problem closer to the real-world situation is therefore suggested. For example, the distribution of material grads could be estimated by experimental and historical data, and some processes in the stockpile blending schedule could be more detailed instead of using fixed parameters.

TABLE 7.7. Fitness values obtained with two chance constraints

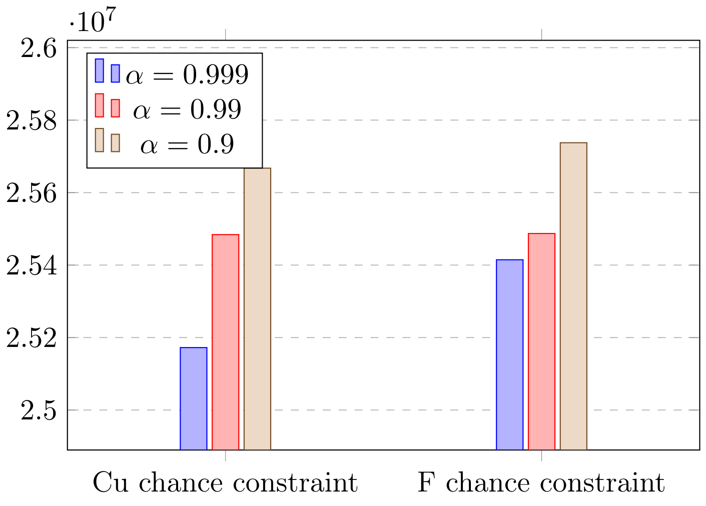
| Instance | | Combine Chance constraints | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\alpha_{Cu} = 0.999$ | | | $\alpha_{Cu} = 0.99$ | | | $\alpha_{Cu} = 0.9$ | | |
| $\alpha_F$ | | 0.999 | 0.99 | 0.9 | 0.999 | 0.99 | 0.9 | 0.999 | 0.99 | 0.9 |
| 1 | Mean | 98787701.39 | 99241024.07 | 99492631.19 | 102600300.50 | 102276579.70 | 102682519.61 | 103388918.00 | 102934747.10 | 102493177.23 |
| | Best | 102031085.80 | 103755489.91 | 102599092.62 | 106167319.20 | 107035409.13 | 109936137.61 | 106369047.00 | 108285976.13 | 108115669.08 |
| | Worst | 96585053.09 | 97115231.16 | 97595258.99 | 99322445.27 | 98370524.81 | 100035474.00 | 100191353.05 | 99131474.70 | 99124405.82 |
| | Success rate | 0.366666667 | 0.366666667 | 0.4 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | Mean | 65983255.56 | 63563990.36 | 64902632.44 | 65902068.31 | 65729892.86 | 65506695.73 | 65678975.14 | 65559860.50 | 65651773.33 |
| | Best | 69079010.41 | 65782360.39 | 69006540.86 | 69991791.36 | 69754992.24 | 70596782.19 | 69703718.21 | 68258615.00 | 70556638.81 |
| | Worst | 62870362.84 | 61877215.08 | 61963922.28 | 59672821.89 | 63038071.83 | 61823714.56 | 61900565.70 | 61618562.90 | 62959270.15 |
| | Success rate | 0.166666667 | 0.133333333 | 0.3 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | Mean | 25960917.39 | 25871636.05 | 25686417.52 | 25459522.87 | 25682180.87 | 25687541.01 | 25569238.21 | 25498602.00 | 25617280.63 |
| | Best | 25960917.39 | 26273134.25 | 25686417.52 | 26575050.89 | 26643851.74 | 26329733.68 | 26457335.14 | 26589310.80 | 26326709.05 |
| | Worst | 25960917.39 | 25473933.01 | 25686417.52 | 24495873.72 | 24826217.28 | 24668501.30 | 24693215.90 | 24721583.71 | 24549825.10 |
| | Success rate | 0.033333333 | 0.133333333 | 0.033333333 | 0.8 | 0.833333333 | 0.866666667 | 0.8 | 0.86666667 | 0.76666667 |



(a) Instance 1



(b) Instance 2



(c) Instance 3

FIGURE 7.1. Results obtained by the DE with using single chance constraint.

# Chapter 8

# Conclusion

Bio-inspired methods are general-purpose optimisation techniques that are inspired by nature. The ability of those algorithms is that they can obtain high-quality solutions for combinatorial optimisation problems without much knowledge about the search space. The present research aimed to examine the theoretical and practical understanding of bio-inspired algorithms dealing with chance-constrained optimisation problems and perform the chance-constrained optimisation in real-world applications.

The first aim of this thesis was to investigate the effects of evolutionary algorithms solving the CCKP and theoretical analyses of their behaviour when solving the CCKP. Chapter 4 has shown that incorporate well-known probability tail inequities into the search process of an evolutionary algorithm can avoid the difficulty of calculating the probability of chance constraints. The comparison between the performance of EAs and deterministic approaches on solving CCKP has shown that EAs are better than the proposed deterministic approaches according to the computation time for all instances. Two problem-specific operators and a new multi-objective model are presented in this chapter and show significant improvement in the performance of EAs. The insights gained from this chapter may assist in exploring the applications of EAs on combinatorial optimisation problems under uncertainties. Then, Chapter 5 has proven upper bounds for optimisation time of the $\text{RLS}_2$ and the (1+1) EA with two different types of chance-constrained knapsack problem with correlated weights. With rigorous runtime analysis of the algorithms on the CCKP with correlated weights, this chapter contributed to our understanding of bio-inspired algorithms in solving chance-constrained optimisation problems. Understanding the behaviour of bio-inspired algorithms is a long-time challenge to researchers and practitioners. A possible future work of this research aims to extend runtime analysis to more complex cases of the CCKP, understand basic working principles of more bio-inspired algorithms, and support them with practical results.

The second aim of this thesis was to adopt the chance-constrained optimisation on a real-world problem, the stockpile blending problem. In Chapter 6, the large-scale stockpile blending problem is formulated as a non-linear optimisation problem with continuous decision variables. For the complexity constraints, two repaired operators are presented to improve the efficiency of finding feasible solutions. Moreover, an approach that divided the large-scale problem into several one-month problems is introduced in this chapter, and a DE-based approach for a large-scale stockpile blending problem that optimises the problem month by month is used in this chapter. This approach guarantees the quality of each unit solution and the balance of used material between stockpiles. The results of this investigation show that the DE-based approach outperforms the methods used in real-world instances. This finding supports the idea that is using bio-inspired algorithms to solve complex, large-scale real-world problems. In Chapter 7, the chance-constrained optimisation is used to tackle the uncertainty of material from the resource in mines. Four fitness functions are presented containing different chance constraints to explore the effectiveness of each chance constraint on the objective. The experiment results give some insight into the quality of solutions based on different fitness functions. Furthermore, in terms of the complexity of the stockpile blending problem, only fundamental approaches are applied in this thesis on both the large-scale problem and the chance-constrained problem. It would be interesting to develop approaches that are able to solve the stockpile blending problems in different scenarios. As many real-world problems simultaneously have stochastic and dynamic properties, future research would be interesting to examine those chance-constrained optimisation problems under a dynamic environment.

# Bibliography

Ahmed, Shabbir and Alexander Shapiro (2008). “Solving chance-constrained stochastic programs via sampling and integer programming”. In: *State-of-the-art decision-making tools in the information-intensive age*. Informs, pp. 261–269.

Ahrens, Joachim H and Gerd Finke (1975). “Merging and sorting applied to the zero-one knapsack problem”. In: *Operations Research* 23.6, pp. 1099–1109.

Akaike, Atsushi and Kadri Dagdelen (1999). “A strategic production scheduling method for an open pit mine”. In: *proceedings of the 28th Application of Computers and Operation Research in the Mineral Industry*, pp. 729–738.

Asad, Mohammad Waqar Ali and Roussos Dimitrakopoulos (2012). “Optimal production scale of open pit mining operations with uncertain metal supply and long-term stockpiles”. In: *Resources policy* 37.1, pp. 81–89.

Assimi, Hirad, Oscar Harper, Yue Xie, Aneta Neumann, and Frank Neumann (2020). “Evolutionary Bi-Objective Optimization for the Dynamic Chance-Constrained Knapsack Problem Based on Tail Bound Objectives”. In: *ECAI*. Vol. 325. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 307–314.

Auger, Anne and Benjamin Doerr, eds. (2011). *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. Vol. 1. Series on Theoretical Computer Science. World Scientific.

Azzouz, Radhia, Slim Bechikh, and Lamjed Ben Said (2017). “Dynamic Multi-objective Optimization Using Evolutionary Algorithms: A Survey”. In: *Recent Advances in Evolutionary Multi-objective Optimization*. Vol. 20. Adaptation, Learning, and Optimization. Springer, pp. 31–70.

Baker, CK and SM Giacomo (1998). “Resource and reserves: their uses and abuses by the equity markets”. In: *Ore reserves and finance: a joint seminar between Australasian Institute of Mining and Metallurgy (AusIMM) and Australian Securities Exchange (ASX), Sydney*.

Beier, René and Berthold Vöcking (2004). “Random knapsack in expected polynomial time”. In: *J. Comput. Syst. Sci.* 69.3, pp. 306–329.

Bellman, Richard (1966). “Dynamic programming”. In: *Science* 153.3731, pp. 34–37.

Bhattacharya, U. K. (2009). “A chance constraints goal programming model for the advertising planning problem”. In: *Eur. J. Oper. Res.* 192.2, pp. 382–395. DOI: 10.1016/j.ejor.2007.09.039. URL: https://doi.org/10.1016/j.ejor.2007.09.039.

Bienaymé, Irénée-Jules (1853). *Considérations à l’appui de la découverte de Laplace sur la loi de probabilité dans la méthode des moindres carrés*. Imprimerie de Mallet-Bachelier.

Birge, John R and Francois Louveaux (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Bley, Andreas, Natashia Boland, Gary Froyland, and Mark Zuckerberg (2012). “Solving mixed integer nonlinear programming problems for mine production planning with stockpiling”. In: *Optimization Online ( http://www. optimization-online. org/DB _ HTML/2012/11/3674. html)* 28, p. 12.

Blom, Michelle, Adrian R. Pearce, and Peter J. Stuckey (2019). “Short-term planning for open pit mines: a review”. In: *International Journal of Mining, Reclamation and Environment* 33.5, pp. 318–339. DOI: 10.1080/17480930.2018.1448248. eprint: https://doi.org/10.1080/17480930.2018.1448248. URL: https://doi.org/10.1080/17480930.2018.1448248.

Calafiore, Giuseppe and Fabrizio Dabbene (2006). *Probabilistic and randomized methods for design under uncertainty.* Springer.

Calafiore, Giuseppe Carlo and Laurent El Ghaoui (2006). “On distributionally robust chance-constrained linear programs”. In: *Journal of Optimization Theory and Applications* 130.1, pp. 1–22.

Charnes, Abraham and William W Cooper (1959). “Chance-constrained programming”. In: *Management science* 6.1, pp. 73–79.

Chebyshev, Pafnutii Lvovich (1867). “Des valeurs moyennes”. In: *J. Math. Pures Appl* 12.2, pp. 177–184.

Cheng, Jianqiang, Michal Houda, and Abdel Lisser (2015). “Chance constrained 0–1 quadratic programs using copulas”. In: *Optimization Letters* 9.7, pp. 1283–1295.

Cheng, Jianqiang and Abdel Lisser (2012). “A second-order cone programming approach for linear programs with joint probabilistic constraints”. In: *Operations Research Letters* 40.5, pp. 325–328.

Cheng, Jianqiang and Abdel Lisser (2013). “A completely positive representation of 0–1 linear programs with joint probabilistic constraints”. In: *Operations Research Letters* 41.6, pp. 597–601.

Chernoff, Herman et al. (1952). “A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations”. In: *The Annals of Mathematical Statistics* 23.4, pp. 493–507.

Chiong, Raymond, Thomas Weise, and Zbigniew Michalewicz, eds. (2012). *Variants of Evolutionary Algorithms for Real-World Applications.* Springer.

Coello, Carlos Artemio Coello, David A. van Veldhuizen, and Gary B. Lamont (2002). *Evolutionary algorithms for solving multi-objective problems.* Vol. 5. Genetic algorithms and evolutionary computation. Kluwer.

Corder, Gregory W and Dale I Foreman (2014). *Nonparametric statistics: A step-by-step approach.* John Wiley & Sons.

Das, Swagatam, Sankha Subhra Mullick, and Ponnuthurai N Suganthan (2016). “Recent advances in differential evolution–an updated survey”. In: *Swarm and Evolutionary Computation* 27, pp. 1–30.

Das, Swagatam and Ponnuthurai Nagaratnam Suganthan (2010). “Differential evolution: A survey of the state-of-the-art”. In: *IEEE transactions on evolutionary computation* 15.1, pp. 4–31.

Deb, Kalyanmoy (2001). *Multi-objective optimization using evolutionary algorithms.* Wiley-Interscience series in systems and optimization. Wiley.

Deb, Kalyanmoy, Samir Agrawal, Amrit Pratap, and T. Meyarivan (2000). “A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimisation: NSGA-II”. In: *PPSN.* Vol. 1917. Lecture Notes in Computer Science. Springer, pp. 849–858.

Deb, Kalyanmoy, Samir Agrawal, Amrit Pratap, and T. Meyarivan (2002). “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Trans. Evol. Comput.* 6.2, pp. 182–197.

Dentcheva, Darinka, András Prékopa, and Andrzej Ruszczynski (2000). “Concavity and efficient points of discrete distributions in probabilistic programming”. In: *Mathematical programming* 89.1, pp. 55–77.

Doerr, Benjamin (2020). “Probabilistic Tools for the Analysis of Randomized Optimization Heuristics”. In: *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization.* Ed. by Benjamin Doerr and Frank Neumann. Cham: Springer International Publishing, pp. 1–87. ISBN: 978-3-030-29414-4. DOI: 10.1007/978-3-030-29414-4_1. URL: https://doi.org/10.1007/978-3-030-29414-4_1.

Doerr, Benjamin, Carola Doerr, Aneta Neumann, Frank Neumann, and Andrew M. Sutton (2020). “Optimization of Chance-Constrained Submodular Functions”. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020.* AAAI Press, pp. 1460–1467. ISBN: 978-1-57735-823-7.

Doerr, Benjamin, Daniel Johannsen, and Carola Winzen (2012). “Multiplicative Drift Analysis”. In: *Algorithmica* 64.4, pp. 673–697.

Doerr, Benjamin, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen (2017). “Fast genetic algorithms”. In: *GECCO.* ACM, pp. 777–784.

Driscoll, Wade C. (1996). “Robustness of the ANOVA and Tukey-Kramer statistical tests”. In: *International Conference on Computers and Industrial Engineering, CIE 1996.* Pergamon Press, Inc.

Droste, Stefan, Thomas Jansen, and Ingo Wegener (2002). “On the analysis of the (1+1) evolutionary algorithm”. In: *Theor. Comput. Sci.* 276.1-2, pp. 51–81.

Du, Dingzhu and Panos M Pardalos (1998). *Handbook of combinatorial optimization.* Vol. 4. Springer Science & Business Media.

Dunn, Jean and Olive Jean Dunn (1961). “Multiple comparisons among means”. In: *American Statistical Association*, pp. 52–64.

Fishburn, Peter C (1974). “Exceptional paper—Lexicographic orders, utilities and decision rules: A survey”. In: *Management science* 20.11, pp. 1442–1471.

Friedrich, Tobias, Timo Kötzing, J. A. Gregor Lagodzinski, Frank Neumann, and Martin Schirneck (2020). “Analysis of the (1+1) EA on subclasses of linear functions under uniform and linear constraints”. In: *Theor. Comput. Sci.* 832, pp. 3–19.

Ghasemi, Asghar and Saleh Zahediasl (2012). “Normality tests for statistical analysis: a guide for non-statisticians”. In: *International journal of endocrinology and metabolism* 10.2, p. 486.

Giel, Oliver (2003). “Expected runtimes of a simple multi-objective evolutionary algorithm”. In: *The 2003 Congress on Evolutionary Computation, 2003. CEC’03.* Vol. 3. IEEE, pp. 1918–1925.

Goel, Ashish and Piotr Indyk (1999). “Stochastic load balancing and related problems”. In: *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039).* IEEE, pp. 579–586.

Goyal, Vineet and R. Ravi (2010). “A PTAS for the chance-constrained knapsack problem with random item sizes”. In: *Oper. Res. Lett.* 38.3, pp. 161–164.

Greenberg, Harold and Robert L Hegerich (1970). “A branch search algorithm for the knapsack problem”. In: *Management Science* 16.5, pp. 327–332.

Hajela, Prabhat and C-Y Lin (1992). “Genetic search strategies in multicriterion optimal design”. In: *Structural optimization* 4.2, pp. 99–107.

Han, Kuk-Hyun and Jong-Hwan Kim (2002). “Quantum-inspired evolutionary algorithm for a class of combinatorial optimization”. In: *IEEE Trans. Evol. Comput.* 6.6, pp. 580–593. DOI: 10.1109/TEVC.2002.804320. URL: https://doi.org/10.1109/TEVC.2002.804320.

He, Jun, Boris Mitavskiy, and Yuren Zhou (2014). “A theoretical assessment of solution quality in evolutionary algorithms for the knapsack problem”. In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, July 6-11, 2014.* IEEE, pp. 141–148.

He, Jun and Xin Yao (2001). “Drift analysis and average time complexity of evolutionary algorithms”. In: *Artif. Intell.* 127.1, pp. 57–85.

Henrion, René (2007). “Structural properties of linear probabilistic constraints”. In: *Optimization* 56.4, pp. 425–440.

Hillier, Fredrick S. (1967). “Chance-constrained programming with 0-1 or bounded bontinuous decision variables”. In: *Management Science* 14.1, pp. 34–57.

Hoeffding, Wassily (1994). “Probability inequalities for sums of bounded random variables”. In: *The Collected Works of Wassily Hoeffding*. Springer, pp. 409–426.

Hoos, Holger H and Thomas Stützle (2004). *Stochastic local search: Foundations and applications*. Elsevier.

Horng, Shih Cheng, Shieh Shing Lin, and Feng Yi Yang (2012). “Evolutionary algorithm for stochastic job shop scheduling with random processing time”. In: *Expert Syst. Appl.* 39.3, pp. 3603–3610.

Ishibuchi, Hisao, Kaname Narukawa, Noritaka Tsukamoto, and Yusuke Nojima (2008). “An empirical study on similarity-based mating for evolutionary multiobjective combinatorial optimization”. In: *Eur. J. Oper. Res.* 188.1, pp. 57–75.

Jagannathan, R. (1974). “Chance-Constrained Programming with Joint Constraints”. In: *Oper. Res.* 22.2, pp. 358–372. DOI: 10.1287/opre.22.2.358. URL: https://doi.org/10.1287/opre.22.2.358.

Jaszkiewicz, Andrzej (2001). *Multiple objective metaheuristic algorithms for combinatorial optimization*. Wydawnictwo Politechniki Poznanskiej.

Johannsen, Daniel (2011). “Evolutionary Computation in Combinatorial Optimization”. In: *Theory of Randomized Search Heuristics*. Vol. 1. Series on Theoretical Computer Science. World Scientific, pp. 53–99.

Johnson, Thys B (1968). *Optimum open pit mine production scheduling*. Tech. rep. California Univ Berkeley Operations Research Center.

Jupp, K, TJ Howard, and JE Everett (2013). “Role of pre-crusher stockpiling for grade control in iron ore mining”. In: *Applied Earth Science* 122.4, pp. 242–255.

Kaelo, P. and M. M. Ali (2006). “A numerical study of some modified differential evolution algorithms”. In: *Eur. J. Oper. Res.* 169.3, pp. 1176–1184.

Kall, Peter, Stein W Wallace, and Peter Kall (1994). *Stochastic programming*. Springer.

Kellerer, Hans, Ulrich Pferschy, and David Pisinger (2004). *Knapsack problems*. Springer.

Kleinberg, Jon, Yuval Rabani, and Éva Tardos (1997). “Allocating bandwidth for bursty connections”. In: *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, STOC 1997*. ACM.

Klopfenstein, Olivier and Dritan Nace (2008). “A robust approach to the chance-constrained knapsack problem”. In: *Operations Research Letters*.

Kolesar, Peter J (1967). “A branch and bound algorithm for the knapsack problem”. In: *Management science* 13.9, pp. 723–735.

Koza, John R. (1993). *Genetic programming - on the programming of computers by means of natural selection*. Complex adaptive systems. MIT Press.

Kroese, Dirk P, Tim Brereton, Thomas Taimre, and Zdravko I Botev (2014). “Why the Monte Carlo method is so important today”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 6.6, pp. 386–392.

Lagoa, Constantino Manuel (1999). “On the convexity of probabilistically constrained linear programs”. In: *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*. Vol. 1. IEEE, pp. 516–521.

Lamghari, Amina and Roussos G. Dimitrakopoulos (2012). “A diversified Tabu search approach for the open-pit mine production scheduling problem with metal uncertainty”. In: *Eur. J. Oper. Res.* 222.3, pp. 642–652. URL: https://doi.org/10.1016/j.ejor.2012.05.029.

Lehre, Per Kristian and Carsten Witt (2013). “General Drift Analysis with Tail Bounds”. In: *CoRR* abs/1307.2559.

Lipovetzky, Nir, Christina N. Burt, Adrian R. Pearce, and Peter J. Stuckey (2014). “Planning for Mining Operations with Time and Resource Constraints”. In: *ICAPS*. AAAI.

Lissovoi, Andrei and Carsten Witt (2017). “A Runtime Analysis of Parallel Evolutionary Algorithms in Dynamic Optimization”. In: *Algorithmica* 78.2, pp. 641–659.

Liu, Baoding (2007). “Uncertainty theory”. In: *Uncertainty theory*. Springer, pp. 205–234.

Liu, Bo, Qingfu Zhang, Francisco V. Fernández, and Georges G. E. Gielen (2013). “An efficient evolutionary algorithm for chance-constrained bi-objective stochastic optimization”. In: *IEEE Trans. Evolutionary Computation* 17.6, pp. 786–796.

Luedtke, James, Shabbir Ahmed, and George L Nemhauser (2010). “An integer programming approach for linear programs with probabilistic constraints”. In: *Mathematical programming* 122.2, pp. 247–272.

Luo, Xiaolin and Pavel V Shevchenko (2009). “Computing tails of compound distributions using direct numerical integration”. In: *arXiv preprint arXiv:0904.0830*.

Marshall, Albert W, Ingram Olkin, and Barry C Arnold (1979). *Inequalities: theory of majorization and its applications*. Vol. 143. Springer.

Martello, Silvano, David Pisinger, and Paolo Toth (1999). “Dynamic programming and strong bounds for the 0-1 knapsack problem”. In: *Management science* 45.3, pp. 414–424.

Martello, Silvano and Paolo Toth (1997). “Upper Bounds and Algorithms for Hard 0-1 Knapsack Problems”. In: *Oper. Res.* 45.5, pp. 768–778.

Martello, Silvano and Paolo Toth (2003). “An Exact Algorithm for the Two-Constraint 0 - 1 Knapsack Problem”. In: *Oper. Res.* 51.5, pp. 826–835.

McConaghy, T., P. Palmers, M. Steyaert, and G. G. E. Gielen (2009). “Variation-aware structural synthesis of analog circuits via hierarchical building blocks and structural homotopy”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28.9, pp. 1281–1294.

Mercado, Lei L, S-M Kuo, Tien-Yu Lee, and Russell Lee (2005). “Analysis of RF MEMS switch packaging process for yield improvement”. In: *IEEE transactions on advanced packaging* 28.1, pp. 134–141.

Miller, Bruce L and Harvey M Wagner (1965). “Chance constrained programming with joint constraints”. In: *Operations Research* 13.6, pp. 930–945.

Moreno, Eduardo, Mojtaba Rezakhah, Alexandra M. Newman, and Felipe Ferreira (2017). “Linear models for stockpiling in open-pit mine production scheduling problems”. In: *Eur. J. Oper. Res.* 260.1, pp. 212–221.

Motwani, Rajeev and Prabhakar Raghavan (1995). *Randomized Algorithms*. Cambridge University Press.

Mühlenbein, Heinz (1992). “How Genetic Algorithms Really Work: Mutation and Hillclimbing”. In: *PPSN*. Elsevier, pp. 15–26.

Murata, Tadahiko and Hisao Ishibuchi (1995). “MOGA: Multi-objective genetic algorithms”. In: *IEEE international conference on evolutionary computation*. Vol. 1, pp. 289–294.

Nemhauser, G. L. and Z. Ullmann (1969). “Discrete Dynamic Programming and Capital Allocation”. In: *Management Science* 15.9, pp. 494–505. ISSN: 00251909, 15265501. URL: `http://www.jstor.org/stable/2628385`.

Neri, Ferrante and Ville Tirronen (2010). “Recent advances in differential evolution: a survey and experimental analysis”. In: *Artif. Intell. Rev.* 33.1-2, pp. 61–106.

Neumann, Aneta and Frank Neumann (2020). “Optimising Monotone Chance-Constrained Submodular Functions Using Evolutionary Multi-objective Algorithms”. In: *Parallel Problem Solving from Nature - PPSN XVI - 16th International Conference, PPSN 2020, Proceedings, Part I*, pp. 404–417.

Neumann, Frank, Mojgan Pourhassan, and Vahid Roostapour (2020). “Analysis of Evolutionary Algorithms in Dynamic and Stochastic Environments”. In: *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Ed. by Benjamin Doerr and Frank Neumann. Cham: Springer International Publishing, pp. 323–357. ISBN: 978-3-030-29414-4.

Neumann, Frank and Andrew M. Sutton (2019). “Runtime analysis of the (1 + 1) evolutionary algorithm for the chance-constrained knapsack problem”. In: *FOGA*. ACM, pp. 147–153.

Neumann, Frank and Ingo Wegener (2007). “Randomized local search, evolutionary algorithms, and the minimum spanning tree problem”. In: *Theor. Comput. Sci.* 378.1, pp. 32–40.

Neumann, Frank and Carsten Witt (2006). “Runtime Analysis of a Simple Ant Colony Optimization Algorithm”. In: *Theory of Evolutionary Algorithms*. Vol. 06061. Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany.

Neumann, Frank and Carsten Witt (2010). *Bioinspired Computation in Combinatorial Optimization*. Natural Computing Series. Springer.

Nguyen, Trung Thanh and Xin Yao (2012). “Continuous dynamic constrained optimization - the challenges”. In: *IEEE Transactions on Evolutionary Computation* 16.6, pp. 769–786.

Oliveto, Peter S. and Xin Yao (2011). “Runtime Analysis of Evolutionary Algorithms for Discrete Optimization”. In: *Theory of Randomized Search Heuristics*. Vol. 1. Series on Theoretical Computer Science. World Scientific, pp. 21–52.

Oliveto, Pietro S., Jun He, and Xin Yao (2007). “Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results”. In: *Int. J. Autom. Comput.* 4.3, pp. 281–293.

Ordoudis, Christos, Viet Anh Nguyen, Daniel Kuhn, and Pierre Pinson (2021). “Energy and reserve dispatch with distributionally robust joint chance constraints”. In: *Operations Research Letters* 49.3, pp. 291–299. ISSN: 0167-6377. DOI: https://doi.org/10.1016/j.orl.2021.01.012. URL: https://www.sciencedirect.com/science/article/pii/S0167637721000213.

Osanloo, M., J. Gholamnejad, and B. Karimi (2008). “Long-term open pit mine production planning: a review of models and algorithms”. In: *International Journal of Mining, Reclamation and Environment* 22.1, pp. 3–35. DOI: 10.1080/17480930601118947. URL: https://doi.org/10.1080/17480930601118947.

Papadimitriou, Christos H. and Kenneth Steiglitz (1982). *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall.

Pham, Nam, Aleksander Malinowski, and Tomasz Bartczak (2011). “Comparative Study of Derivative Free Optimization Algorithms”. In: *IEEE Trans. Ind. Informatics* 7.4, pp. 592–600. DOI: 10.1109/TII.2011.2166799. URL: https://doi.org/10.1109/TII.2011.2166799.

Pinter, Janos (1989). “Deterministic approximations of probability inequalities”. In: *Zeitschrift für Operations-Research* 33.4, pp. 219–239.

Poojari, Chandra A. and Boby Varghese (2008). “Genetic Algorithm based technique for solving Chance Constrained Problems”. In: *Eur. J. Oper. Res.* 185.3, pp. 1128–1154.

Prékopa, András (1971). “Logarithmic concave measures with application to stochastic programming”. In: *Acta Scientiarum Mathematicarum* 32, pp. 301–316.

Prékopa, András (1990). “Dual method for the solution of a one-stage stochastic programming problem with random RHS obeying a discrete probability distribution”. In: *Zeitschrift für Operations Research* 34.6, pp. 441–461.

Prékopa, András (1995). “Programming under probabilistic constraint and maximizing probabilities under constraints”. In: *Stochastic Programming*. Springer, pp. 319–371.

Prékopa, András (2003). “Probabilistic programming”. In: *Handbooks in operations research and management science* 10, pp. 267–351.

Prékopa, András, Kunikazu Yoda, and Munevver Mine Subasi (2011). “Uniform quasi-concavity in probabilistic constrained stochastic programming”. In: *Operations Research Letters* 39.3, pp. 188–192.

Price, Kenneth, Rainer M Storn, and Jouni A Lampinen (2006). *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media.

Price, Kenneth V (1996). “Differential evolution: a fast and simple numerical optimizer”. In: *Proceedings of North American Fuzzy Information Processing*. IEEE, pp. 524–527.

Raghavan, Prabhakar and Rajeev Motwani (1995). *Randomized algorithms*. Cambridge University Press Cambridge.

Rakshit, Pratyusha, Amit Konar, and Swagatam Das (2017). “Noisy evolutionary optimization algorithms - A comprehensive survey”. In: *Swarm Evol. Comput.* 33, pp. 18–45.

Ravichandran, Adhithya, Shahin Sirouspour, Pawel Malysz, and Ali Emadi (2018). “A Chance-Constraints-Based Control Strategy for Microgrids With Energy Storage and Integrated Electric Vehicles”. In: *IEEE Trans. Smart Grid* 9.1, pp. 346–359. DOI: `10.1109/TSG.2016.2552173`. URL: `https://doi.org/10.1109/TSG.2016.2552173`.

Raychaudhuri, Samik (2008). “Introduction to monte carlo simulation”. In: *2008 Winter simulation conference*. IEEE, pp. 91–100.

Rezakhah, Mojtaba and Eduardo Moreno (2019). “Open Pit Mine Scheduling Model Considering Blending and Stockpiling”. In: *International Symposium on Mine Planning & Equipment Selection*. Springer, pp. 75–82.

Riff, María Cristina, Teddy Alfaro, Xavier Bonnaire, and Carlos Grandón (2008). “EA-MP: An evolutionary algorithm for a mine planning problem”. In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008, June 1-6, 2008, Hong Kong, China*. IEEE, pp. 4011–4014. DOI: `10.1109/CEC.2008.4631344`. URL: `https://doi.org/10.1109/CEC.2008.4631344`.

Robinson, GK (2004). “How much would a blending stockpile reduce variation?” In: *Chemometrics and intelligent laboratory systems* 74.1, pp. 121–133.

Roostapour, Vahid, Aneta Neumann, and Frank Neumann (2018). “On the performance of baseline evolutionary algorithms on the dynamic knapsack problem”. In: *International Conference on Parallel Problem Solving from Nature - PPSN XV*. Lecture Notes in Computer Science. Springer, Cham, pp. 158–169.

Roostapour, Vahid, Aneta Neumann, Frank Neumann, and Tobias Friedrich (2019). “Pareto Optimization for Subset Selection with Dynamic Cost Constraints”. In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*. AAAI Press, pp. 2354–2361.

Ruszczyński, Andrzej (2002). “Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra”. In: *Mathematical Programming* 93.2, pp. 195–215.

Samavati, Mehran, Daryl Essam, Micah Nehring, and Ruhul A. Sarker (2017). “A local branching heuristic for the open pit mine production scheduling problem”. In: *Eur. J. Oper. Res.* 257.1, pp. 261–271.

Sasaki, Galen H. and Bruce E. Hajek (1988). “The time complexity of maximum matching by simulated annealing”. In: *J. ACM* 35.2, pp. 387–403.

Schaffer, J. David (1985). “Multiple Objective Optimization with Vector Evaluated Genetic Algorithms”. In: *ICGA*. Lawrence Erlbaum Associates, pp. 93–100.

Segura, Carlos, Salvador Botello Rionda, Arturo Hernández Aguirre, and Sergio Ivvan Valdez Peña (2015). “A Novel Diversity-based Evolutionary Algorithm for the Traveling Salesman Problem”. In: *GECCO*. ACM, pp. 489–496.

Shambour, Moh’d Khaled Yousef (2019). “Adaptive multi-crossover evolutionary algorithm for real-world optimisation problems”. In: *Int. J. Reason. based Intell. Syst.* 11.1, pp. 1–10. DOI: 10.1504/IJRIS.2019.098058. URL: https://doi.org/10.1504/IJRIS.2019.098058.

Shapiro, Alexander, Darinka Dentcheva, and Andrzej Ruszczyński (2014). *Lectures on stochastic programming: modeling and theory*. SIAM.

Sim, Kevin and Paul Kaufmann, eds. (2018). *Applications of Evolutionary Computation - 21st International Conference, EvoApplications 2018, Parma, Italy, April 4-6, 2018, Proceedings*. Vol. 10784. Lecture Notes in Computer Science. Springer. ISBN: 978-3-319-77537-1. DOI: 10.1007/978-3-319-77538-8. URL: https://doi.org/10.1007/978-3-319-77538-8.

Sotoudeh, Farzad, Micah Nehring, Mehmet Kizil, Peter Knights, and Amin Mousavi (2020). “Production scheduling optimisation for sublevel stoping mines using mathematical programming: A review of literature and future directions”. In: *Resources Policy* 68, p. 101809.

Srinivas, N. and Kalyanmoy Deb (1994). “Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms”. In: *Evol. Comput.* 2.3, pp. 221–248.

Storn, Rainer and V Kenneth (1995). “Price, K.: Differential Evolution–A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces”. In: *International Computer Science Institute, Report no. TR-95-012, Berkeley*.

Storn, Rainer and Kenneth Price (1997). “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces”. In: *Journal of global optimization* 11.4, pp. 341–359.

Till, Jochen, Guido Sand, Maren Urselmann, and Sebastian Engell (2007). “A hybrid evolutionary algorithm for solving two-stage stochastic integer programs in chemical batch scheduling”. In: *Computers & Chemical Engineering* 31.5-6, pp. 630–647.

Topal, E. and S. Ramazan (2012). “Strategic mine planning model using network flow model and real case application”. In: *International Journal of Mining, Reclamation and Environment* 26.1, pp. 29–37. DOI: 10.1080/17480930.2011.600827. URL: https://doi.org/10.1080/17480930.2011.600827.

Toth, P. (1980). “Dynamic programming algorithms for the Zero-One Knapsack Problem”. In: *Computing* 25.1, pp. 29–45.

Toth, Paolo and Silvano Martello (1990). *Knapsack problems: Algorithms and computer implementations*. Wiley Chichester, UK.

Uryasev, Stanislav (2013). *Probabilistic constrained optimization: methodology and applications*. Vol. 49. Springer Science & Business Media.

Vance, Pamela H. (1993). “Knapsack Problems: Algorithms and Computer Implementations”. In: *SIAM Review* 35.4, pp. 684–685.

Wang, Yu, Jiafu Tang, and Richard YK Fung (2014). "A column-generation-based heuristic algorithm for solving operating theater planning problem under stochastic demand and surgery cancellation risk". In: *International Journal of Production Economics* 158, pp. 28–36.

Watson, Ray and Ian Gordon (1986). "On quantiles of sums". In: *Australian Journal of Statistics* 28.2, pp. 192–199.

Wegener, Ingo (2002). "Methods for the Analysis of Evolutionary Algorithms on Pseudo-Boolean Functions". In: *Evolutionary Optimization*. Boston, MA: Springer US, pp. 349–369. ISBN: 978-0-306-48041-6. DOI: 10.1007/0-306-48041-7_14. URL: https://doi.org/10.1007/0-306-48041-7_14.

Xie, Yue, Oscar Harper, Hirad Assimi, Aneta Neumann, and Frank Neumann (2019). "Evolutionary algorithms for the chance-constrained knapsack problem". In: *GECCO*. ACM, pp. 338–346.

Xie, Yue, Aneta Neumann, and Frank Neumann (2020). "Specific single- and multi-objective evolutionary algorithms for the chance-constrained knapsack problem". In: *GECCO*. ACM, pp. 271–279.

Xie, Yue, Aneta Neumann, and Frank Neumann (2021a). "Heuristic Strategies for Solving Complex Interacting Large-Scale Stockpile Blending Problems". In: *CEC*. IEEE, pp. 1288–1295.

Xie, Yue, Aneta Neumann, and Frank Neumann (2021b). "Heuristic strategies for solving complex interacting stockpile blending problem with chance constraints". In: *GECCO*. ACM, pp. 1079–1087.

Xie, Yue, Aneta Neumann, Frank Neumann, and Andrew M. Sutton (2021). "Runtime analysis of RLS and the (1+1) EA for the chance-constrained knapsack problem with correlated uniform weights". In: *GECCO*. ACM, pp. 1187–1194.