Feature-Based Selection of Bio-Inspired Algorithms for Constrained Continuous Optimisation

Shayan Poursoltan

A thesis submitted for the degree of

Doctor of Philosophy (Ph.D.)

School of Computer Science
Faculty of Engineering, Computer, and Mathematical Sciences

The University of Adelaide

March 2016

THE UNIVERSITY OF ADELAIDE

Abstract

Engineering, Computer and Mathematical Sciences
School of Computer Science

Doctor of Philosophy

Feature-Based Selection of Bio-Inspired Algorithms for Constrained Continuous Optimisation

Shayan Poursoltan

Constrained continuous optimisation problems are widespread in the real-world and often very complex. Bio-inspired algorithms such as evolutionary algorithms (EAs) or particle swarm optimisation (PSO) algorithms have been successful in solving these problems. Recently, there has been an increasing interest in understanding the features of problems that make them hard to solve. These studies have been carried out for discrete and unconstrained continuous optimisation problems, to find the relationship between problem features and algorithm performance.

To study the connection between algorithms and constrained optimisation problems (COPs), more practical perspectives of problem features analysis and their relations to algorithms are essential. Thus, this thesis contributes to the understanding of constrained optimisation problems and their constraint features that make them hard to solve by algorithms. We introduce an empirical feature-based analysis for COPs and bio-inspired algorithms. Furthermore, the relationships between the constraint features of given COPs and algorithms are studied here. By linking the features of the constraints and different bio-inspired algorithms, we design a new model for predicting the algorithm performance for COPs based on their constraint features. In this thesis, we present a novel approach to analyse constrained continuous optimisation problems based on their constraint features. Furthermore we use this knowledge to implement an automated feature-based algorithm selection model for constrained continuous optimisation.

Declaration

Signature

Date: 7/3/2016

Acknowledgements

I would like to acknowledge the following people who helped me through the various stages of this PhD:

- My supervisor, Professor Frank Neumann, for his sharp insights, clear guidance and his all time support. Without his support, no doubt I could not finish this chapter of my life. I am grateful to him for holding me to a high research standard and enforcing strict validations for each research result, and thus teaching me how to do research.

- My co-supervisor, Professor Zbigniew Michalewicz, for his encouragement and practical advice. I am also thankful to him for encouraging me to start my PhD studies in the optimisation field.

- The optimisation and logistic research group with the countless benefits that come from their brainstorming sessions

- My parents, for all the moral support and the amazing chances they have given me over these years. Without them, I could not be the person I am today. In addition I want to thank my brother for sharing the highs and lows of my and his PhD.

- To all managers and staff in Bradford college and University of Adelaide to give me the chance of teaching and improving my confidence.

- All my teachers from the elementary school to the university for cherishing my intellect, passion and discipline that lead to a PhD,

- My friends, for the memorable moments they shared with me, helping me to forget the stressful life even for a second.

Statements of Authorships

This thesis is based on one book chapter and four conference papers. The book chapter and three conference papers are already been published. One conference paper is accepted for publication. I have provided a statement of authorship for each of these articles to certify that I was actively involved in the process of preparing each article.

The following is the list of all publications included in this thesis. For three of the conference papers the extended versions are included in this thesis.

References

[1] S. Poursoltan and F. Neumann, "Ruggedness quantifying for constrained continuous fitness landscapes," in *Evolutionary Constrained Optimisation*. Springer, 2015, pp. 29–50

[2] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of linear constraints for ε-constrained differential evolution. In *Evolutionary Computation (CEC), 2014 IEEE Congress on, Beijing 12-14 July*, pages 3088–3095. IEEE, 2014

[3] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of set of constraints for ε-constrained differential evolution. In S. Arik, T. Huang, W. K. Lai, and Q. Liu, editors, *Neural Information Processing*, volume 9491 of *Lecture Notes in Computer Science*, pages 344–355. Springer International Publishing, 2015

[4] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of set of constraints for e-constrained differential evolution. *CoRR*, abs/1506.06848, 2015. Extended version of [3]

[5] S. Poursoltan and F. Neumann. A feature-based comparison of evolutionary computing techniques for constrained continuous optimisation.

In S. Arik, T. Huang, W. K. Lai, and Q. Liu, editors, *Neural Information Processing*, volume 9491 of *Lecture Notes in Computer Science*, pages 332–343. Springer International Publishing, 2015

[6] S. Poursoltan and F. Neumann. A feature-based comparison of evolutionary computing techniques for constrained continuous optimisation. *CoRR*, abs/1509.06842, 2015. Extended version of [5]

[7] S. Poursoltan and F. Neumann. A feature-based prediction model of algorithm selection for constrained continuous optimisation. Accepted at The IEEE Congress on Evolutionary Computation 2016 (IEEE CEC 2016)

[8] S. Poursoltan and F. Neumann. A feature-based prediction model of algorithm selection for constrained continuous optimisation. *CoRR*, abs/1602.02862, 2016. Extended version of [7]

Contents

Chapter 1

Introduction

In many ways we require optimality in our lives. For instance, we seek the best solution with minimum cost. Optimisation is the notation that is given to this optimality. Examples of this are designing aircraft wings by minimising drafts, finding a set of trading rules which maximises the total profit or locating an optimised route between networks that minimises the traffic. In other words, what are called the variables of problem, maximise or minimise the problem's objective. Many optimisation problems in science and engineering involve constraints which reduce their search space feasibility and increase the searching process complexity. These problems are called constrained optimisation problems (COPs). In solving constrained optimisation problems, solutions that satisfy all the constraints are considered as feasible individuals, while the ones that fail to satisfy any of the constraints are infeasible ones.

There are many real-world examples of constrained continuous optimisation problems. Such problems are widespread in the mathematical modelling of real-world systems for a very broad range of applications. The applications include engineering and numerical design, VLSI chip design, database problems, chemical engineering design and control, fixed charges, economies of scale, location problems, quadratic assignment and maximising profit within trading rules [6, 11]. Thus, constrained continuous optimisation problems are an intrinsic part of real-world applications.

1.1 Constrained Continuous Optimisation Problem

A constrained optimisation problem in a numerical continuous space is an optimisation problem in which the value of the variables should satisfy some equalities/inequalities, while they minimise the value of an objective function. In general the COP can be formulated as follows:

$$\begin{aligned} \text{minimise} \quad & f(x), \quad x = (x_1, \ldots, x_n) \in \mathbb{R}^n \\ \text{subject to} \quad & g_i(x) \leq 0 \quad \forall i \in \{1, \ldots, q\} \\ & h_j(x) = 0 \quad \forall j \in \{q+1, \ldots, p\} \end{aligned}$$

where $x = (x_1, x_2, \ldots, x_n)$ is an n dimensional vector and $x \in S \cap F$. Also $g_i(x)$ and $h_j(x)$ are inequality and equality constraints, q and $r = p - q$ are the numbers of them respectively. Also, the feasible region $F \subseteq S$ of the search space S is defined by

$$l_i \leq x_i \leq u_i, \qquad 1 \leq i \leq n$$

where both l_i and u_i denote lower and upper boundaries for the ith variable and $1 \leq i \leq n$ respectively.

To handle equality constraints, they are usually transformed into inequality constraints as:

$$|h_j(x)| \leq \varepsilon \quad \text{for} \;\; j = q+1 \;\; \text{to} \;\; p$$

where ε is a small positive value. In all experiments carried out in this thesis the value of ε is as 1E-4, which is the standard setting in the CEC 2010 benchmark competition for constrained real-valued optimisation problems [20].

1.2 Evolutionary Algorithms for Solving Constrained Optimisation Problems

In order to solve these constrained continuous optimisation problems, many algorithms have been proposed. The majority of COPs cannot be solved using traditional mathematical methods, such as gradient-based techniques. For instance, in multi-modal problems, those with non-differentiable search spaces or black-box optimisation problems, gradient information is not sufficient to solve problems. In order to find solutions for these problem types, meta-heuristic methods that find approximate solutions are used. During recent years, bio-inspired population-based stochastic evolutionary algorithms (EAs) have been introduced to solve non-linear complex problems. For a general review of evolutionary algorithms, we refer the reader to [5]. Real-world optimisation problems mostly have real-valued variables and there are many algorithms proposed to search their continuous search space, such as Differential Evolution (DE) [32], Particle Swarm Optimisation (PSO) [16] and Evolutionary Strategies (ES) [27].

Differential evolution is arguably one of the most powerful and popular stochastic real-parameter optimisation algorithms. It is a parallel direct search method that fulfils the following optimisation practice requirements: the ability to handle non-linear, non-differentiable and multi-modal objective functions, optimising in parallel to deal with high computational cost functions, ease of use and acceptable convergence properties. The initial population is chosen randomly, while covering the entire parameter space. The differential evolution generates new parameter vectors by adding the weighted difference between two population vectors and the third one (x_p, x_q and x_r). This operation is called a mutation. Next, the parameters of the mutated vector is mixed with the parameters of another predetermined vector (the target vector) to create trial vector. This process is called crossover. For more details of varieties of crossovers we refer the reader to [32]. If the trial vector has a lower fitness value than the target vector (for minimising), the trial vector replaces the target vector. To decide whether or not it should become a member of the next generation ($g+1$), the generated trial vector is compared with the target vector, using greedy criteria and the better vector is chosen for the next generation. Algorithm 1 shows the pseudo-code for differential evolution with the differential weight F, crossover probability C_r and generation number g. Also, d denotes the algorithm population size.

In the ES algorithm (($\mu \overset{+}{,} \lambda$)-ES), at a very abstract level, the evolution can be considered as a process of selecting states where the selection is based on their fitness value. The original formulation is the application of mutation and selection in the population of the new candidate solution. μ stands for the number of parents appearing in population. Also, λ denotes the number of all the offspring created by these parents within a generation. In (μ + λ)-ES, the parents and offspring are united before μ fittest individuals are selected. But, in (μ , λ)-ES, new individuals are only chosen from λ offspring. Within one ES generation step, λ offspring individuals are generated from the set of μ parent individuals. Then, based on the selection process, the fittest individuals are selected for next generation. We refer the reader to [27] for a detailed discussion of the various ES customisations. Algorithm 2 shows the pseudo-code for standard (($\mu \overset{+}{,} \lambda$)-ES).

For the PSO algorithm, a number of particles are placed in the search space and each evaluates the objective function value at its own location. Then, each particle updates its movement direction through the search space by combining the information about its own current and best locations with other particles. Once all the particles are moved through the search space, the current iteration has been completed. In the PSO system, each agent makes its decision according to its own experiences and other agents experiences. The system initially has a population of random solutions. Each potential solution, called a particle (agent), is given a random

Algorithm 1 Differential Evolution pseudo-code [32]

```
1: Initialise the population with randomly generated solutions
2: Set the weight $F \in [0,2]$ and crossover probability $C_r \in [0,1]$
3: while (stopping condition) do
4:     for $i = 1$ to $d$ do
5:         For each $x_i$, choose 3 distinct vectors $x_p$, $x_q$ and $x_r$ randomly
6:         Generate a new mutated vector (using the mutation) using $x_p$, $x_q$ and $x_r$
7:         Generate a trial vector (using the crossover) using the mutated and target vectors
8:         Select and update the solutions by selection
9:     end for
10:    Update the counters such as $g = g + 1$
11: end while
12: Find the best found solution (post-process)
```

Algorithm 2 Evolutionary Strategy pseudo-code [27]

```
1: Set $g = 0$ (generation counter)
2: Initialise and create the population of solutions of $x$ using uniform $n$ dimensional probability distribution in a problem search space ($\mu$ individuals)
3: Evaluate the fitness of the population
4: while (stopping condition) do
5:     Generate a new offspring (using the mutation)
6:     Evaluate the offspring fitness
7:     Apply selection from the offspring individuals
8:     $g = g + 1$
9: end while
```

velocity and is flown through the problem space. The agents have memories and each agent keeps track of its previous best position (P_{best}) and its corresponding fitness. There exist a number of P_{best} for the respective agents in the swarm and the agent with the greatest fitness is called the global best (G_{best}) of the swarm. Each particle is treated as a point in an n dimensional space. The swarm, as a whole, is likely to move through an optimum of objective function in search space. The simple version of PSO pseudo-code is shown in Algorithm 3.

Constrained optimisation problems have constraints that increase their complexity. In order to tackle these constraints, evolutionary algorithms use mechanisms known as constraint handling methods. One of the issues with EAs is how to deal with these constraints. Based on [22], constraint handling methods are categorised into preserving the feasibility of solutions, separating feasible and infeasible solutions, penalty functions and

Algorithm 3 Particle Swarm Optimisation pseudo-code [16]

```
1: Randomly the initialise positions and velocities of all particles
2: while (stopping condition) do
3:     Set $P_{best}$ and $G_{best}$
4:     Calculate particle velocity
5:     Update particle position
6:     Evaluate the objective function value (fitness value)
7: end while
```

hybrid methods. We refer the reader to [4, 22] for a detailed survey of constraint handling methods.

In this thesis, our studies cover three constrained handling algorithms from DE, ES and PSO types. These algorithms are ε-constrained differential evolution with an archive and gradient-based mutation (εDEag), $(1+1)$ CMA-ES for constrained optimisation and hybrid multi-swarm particle swarm optimisation (HMPSO), respectively [33, 38, 1]. The (εDEag) is one of the best DE algorithms and the winner of the 2010 CEC for continuous COPs [20]. The algorithm uses an ε-constrained transformation method for converting an unconstrained problem to a constrained one. Possible solutions are compared using the ε-level technique where the constraint violation ($\phi(x)$) has a greater priority than the function cost ($f(x)$). We also use $(1+1)$ CMA-ES for constrained optimisation problems. This algorithm is a variant of $(1+1)$-ES that adapts the covariance matrix of its offspring distribution in addition to its global step size. The algorithm obtains approximations to the direction of normal vectors in the current solution location vicinity by using low-pass filtering steps, which violates the constraints and decrease the offspring distribution variance in these directions. This technique makes $(1+1)$ CMA-ES one of the most efficient algorithms for constrained continuous problems. The next algorithm we use in this thesis is HMPSO. This algorithm solves the COP by dividing the current swarm into smaller sub-swarms and searching them in parallel. Using this technique, enables the algorithm to increase the diversity of solutions near different optima.

1.3 Analysis of Bio-inspired Algorithms for Constrained Continuous Optimisation Problems

Frequently, solving a COP requires significant time using trial and error attempts to choose the best single optimisation algorithm. With increasing numbers of proposed evolutionary algorithms, it has always been desirable to find the best algorithm to solve a given COP. In other words, finding the best algorithm (among a range of different options) to perform well on a given COP is not a straightforward task. Although there are many algorithms that have been used successfully to solve COPs, there is no clear understanding as to which variations of algorithms or approaches are better than the others for a given problem. In order to find a proper algorithm for a problem, many approaches have been tested. Such studies have focused on finding a measure that can divide problems into those that are easy and those that are hard to solve for all algorithms [3, 8]. However, these attempts have not been very successful as in the literature there are many counter-examples for which the proposed measure is not a reliable methodology [10, 9]. Such studies found that a more realistic method is to focus on the characteristics of problems that make them hard to solve by certain evolutionary algorithms. In other words, what is hard for DE might not necessarily hard for ES or PSO. So, this motivates researches to propose various techniques to characterise the optimisation problems. It is hoped that by analysing problems in a greater depth, it will become possible to distinguish problems based on their characteristics.

1.3.1 Fitness Landscape Ruggedness Analysis

Instead of trying to find one measure of difficulty, a more realistic approach could be to determine the characteristics of a problem and then use these characteristics to determine which algorithm would be best suited for solving the problem. Such techniques are grouped into theoretical and empirical analysis. These estimators of problem complexity, which are theoretical in nature, include measures of correlations between fitness values and measuring hardness using landscape information [39, 31, 3, 14]. The techniques discussed are focused on measuring various problem characteristics. Examples of such characteristics include deception with respect to a genetic algorithm, the fitness distribution layout, neutrality and evolvability. We refer the reader to [35, 37, 7] for the detailed studies. It is clear that there is no stand-alone problem characteristic that makes it hard or easy for EAs, but there is no doubt that COP's fitness landscape is effective in supporting an algorithm's ability to search the space. Among several fitness landscape characteristics, the notion of *fitness landscape ruggedness* influences the problem difficulty. Hence, many studies have been conducted to analyse and measure this feature in discrete optimisation and unconstrained numerical search spaces [19, 36, 39, 12, 18]. One of the most general and popular approaches in problem ruggedness characterising is discussed by Vassilev in [36]. The idea behind this is to propose theoretic information about discrete landscape smoothness, ruggedness and neutrality. To obtain a landscape path, a random walk is used on the discrete problem fitness landscape. This path is represented as an ensemble of three-point objects, where each point is the objective function value of the current point and its neighbours. In other words, the object is neutral (the current point has equal value to its neighbours), smooth (the fitness difference between three points changes in one direction: like a slope) and rugged (the fitness difference between three points changes in two directions: like a peak). Then, the ruggedness estimation with respect to neutrality is calculated using three-point objects obtained from the random walk. As discussed earlier, this approach is used in discrete problem fitness landscapes. Later, Malan modified the random walk to quantify the fitness landscape's ruggedness for unconstrained continuous problems in [19].

1.3.2 Feature-Based Analysis

The main objective of this thesis lies in the area of feature-based analysis, which is an empirical approach. Understanding the conditions under which these algorithms perform well is essential for selecting the most suitable algorithm for a given COP. In [24], Smith-Miles shows the problem difficulty analysis in two ways: one is considering a problem as a learning problem, where the automatic algorithm selection is based on the obtained knowledge from the previous algorithm's performance [15, 2]. The second one is analysing the algorithms and problems to understand the reasons for their performance [29, 30]. The terms "feature" and "characteristic" are used synonymously throughout this research.

The key concept in the problem difficulty analysis is to find the features of the problems that make them hard or easy to solve. A solution to this is to investigate easy and hard instances of a problem for certain algorithms. The major purpose of this analysis is to generate problem instances, difficulty criteria and a set of problem features. The method applied to generate the easy and hard instances is an evolutionary algorithm. The problem's hardness criteria is considered as the algorithm's performance on the generated instances. Such studies have been conducted in combinatorial optimisation [21, 34, 29]. To extend this, the approaches are designed to generate extremely hard and easy problem instances for evolutionary algorithms using an evolver. Using an evolutionary algorithm (evolver) enables us to obtain the vast diversity of generated instances for a certain algorithm(s). Then, analysing these instances could help us to identify the problem features that make it hard/easy for an algorithm. It can also represent the strengths and weaknesses of certain algorithms on a particular problem instance feature. Figure 1.1 shows the evolving process for easy or hard problem instances that we use in this thesis. Later, we use customised version of this figure for linear and quadratic constraints. In this approach, at first we define COP instances using a fixed objective function. Then the constraints coefficients for COP are uniform randomly chosen. We later solve these problem instances using an evolutionary algorithm (solver). The required function evaluation number (FEN) for solving the

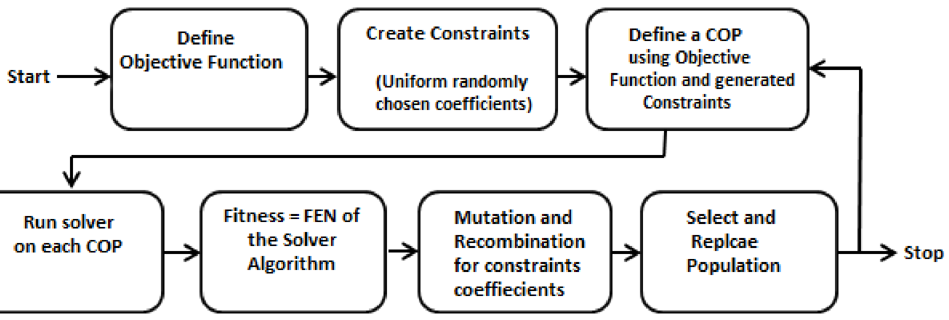
Start
Define Objective Function
Create Constraints
(Uniform randomly chosen coefficients)
Define a COP using Objective Function and generated Constraints
Run solver on each COP
Fitness = FEN of the Solver Algorithm
Mutation and Recombination for constraints coeffiecients
Select and Replcae Population
Stop

Figure 1.1: The evolving approach to generate hard/easy problem instances

COPs using an algorithm is considered as the fitness value for the evolver algorithm. By performing mutation, combination, selection or replacing these instances (based on the FEN), we generate easy or hard problems for the solver algorithm. Applying this method, we can generate a diverse set of COP instances from easy to hard for each algorithm.

1.3.3 Algorithm Selection

Given a range of algorithms (such as DE, ES and PSO) to solve a COP, there has been always a question: ”Can we estimate the likelihood that Algorithm A will perform successfully on a given constrained problem?”. Selecting the best algorithm from a suite of algorithms for a given problem's referred to an algorithm selection problem (ASP) [26]. In this work, Rice et al. proposed a method with four characteristics: a set of problem instances (F), a set of algorithms (A), the cost of algorithm performance on a particular problem (Y) and a set of characteristics of problem instances (C). An example of his algorithm selection framework is represented in Figure 1.2. This figure predicts the performance of $y(a(f))$ of a given algorithm (a) on a problem (f) by extracted features (c). Various studies have extended this framework, see [13, 17, 30]. Also, many studies have been conducted to show the possibility of finding the best suited algorithm for a given problem [23, 28, 2]. Using a learning strategy from previous experiments and extracting given problem features, it is possible to predict an algorithm's performance on a given problem. Based on these findings, it is possible to find the links between problem characteristics and an algorithm's performance. The key to this lies in the problem features, which can be used for best algorithm prediction. This knowledge can be used to create future prediction framework for selecting the most suitable algorithm for a given optimisation problem. This thesis proposes how Rice's model [26] can be applied to constrained optimisation problems and evolutionary algorithms,

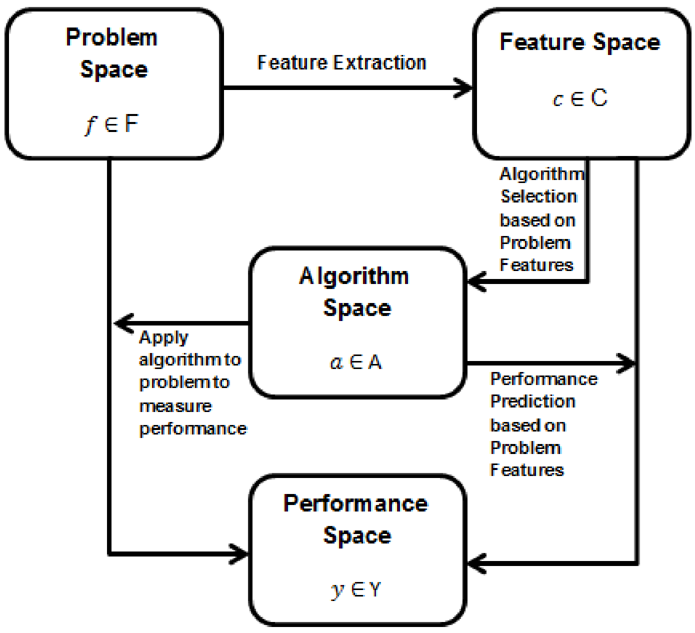
Problem Space
$f \in F$
Feature Extraction
Feature Space
$c \in C$
Algorithm Selection based on Problem Features
Algorithm Space
$a \in A$
Apply algorithm to problem to measure performance
Performance Prediction based on Problem Features
Performance Space
$y \in Y$

Figure 1.2: Algorithm selection framework

where the features of the problems are based on feature-based analysis for constraints. A neural network is used to solve the problem of mapping COP features to performance measures (FENs).

1.4 Goals and Contribution

The goal of this thesis is to design an automated algorithm selection method for a given COP based on its constraints. In order to select the most suited algorithm, firstly, the features of the constraints that make the problem hard/easy to solve are analysed. These features are related to the complex set of problem constraints. Then, this knowledge is used to design and implement an algorithm prediction model for constrained optimisation problems. This algorithm selection framework can predict the best algorithm type with its required FEN for a given COP using only its constraint features.

The organisation of the thesis is as follows: In Chapter 2, we first review the preliminary theoretical analysis of fitness landscapes. We also discuss the application of landscape ruggedness measurement for discrete and unconstrained real-valued optimisation problems. By applying these techniques on constrained optimisation problems, we prove that the ruggedness measure is not accurate. Therefore, to cope with constraints and infeasible areas in COPs or their highly infeasible search space, we use a biased walk to collect ruggedness information. We also investigate the effects of our new approach in quantifying the constraint optimisation problems with various benchmarks. Results show that the ruggedness information collected by our proposed method is more reliable and useful comparing to the techniques. Therefore, it is more beneficial to use this theoretical characterising analysis for future problem difficulty predictions and algorithm selection models.

Having initiated a theoretical constrained optimisation problem analysis, Chapter 3 shows a first step towards more practical feature-based analysis for COPs. It is obvious that constraints are important in constrained optimisation problems. Hence, this motivates us to focus on the features of constraints that make COPs hard/easy for various types of algorithms. We adopt an evolving approach to ensure that the sets of instances are varied from easy to hard for constraint handling algorithms. Then, by extracting and analysing their features, the relationship between the algorithms and constraints in COPs can be identified. Thus, as our initial step, we conduct a feature-based analysis on the impact of a linear constraint for an ε-constrained differential evolution (ε-DEag) [33]. We use an evolver to generate COP instances with a linear constraint and different fixed objective functions for a solver algorithm. By extracting the constraint features, we study their effects on COP difficulty.

Chapter 4 further extends the feature-based analysis on COPs for more complex sets of constraints. The results for previous steps motivate us to study a set of linear, quadratic (and their combination) constraints to make a problem hard or easy for an algorithm. We carry out an evolving approach to generate COP instances with a set of linear and/or quadratic constraints. Our results provide the capability of a set of various types of constraints to problem difficulty solving by ε-DEag algorithm. With this feature-based analysis, we obtain knowledge about the influence of constraints on problem hardness, which can be used to design a successful prediction model for selecting an algorithm. Using the results for Chapters 3 and 4, we proved the high quality of this evolving approach to analyse the effects of constraint features on COPs difficulty for an ε-DEag algorithm.

To design a useful algorithm selection model, we require information about strengths and weaknesses of other algorithm types over the features of constraints. This leads us to study the effect of constraints and their features on various algorithm types over COP difficulty. Thus, in Chapter 5, by using a single-objective evolver, we generate hard and easy COP instances for DE, ES and PSO algorithm types. Then, to gain deeper insight, we solve each set of generated instances (using an algorithm) by the other algorithms. Results suggest that the instances that are hard/easy for one algorithm type are still hard/easy for the others. This information is good but not sufficient, especially for designing a future algorithm selection model. So, we apply a multi-objective evolver that generates different sets of problem instances. This evolving approach generates COP instances that are hard/easy for one but still easy/hard for the other algorithms. The experiments show that over which features of constraints, they can make the problem hard/easy for one and still easy/hard for the other algorithms. As is observed in [25], some constraint features have a greater effect on problem difficulty for various types of algorithms. By analysing how well an algorithm performs on a given COP where the others fail, we can obtain their strengths and weaknesses. This knowledge is helpful for implementing more reliable algorithm prediction model.

In Chapter 6, we design a meta-learning framework to build a prediction model for a given COP. The inputs to this model are given problem constraint features and selected parameter settings. The outputs include the required FEN and the most suitable algorithm type for a given COP. To build a reliable prediction model, we need to train it with the proper data, so as to show the differences between various algorithm types and their performances on different constraints features. Therefore, for our algorithm prediction model, we use the approach in Chapter 5 to obtain evolving COP instances. However, selecting the best sets of instances from a multi-objective evolver population in order to build a reliable prediction model is a challenge. Experimenting various subsets of instances from a multi-objective evolver population for the training phase is effective for model accuracy. Hence, we trained the model with different subsets from the evolver population. Based on the experimental results, it is observed that the prediction model trained with the combination of Pareto front and random points has the highest accuracy in algorithm type prediction for a given COP. The results indicate that our model is reliable to predict the most suitable algorithm with highly close required FEN for a given COP. These results clearly demonstrate the ability of prediction model to choose the best algorithm type, using only constraint features. We then conclude this thesis in Chapter 7 with some final remarks.

References

[1] D. V. Arnold and N. Hansen. A (1+ 1)-cma-es for constrained optimisation. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 297–304. ACM, 2012.

[2] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuß. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 313–320. ACM, 2012.

[3] Y. Borenstein and R. Poli. Information landscapes. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1515–1522. ACM, 2005.

[4] C. A. C. Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, 191(11):1245–1287, 2002.

[5] A. E. Eiben and J. E. Smith. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[6] C. A. Floudas and P. M. Pardalos. *A collection of test problems for constrained global optimization algorithms*, volume 455. Springer Science & Business Media, 1990.

[7] D. E. Goldberg. Simple genetic algorithms and the minimal, deceptive problem. *Genetic algorithms and simulated annealing*, 74:88, 1987.

[8] D. E. Goldberg, J. rey Horn, and K. Deb. What makes a problem hard for a classi er system? *Urbana*, 51:61801, 1992.

[9] J. J. Grefenstette. Deception considered harmful sk. *Foundations of Genetic Algorithms 1993 (FOGA 2)*, 2:75, 2014.

[10] H. Guo and W. H. Hsu. Ga-hardness revisited. In *GECCO*, pages 1584–1585. Citeseer, 2003.

[11] D. M. Himmelblau. *Applied nonlinear programming*. McGraw-Hill Companies, 1972.

[12] W. Hordijk and P. F. Stadler. Amplitude spectra of fitness landscapes. *Advances in Complex Systems*, 1(01):39–66, 1998.

[13] F. Hutter, Y. Hamadi, H. H. Hoos, and K. Leyton-Brown. Performance prediction and automated tuning of randomized and parametric algorithms. In *Principles and Practice of Constraint Programming-CP 2006*, pages 213–228. Springer, 2006.

[14] T. Jones, S. Forrest, et al. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *ICGA*, volume 95, pages 184–192, 1995.

[15] J. Kanda, A. Carvalho, E. Hruschka, and C. Soares. Selection of algorithms to solve traveling salesman problems using meta-learning. *International Journal of Hybrid Intelligent Systems*, 8(3):117–128, 2011.

[16] J. Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

[17] K. Leyton-Brown, E. Nudelman, and Y. Shoham. Empirical hardness models: Methodology and a case study on combinatorial auctions. *Journal of the ACM (JACM)*, 56(4):22, 2009.

[18] M. Lipsitch. Adaptation on rugged landscapes generated by iterated local interactions of neighboring genes. In *ICGA*, pages 128–135, 1991.

[19] K. M. Malan and A. P. Engelbrecht. Quantifying ruggedness of continuous landscapes using entropy. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 1440–1447. IEEE, 2009.

[20] R. Mallipeddi and P. N. Suganthan. Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. *Nanyang Technological University, Singapore*, 2010.

[21] O. Mersmann, B. Bischl, H. Trautmann, M. Wagner, J. Bossek, and F. Neumann. A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. *Annals of Mathematics and Artificial Intelligence*, 69(2):151–182, 2013.

[22] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, 4(1):1–32, 1996.

[23] M. A. Muñoz, M. Kirley, and S. K. Halgamuge. A meta-learning prediction model of algorithm performance for continuous optimization

problems. In *Parallel Problem Solving from Nature-PPSN XII*, pages 226–235. Springer, 2012.

[24] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of linear constraints for ε-constrained differential evolution. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 3088–3095. IEEE, 2014.

[25] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of set of constraints for ε-constrained differential evolution. In S. Arik, T. Huang, W. K. Lai, and Q. Liu, editors, *Neural Information Processing*, volume 9491 of *Lecture Notes in Computer Science*, pages 344–355. Springer International Publishing, 2015.

[26] J. R. Rice. The algorithm selection problem. 1975.

[27] H.-P. P. Schwefel. *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.

[28] K. Smith-Miles. Towards insightful algorithm selection for optimisation using meta-learning concepts. In *WCCI 2008: IEEE World Congress on Computational Intelligence*, pages 4118–4124. IEEE, 2008.

[29] K. Smith-Miles, J. van Hemert, and X. Y. Lim. Understanding tsp difficulty by learning from evolved instances. In *Learning and intelligent optimization*, pages 266–280. Springer, 2010.

[30] K. A. Smith-Miles, R. J. James, J. W. Giffin, and Y. Tu. A knowledge discovery approach to understanding relationships between scheduling problem structure and heuristic performance. In *Learning and intelligent optimization*, pages 89–103. Springer, 2009.

[31] P. F. Stadler. Landscapes and their correlation functions. *Journal of Mathematical chemistry*, 20(1):1–45, 1996.

[32] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[33] T. Takahama and S. Sakai. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–9. IEEE, 2010.

[34] J. I. van Hemert. Evolving combinatorial problem instances that are difficult to solve. *Evolutionary Computation*, 14(4):433–462, 2006.

[35] L. Vanneschi, Y. Pirola, and P. Collard. A quantitative study of neutrality in gp boolean landscapes. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 895–902. ACM, 2006.

[36] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Information characteristics and the structure of landscapes. *Evolutionary Computation*, 8(1):31–60, 2000.

[37] S. Verel, P. Collard, and M. Clergue. Where are bottlenecks in nk fitness landscapes? In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 1, pages 273–280. IEEE, 2003.

[38] Y. Wang and Z. Cai. A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems. *Frontiers of Computer Science in China*, 3(1):38–52, 2009.

[39] E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological cybernetics*, 63(5):325–336, 1990.

Chapter 2

Analytical Feature Analysis for Constrained Optimisation Problems

The article in this chapter represents the literature review in theoretical problem analysis [1]. It reviews the theoretical analysis on both constrained and unconstrained optimisation problems. The aim is to understand the relationship between theoretical problem features and success of certain types of algorithms to solve a problem. To measure the ruggedness for constrained continuous optimisation problem landscapes, we introduce a new approach which uses biased walk. The experimented results show the reliability of the proposed approach in highly infeasible problems.

References

[1] S. Poursoltan and F. Neumann, "Ruggedness quantifying for constrained continuous fitness landscapes," in *Evolutionary Constrained Optimisation*. Springer, 2015, pp. 29–50

Statement of Authorship

Title of Paper	Ruggedness Quantifying for Constrained Continuous Fitness Landscapes		
Publication Status	[✓ Published]	Accepted for Publication	
	Submitted for Publication	Unpublished and Unsubmitted w ork w ritten in manuscript style	
Publication Details	S. Poursoltan and F. Neumann, "Ruggedness quantifying for constrained continuous fitness landscapes," in Evolutionary Constrained Optimisation. Springer, 2015, pp. 29–50.		

Principal Author

Name of Principal Author (Candidate)	Shayan Poursoltan		
Contribution to the Paper	Read the existing articles, categorised them. Performed analysis on all samples and data, wrote the first draft of the book chapter and acted as corresponding author.		
Overall percentage (%)	85%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	3/2/2016

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Frank Neumann		
Contribution to the Paper	Supervised development of work. Read the chapter and provided comment including editorial comments and conceptual feedback to improve the book chapter.		
Signature		Date	3/2/2016

Name of Co-Author			
Contribution to the Paper			
Signature		Date	

Please cut and paste additional co-author panels here as required.

Poursoltan, S. & Neumann, F. (2015) Ruggedness quantifying for constrained continuous fitness landscapes. In R. Datta & K. Deb (Eds.), *Evolutionary Constrained Optimisation*. Infosys Science Foundation Series. (pp. 29-50). New Delhi, India: Springer.

Chapter 3

A Feature-Based Analysis on the Impact of Linear Constraints for ε-Constrained Differential Evolution

The article in this chapter presents the first empirical feature-based analysis for constrained continuous optimisation problems [1]. It reviews the evolving approach to generate easy and hard instances for a certain type of algorithm by evolving a linear constraint in COP. We then analyse the relationship between constraint features and algorithm performance. The constraint features are discussed in detail. The results show that a linear constraint coefficients can make a problem up to 30% more difficult to solve in terms of function evaluation number.

References

[1] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of linear constraints for ε-constrained differential evolution. In *Evolutionary Computation (CEC), 2014 IEEE Congress on, Beijing 12-14 July*, pages 3088–3095. IEEE, 2014

Statement of Authorship

Title of Paper	A Feature-Based Analysis on the Impact of Linear Constraints for e-Constrained Differential Evolution
Publication Status	[✓ Published] Accepted for Publication Submitted for Publication Unpublished and Unsubmitted work written in manuscript style
Publication Details	S. Poursoltan and F. Neumann. A feature-based analysis on the impact of linear constraints for e-constrained differential evolution. In Evolutionary Computation (CEC), 2014 IEEE Congress on, pages 3088–3095. IEEE, 2014.

Principal Author

Name of Principal Author (Candidate)	Shayan Poursoltan		
Contribution to the Paper	Read the existing articles, categorised them, tested the idea to confirm its efficiency, wrote the first draft and applied comments from Co-author. The principal author also presented this paper in the conference.		
Overall percentage (%)	85%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	3/2/2016

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Frank Neumann		
Contribution to the Paper	Supervised development of work. Read the paper and provided comment including editorial comments and conceptual feedback to improve the paper.		
Signature		Date	3/2/2016

Name of Co-Author			
Contribution to the Paper			
Signature		Date	

Please cut and paste additional co-author panels here as required.

S. Poursoltan and F. Neumann. (2014). A feature-based analysis on the impact of linear constraints for ε -constrained differential evolution. *2014 IEEE Congress on Evolutionary Computation (CEC),* (pp. 3088–3095). Beijing, China: IEEE.

Chapter 4

A Feature-Based Analysis on the Impact of Set of Constraints for ε-Constrained Differential Evolution

In this article [1], a feature-based analysis of evolved constrained continuous optimisation instances is carried out to understand the characteristics of constraints that make a problem hard for evolutionary algorithms. In this study, we examined more complex set of constraints such as linear, quadratic and their combinations. Investigating the features of the constraint in COP, we can obtain the knowledge of what type of constraints and their features make a COPs difficult for the examined algorithm. The results have been published in [1]. We include the extended version [2] in this chapter.

References

[1] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of set of constraints for ε-constrained differential evolution. In S. Arik, T. Huang, W. K. Lai, and Q. Liu, editors, *Neural Information Processing*, volume 9491 of *Lecture Notes in Computer Science*, pages 344–355. Springer International Publishing, 2015

[2] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of set of constraints for e-constrained differential evolution. *CoRR*, abs/1506.06848, 2015

Statement of Authorship

Title of Paper	A Feature-Based analysis on the Impact of Set of Constraints for e-Constrained Differential Evolution
Publication Status	[✓ Published] Accepted for Publication Submitted for Publication Unpublished and Unsubmitted work written in manuscript style
Publication Details	S. Poursoltan and F. Neumann. A feature-based analysis on the impact of set of constraints for e-constrained differential evolution. In S. Arik, T. Huang, W. K. Lai, and Q. Liu, editors, Neural Information Processing, volume 9491 of Lecture Notes in Computer Science, pages 344–355. Springer International Publishing, 2015.

Principal Author

Name of Principal Author (Candidate)	Shayan Poursoltan		
Contribution to the Paper	Read the existing articles, categorised them, tested the idea to confirm its efficiency, wrote the first draft and applied comments from co-author. Also, principa author presented the paper in the conference.		
Overall percentage (%)	85%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	3/2/2016

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Frank Neumann		
Contribution to the Paper	Supervised development of work. Read the paper and provided comment including editorial comments and conceptual feedback to improve the paper.		
Signature		Date	3/2/2016

Name of Co-Author			
Contribution to the Paper			
Signature		Date	

Please cut and paste additional co-author panels here as required.

Statement of Authorship

Title of Paper	A Feature-Based analysis on the Impact of Set of Constraints for e-Constrained Differential Evolution
Publication Status	[✓ Published] ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	S. Poursoltan and F. Neumann. A feature-based analysis on the impact of set of constraints for e-constrained differential evolution. CoRR, abs/1506.06848, 2015. This paper is the extended version of published conference paper.

Principal Author

Name of Principal Author (Candidate)	Shayan Poursoltan		
Contribution to the Paper	Read the existing articles, categorised them, tested the idea to confirm its efficiency, wrote the first draft and applied comments from Co-author. Also, principa author presented the paper in the conference.		
Overall percentage (%)	85%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	3/2/2016

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Frank Neumann		
Contribution to the Paper	Supervised development of work. Read the paper and provided comment including editorial comments and conceptual feedback to improve the paper.		
Signature		Date	3/2/2016

Name of Co-Author			
Contribution to the Paper			
Signature		Date	

Please cut and paste additional co-author panels here as required.

A Feature-Based Analysis on the Impact of Set of Constraints for ε-Constrained Differential Evolution

Shayan Poursoltan
Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, Australia

Frank Neumann
Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, Australia

June 24, 2015

Abstract

Different types of evolutionary algorithms have been developed for constrained continuous optimization. We carry out a feature-based analysis of evolved constrained continuous optimization instances to understand the characteristics of constraints that make problems hard for evolutionary algorithm. In our study, we examine how various sets of constraints can influence the behaviour of ε-Constrained Differential Evolution. Investigating the evolved instances, we obtain knowledge of what type of constraints and their features make a problem difficult for the examined algorithm.

1 Introduction

Constrained optimisation problems (COPs), specially non-linear ones, are very important and widespread in real world applications [1]. This has motivated introducing various algorithms to solve COPs. The focus of these algorithms is to handle the involved constraints. In order to deal with the constraints, various mechanisms have been adopted by evolutionary algorithms. These techniques include penalty function, decoder-based methods and special operators that separate the treatment of constraints and objective functions. For an overview of different types of methods we refer the reader to Mezura-Montes and Coello Coello [6].

With the increasing number of evolutionary algorithms, it is hard to predict which algorithm performs better for a newly given COP. Various benchmark sets such as CEC'10 [3] and BBOB'10 [2] have been proposed to evaluate the algorithm performances on continuous optimization problems. The aim of these benchmarks is to find out which algorithm is good on which classes of problems. For constrained continuous optimization problems, there has been an increasing interest to understanding problem features from a theoretical perspective [9, 14]. The feature-based analysis of of hardness for certain classes of algorithms is a relatively new research area. Such studies classify problems as hard or easy for a given algorithm based on the features of given instances. Initial studies in the context of continuous optimization have recently been carried out in [4, 5]. Having enough knowledge on problem properties that make it hard or easy, we may choose the most suited algorithm to solve it. To do this, two steps approach has been proposed by Mersmann et al. [4]. First, one has to extract the important features from a group of investigated problems. Second, in order to build a prediction model, it is necessary to analyse the performance of various algorithms on these features. Feature-based analysis has also been used to gain new insights in algorithm performance for discrete optimization problems [7, 10].

In this paper, we carry out a feature-based analysis for constrained continuous optimisation and generate a variety of problem instances from easy to hard ones by evolving constraints. This ensures that the knowledge obtained by analysing problem features covers a wide range of problem instances that are of particular interest. Although what makes a problem hard to solve is not a standalone feature, it is assumed that constraints are certainly important in constrained problems. Evolving constraints is a new technique to generate hard and easy instances. So far, the influence of one linear constraint has been studied [8]. However, real world problems have more than one linear constraint (such as linear, quadratic and their combination). Hence, our study is to generate COP instances to investigate which features of the linear and quadratic constraints make the constrained problem hard to solve. To provide this knowledge, we need to use a common suitable evolutionary algorithm that handles the constraints. In this research, the ε-constrained differential evolution with an archive and gradient-based mutation (εDEag) [13] is used. The εDEag (winner of CEC 10 special session for constrained problems) is applied to generate hard and easy instances to analyse the impact of set of constraints on it.

Our results provide evidence on the capability of constraints (linear, quadratic or their set of combination) features to classify problem instances to easy and hard ones. Feature analysis by solving the generated instances with εDEag enables us to obtain the knowledge of influence of constraints on problem hardness which could later could be used to design a successful prediction model for algorithm selection.

The rest of the paper is organised as follows. In Section 2, we introduce the constrained optimisation problems. Then, we discuss εDEag algorithm that we use to solve the generated problem instances. Section 3 includes our approach to evolve and generate problem instances. Furthermore, the constraint features are discussed. In Section 4, we carry out the analysis of the linear and quadratic constraint features. Finally, section 5 concludes with some remarks.

2 Preliminaries

2.1 Constrained continuous optimisation problems

Constrained continuous optimisation problems are optimisation problems where a function $f(x)$ on real-valued variables should be optimised with respect to a given set of constraints. Constraints are usually given by a set of inequalities and/or equalities. Without loss of generality, we present our approach for minimization problems.

Formally, we consider single-objective functions $f\colon S \to \mathbb{R}$, with $S \subseteq \mathbb{R}^n$. The constraints impose a feasible subset $F \subseteq S$ of the search space S and the goal is to find an element $x \in S \cap F$ that minimizes f.

We consider problems of the following form:

$$\begin{aligned} \text{minimize} \quad & f(x), \quad x = (x_1, \ldots, x_n) \in \mathbb{R}^n \\ \text{subject to} \quad & g_i(x) \leq 0 \quad \forall i \in \{1, \ldots, q\} \\ & h_j(x) = 0 \quad \forall j \in \{q+1, \ldots, p\} \end{aligned} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n)$ is an n dimensional vector and $x \in S \cap F$. Also $g_i(x)$ and $h_j(x)$ are inequality and equality constraints respectively. Both inequality and equality constraints could be linear or nonlinear. To handle equality constraints, they are usually transformed into inequality constraints as $|h_j(x)| \leq \varepsilon$, where $\varepsilon = 10e^{-4}$ (used in [3]). Also, the feasible region $F \subseteq S$ of the search space S is defined by

$$l_i \leq x_i \leq u_i, \qquad 1 \leq i \leq n \tag{2}$$

where both l_i and u_i denote lower and upper bounds for the ith variable and $1 \leq i \leq n$ respectively.

2.2 εDEag algorithm

One of the most prominent evolutionary algorithms for COPs is ε-constrained differential evolution with an archive and gradient-based mutation (εDEag). The algorithm is the winner of latest CEC competition for constrained constrained continuous problems [3]. The εDEag uses ε-constrained method to transform algorithms for unconstrained problems to constrained ones [12]. It adopts ε-level comparison instead of ordinary ones to order the possible solutions. In other words, the lexicographic order is performed in which constraint violation ($\phi(x)$) has more priority and proceeds the function value ($f(x)$). This means feasibility is more important. Let f_1,f_2 and ϕ_1,ϕ_2 are objective function values and constraint violation at x_1,x_2 respectively. Hence, for all $\varepsilon \geq 0$, the ε-level comparison of two candidates (f_1, ϕ_1) and (f_2, ϕ_2) is defined as the follows:

$$(f_1, \phi_1) <_\varepsilon (f_2, \phi_2) \iff \begin{cases} f_1 < f_2, & \text{if} \quad \phi_1, \phi_2 \leq \varepsilon \\ f_1 < f_2, & \text{if} \quad \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{otherwise} \end{cases}$$

$$(f_1, \phi_1) \leq_\varepsilon (f_2, \phi_2) \iff \begin{cases} f_1 \leq f_2, & \text{if} \quad \phi_1, \phi_2 \leq \varepsilon \\ f_1 \leq f_2, & \text{if} \quad \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{otherwise} \end{cases}$$

In order to improve the usability, efficiency and stability of the algorithm, an archive has been applied. Using it improves the diversity of individuals (see Algorithm 1). The

Algorithm 1 The ε-constrained differential evolution with an archive and gradient-based mutation (εDEag)

- Initializations:
  - M randomly selected individuals from search space S is archived in A.
  - Set ε level Using control level function
  - Population: Top N individuals are selected from archive. The Archive is ranked using ε level comparison
- Termination condition is set to Maximum function evaluation number.
- DE operation: Use DE/rand/1/exp to generate new child. Comparing is based on the ε level comparison
- Gradient based mutation: If child is infeasible, it is changed by the gradient-based mutation with probability P. Go to step 3 and parent is considered as parent.
- Update and control the ε-level
- Go to step 2

offspring generation is adopted in such a way that if the child is not better than its parent, the parent generates another one (see [13]). This leads to more stability to the algorithm. For a detailed presentation of the algorithm, we refer the reader to [13].

3 Evolving Constraints

It is assumed that the role of constraints in problem difficulty is certainly important for constrained optimisation problem. Hence, it is necessary to analyse various effects that constraint can impose on a constrained optimisation problems. Evolving constraints is a novel methodology to generate hard and easy instances based on the performance of the problem solver (optimisation algorithm).

3.1 Algorithm

In order to analyse the effects of constraints, the variety of them needs to be studied over a fixed objective function. First, constraint coefficients are randomly chosen to construct problem instances. Second, the generated constrained problem is solved by a solver algorithm (εDEag). Then, the required function evaluation number (FEN) to solve this instance is considered as the fitness value for evolving algorithm. This process is repeated until hard and easy instances of constraint problem are generated (see Figure 1).

To generate hard and easy instances, we use the approach outlined in [8].

It uses fast and robust differential evolution (DE) proposed in [11] (see Algorithm 2, 3) to evolve through the problem instances (by generating various constraint coefficients). It is necessary to note that the aim is to optimise (maximise/minimise) the FEN that is required by a solver to solve the generated problem. Also, to solve this generated problem instance and find the required FEN we use εDEg as a solver. The

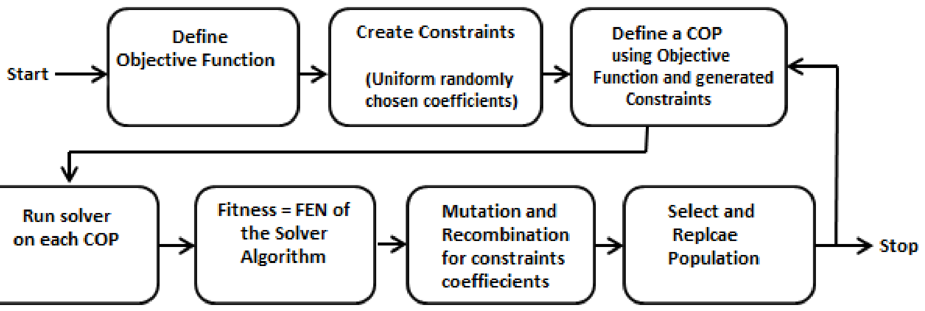
Start
Define Objective Function
Create Constraints (Uniform randomly chosen coefficients)
Define a COP using Objective Function and generated Constraints
Run solver on each COP
Fitness = FEN of the Solver Algorithm
Mutation and Recombination for constraints coeffieceints
Select and Replcae Population
Stop

Figure 1: Evolving constraints process

termination condition of this algorithm (evolver) is set to reaching FENmax number of function evaluations or finding a solution close enough to the feasible optimum solution as follows:

$$|f(x_{optimum}) - f(x_{best})| \leq e^{-12} \tag{3}$$

This process generates harder and easier problem instances until it reaches the certain number of generation for the DE algorithm (evolver). Once two distinct sets of easy and hard instances are ready, we start analysing various features of the constraints for these two categories. This could give us the knowledge to understand which features of constraints have more contribution to problem difficulty.

3.2 Evolving a set of inequality constraints

We focus on analysing the effects of constraints (linear, quadratic and their combination) on the problem and algorithm difficulty. We will extract features of constraints and analyse their effect on constrained problem difficulty. The experimented constraints are linear and quadratic as the form of:

$$\text{linear constraint} \quad g(x) = b + a_1x_1 + \ldots + a_nx_n \tag{4}$$

$$\text{quadratic constraint} \quad g(x) = b + a_1x_1^2 + a_2x_1 \ldots + a_{2n-1}x_n^2 + a_{2n}x_n \tag{5}$$

or combination of them. We also consider various numbers of these constraints in this study. Here, $x_1, x_2 \ldots, x_n$ are the variables from Equation 1 and $a_1, a_2 \ldots, a_n$ are coefficients within the lower and upper bounds (l_c, u_c). In our research, we construct constrained problems where the optimum of the experimented unconstrained problem is feasible. We use quadratic function as the form of Equation 5 (univariate) since it is more popular in recent constrained problem benchmarks. Also, the influence of

Algorithm 2 Differential evolution (DE) algorithm

```
• inputs: Problem and $Pop_{size}$, $Crossover_{rate}$ ,$weighting_{factor}$, outputs: $S_{best}$
• Population ← InitializePop
  EvaluatePopulation(population)
  $S_{best}$ ← GetBestSolution(Population)
• Repeat
•   NewPopulation ← $\phi$
•   For i starts at 1, i< $Pop_{size}$-1, increment i
•     $S_i$ ← Newsample
•     If Cost($S_i$)≤Cost($P_i$)
•       NewPopulation ← $S_i$
•     else
•       NewPopulation ← $P_i$
•     Endif
•   Endfor
•   Population ← NewPopulation
•   EvaluatePopulation(population)
•   $S_{best}$ ← GetBestSolution(Population)
• Until (stop condition)
```

each x_ns can be analysed independently (exponent 2). The optimum of these problems is $x^* = (0,\dots,0)$ and we ensure that this point is feasible by requiring $b \leq 0$, when evolving the constraints.

3.3 Constraints Features

In this paper, we study a set of statistic based features that leads to generating hard and easy problem instances. These features are discussed as follows:

- **Constraint Coefficients Relationship:** It is likely that the statistics such as standard deviation, population standard deviation and variance of the constraints coefficients can represent the constraints influences to problem difficulty. These constraint coefficients are $(b, a_1, a_2, \dots, a_n)$ in Equation 4 and 5.

- **Shortest Distance:** This feature is related to the shortest distance between the objective function optimum and constraint. In this paper, the shortest distance to the known optimum from each constraint and their relations to each other is discussed. To find the shortest distance of optimum point $(x_1, x_2, \dots, x_n)$ to the linear constraint hyperplane $(a_1x_1 + a_2x_2 + \dots a_nx_n + b = 0)$ we use Equation 6. also, for quadratic constraint hyperplane $(a_1x_1^2 + a_2x_1 \dots + a_{(2n-1)}x_n^2 + a_{2n}x_n + b = 0)$ we need to find the minimum of Equation 7.

Algorithm 3 Newsample function in Algorithm 2

```
• inputs: $P_0$, population, NP, F, CR, outputs: $S$
• Repeat
•   $P_1 \leftarrow$ RandomMember(population)
• Untill $P_1 \neq P_1$
• Repeat
•   $P_2 \leftarrow$ RandomMember(population)
• Untill $P_2 \neq P_0 \vee P_2 \neq P_1$
• Repeat
•   $P_3 \leftarrow$ RandomMember(population)
• Untill $P_3 \neq P_0 \vee P_3 \neq P_1$ $P_3 \neq P_2$
• cutpoint $\leftarrow$ RandomMember(population)
• $Sample \leftarrow 0$
• For i starts a 1 to NP
•   If i $\equiv$ cutpoint $\wedge$ Rand() $\leq$ CR
•     $S_i \leftarrow P_{3_i}$ + F*($P_{1_i}$-$P_{2_i}$)
•   Else
•     $S_i \leftarrow P_{0_i}$
•   Endif
•   Endfor
• Return $S$
```

$$d_\perp = \frac{a_1x_{01} + a_2x_{02} + \ldots a_nx_{0n} + b}{\sqrt{a_1{}^2 + a_2{}^2 + \cdots + a_n{}^2}} \tag{6}$$

$$d_\perp = \sqrt{(x_1 - x_{01})^2 + (x_2 - x_{02})^2 + \cdots + (x_n - x_{0n})^2} \tag{7}$$

where $d_\perp$ in Equation 7 is the distance from a point to a quadratic hyperplane. Minimizing the distance squared ($d_\perp^2$) is equivalent to minimizing the distance $d_\perp$.

- **Angle:** This feature describes the angle of the constraints hyperplanes to each other. It is assumed that the angle between the constraints can influence problem difficulty. To calculate the angle between two linear hyperplanes, we need to find their normal vectors and angle between them using the following equation:

$$\theta = \arccos \frac{n_1 \cdot n_1}{|n_1||n_2|} \tag{8}$$

where n_1,n_2 are normal vectors for two hyperplanes. Also, the angle between two quadratic constraints is the angle between two tangent hyperplanes of their intersection. Then, the angle between these tangent hyperplanes can be calculated by Equation 8.

- **Number of Constraints:** Number of constraints plays an important role in problem difficulty. In this research, number of constraints and their effects to make easy and hard problem instances is analysed.

- **Optimum-local Feasibility Ratio:** Although the global feasibility ratio is important to find the initial feasible point, it should not affect the convergence rate during solving the problem. So, in this research, he feasibility ratio of generated COP is calculated by choosing $10e6$ random points within the vicinity of the optimum in search space and the ratio of feasible points to all chosen ones is reported. In our experiment, the vicinity of optimum is equivalent to 1/10 of boundaries from optimum for each dimension.

4 Experimental Analysis

We now analyse the features of constraints (linear, quadratic and their combination) for easy and hard instances. We generate these instances for (εDEg) algorithm using well known objective functions. In our experiments, we generate two sets of hard and easy problem instances. Due to stochastic nature of evolutionary algorithms, for each number of constraints we perform 30 independent runs for evolving easy and hard instances. We set the evolving algorithm (DE) generation number to 5000 for obtaining the proper easy and hard instances. The other parameters of evolving algorithm are set to population size = 40, crossover rate = 0.5, scaling factor = 0.9 and FEN_{max} is $300,000$. Values for these parameters have been obtained by optimising the performance of the evolving algorithm in order to achieve the more easier and harder problem instances. For (εDEg) algorithm, its best parameters are chosen based on [13]. The (εDEg) algorithm parameters are considered as: generation number = 1500, population size = 40, crossover rate = 0.5, scaling factor = 0.9. Also, the parameters for e-constraint method are set to control generation (Tc) = 1000, initial e level (q) = 0.9, archive size = $100n$ (n is dimension number), gradient-based mutation rate (Pg) = 0.2 and number of repeating the mutation (Rg) = 3.

4.1 Analysis for Linear Constraints

In order to focus only on constraints, we carry out our experiments on various well-known objective functions. These functions are considered as: Sphere (bowl shaped), Ackley (many local optima), Rosenbrock (valley shaped) and Schaffer (many local minima) (see [2]). The linear constraint is as the form of Equation 4 with dimension (n) as 30 and all coefficients (a_n)s and bs are within the range of $[-5, 5]$. Also, number of constraints is considered as 1 to 5. To discuss and study some features such as shortest distance to optimum, we assume that zero is optimum (all bs should be negative). We used (εDEg) algorithm as solver to generate more easy and hard instances.

To illustrate our investigation, we plot a 2 dimension Sphere function with 2 to 5 linear constraints in Figure 2. It is obvious that the first row (easy) instances have higher feasibility ratio than the second row (hard).

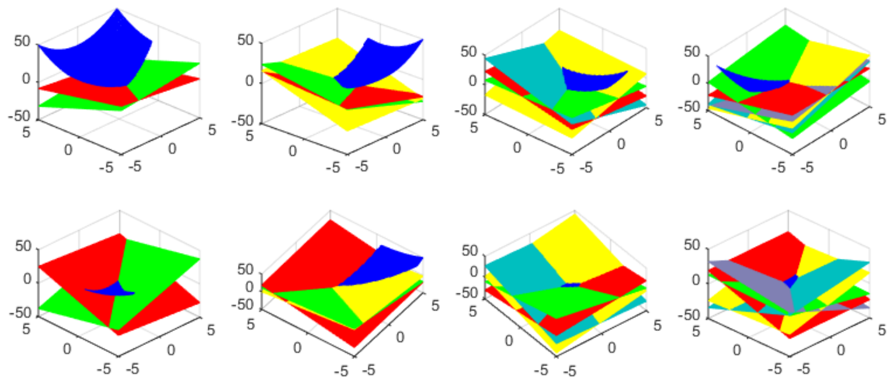

Figure 2: Easy (first row) and hard (second row) instances for 1 to 5 number of linear constraints using εDEg (2 dimension). The dark blue hyperplane is the feasible solution

In the following we will present our findings based on various features for linear constraints (for each dimension).

Figure 4 shows some evidence about linear constraints coefficients relationship such as standard deviation. It is obvious that there is a systematic relationship between the standard deviation of linear constraint coefficients and problem difficulty. The box plot (see Figure 4) represents the results for easy and hard instances using Sphere, Ackley, Rosenbrock and Schaffer objective function for (εDEg) algorithm (solver). As it is observed, the standard deviation for coefficients in each constraint (1 to 5) for easy instances are lower than hard ones. Both these coefficient values can be a significant role to make a constrained problem harder or easier to solve. Also, interestingly, all different objective functions follow the same pattern.

Figure 5 represents variation of shortest distance to optimum feature for easy and hard instances using (εDEg) algorithm. The lower value means the higher distance from optimum. This means, the linear hyperplanes in easy instances are further from optimum. Based on results, there is a strong relationship between problem hardness and shortest distance of constraint hyperplanes to optimum. In other word, this feature is contributing to problem difficulty. As expected, all objective functions follow the same systematic relationship between their feature and problem difficulty. This means, this feature can be used as a proper source of knowledge for predicting problem difficulty.

The angle between linear constraint hyperplanes feature shows relationship between the angle and problem difficulty (see Table 1). As it is observed in this table, the angle between constraints in easier instances are less than higher ones. So, this feature is contributing in problem difficulty.

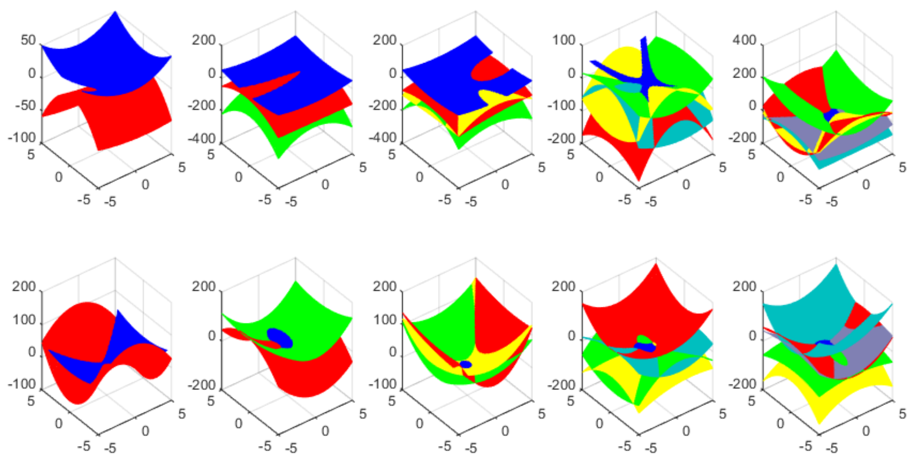

Figure 3: Easy (first row) and hard (second row) instances for 1 to 5 number of quadratic constraints using εDEg (2 dimension). The dark blue hyperplane is the feasible solution

Table 1: The angle feature for Sphere objective function

	Cons 1,2	Cons 1,3	Cons 1,4	Cons 1,5	Cons 2,3	Cons 2,4	Cons 2,5	Cons 3,4	Cons 3,5	Cons 4,5
DE Easy	15	17	25	21	32	27	41	47	45	43
DE Hard	45	51	63	59	62	73	76	69	79	86

Table 2: The FEN for linear constraints

Constraint - Function	Easy Instance	Hard Instance
1 c Sphere	25.6K	91.2K
2 c Sphere	28.9K	93.4K
3 c Sphere	32.4K	98.3K
4 c Sphere	34.2K	104.2K
5 c Sphere	35.5K	123.2K
1 c Ackley	65.2K	232.1K
2 c Ackley	69.3K	243.7K
3 c Ackley	74.2K	265.4K
4 c Ackley	86.4K	271.3K
5 c Ackley	92.3K	277.2K
1 c Rosenbrock	32.8K	145.2K
2 c Rosenbrock	35.9K	153.3K
3 c Rosenbrock	34.5K	167.9K
4 c Rosenbrock	42.2K	172.4K
5 c Rosenbrock	48.3K	176.8K
1 c Schaffer	84.8K	247.1K
2 c Schaffer	87.9K	259.1K
3 c Schaffer	93.5K	280.3K
4 c Schaffer	103.2K	293.8K
5 c Schaffer	112.4K	297.4K

Table 3: The FEN for quadratic constraints

Constraint - Function	Easy Instance	Hard Instance
1 c Sphere	24.2K	129.3K
2 c Sphere	25.3K	132.6K
3 c Sphere	27.9K	136.2K
4 c Sphere	34.1K	141.2K
5 c Sphere	38.7K	149.3K
1 c Ackley	68.4K	228.3
2 c Ackley	72.9K	232.5K
3 c Ackley	84.5K	239.6K
4 c Ackley	95.3K	247.9K
5 c Ackley	98.1K	251.9K
1 c Rosenbrock	31.4K	173.2K
2 c Rosenbrock	32.45K	182.3K
3 c Rosenbrock	42.5K	190.6K
4 c Rosenbrock	52.7K	192.8K
5 c Rosenbrock	71.1K	213.4K
1 c Schaffer	91.3K	278.9K
2 c Schaffer	94.9K	283.1K
3 c Schaffer	103.7K	289.3K
4 c Schaffer	114.1K	296.1K
5 c Schaffer	123.4	300k

Table 2 explains the variation of number of constraints feature group. It is shown that the problem difficulty (required FEN for easy and hard instances) has a strong systematic relationship with number of constraints for the experimented algorithm.

To calculate the optimum-local feasibility ratio, $10e^6$ points are generated within the vicinity of optimum (zero in our problems). Later, the ratio of feasible points to all generated points are investigated for easy and hard instances. Results point out that increasing number of linear constraints, decreases the feasibility ratio for experimented algorithms (see Table 4).

In summary the variation of feature values over the problem difficulty is more prominent in some of them than the other groups of features. Features such as, coefficients standard deviation, shortest distance, angle between constraints, number of constraints and feasibility ratio exhibit a relationship to problem hardness. This relationship is stronger for some features.

4.2 Analysis for Quadratic Constraints

In this section, we carry out our experiments on quadratic constraints. We use various objective functions, dimension and coefficient range similar to linear analysis. In the following the group of features are studied for easy and hard instances using quadratic constraints.

Table 4: Optimum-local feasibility ratio of search space near the optimum for 1,2,3,4 and 5 linear constraint

	DE Easy	DE Hard
1 cons	42%	7%
2 cons	32%	6%
3 cons	22%	4%
4 cons	17%	3%
5 cons	11%	2%

Observing the Figure 4, we can identify the relationship of quadratic coefficients and their ability to make problem hard or easy. Based on the experiments, quadratic coefficients has the ability to make problems hard or easier for algorithms. In other words, in each constraint, the quadratic coefficients (within the quadratic constraint) are more contributing to problem difficulty than linear coefficients (see Equation 5). Figure 4 shows the standard deviation of quadratic coefficients for easy and hard COPs. As shown, the standard deviation of quadratic coefficient in 1 to 5 constraints in easy instances are less than harder one. In contrast to quadratic coefficients, our experiments show there is no systematic relationship between the linear coefficient in quadratic constraints and problem hardness. In other words, quadratic coefficients are more contributing than linear ones in the same quadratic constraint.

Box plots shown in Figure 5 represent the shortest distance of a quadratic constraint hyperplanes to optimum. As it is observed, harder instances have constraint hyperplanes closer to optimum than easier ones. The lower values in these box plots means closer to optimum. Calculating the angles between constraints do not follow any systematic pattern and there is no relationship between angle feature and problem difficulty for quadratic constraints. We also study the number of quadratic constraints feature. As it is shown in Table 3, number of quadratic constraints is contributing to problem difficulty. It is obvious that increasing number of quadratic constraints makes a problem harder to solve (increases FEN). As observed in Table 5, investigations on feasibility ratio show that increasing number of constraint decreases the problem optimum-local feasibility ratio for easy and hard instances respectively. As it is observed, some group of features are more contributing to problem difficulty than the others. It is shown that angle feature does not follow any systematic relationship with problem hardness for experimented algorithm for quadratic constraints. On the other hand standard deviation, feasibility ratio and number of constraints are more contributing for εDEag.

4.3 Analysis for Combined Constraints

In this section, we consider the combination of linear and quadratic constraints. The generated COPs have different numbers of linear and quadratic constraints (5 constraints). The obtained results show the higher effectiveness of quadratic constraints. In other words, these constraints are more contributing to problem difficulty than linear

Table 5: Optimum-local feasibility ratio of search space near the optimum for 1,2,3,4 and 5 quadratic constraint

	DE Easy	DE Hard
1 cons	36%	11%
2 cons	27%	7%
3 cons	12%	4%
4 cons	11%	3%
5 cons	8%	2%

Table 6: The FEN for combined constraints using Sphere objective function

	DE Easy	DE Hard
1 Lin 4 Quad	22.4K	97.5K
2 Lin 3 Quad	17.5K	95.1K
3 Lin 2 Quad	16.5K	94.2K
4 Lin 1 Quad	14.1K	91.4K

ones. By analysing the various number of constraints (See Table 6) we can conclude that required FEN for sets of constraints with more quadratic ones is higher than sets with more linear constraints. This relationship holds the pattern for both easy and hard instances.

In summary it is observed that the variation of linear and quadratic constraint coefficients over the problem difficulty is more contributing for some group of features. Considering quadratic constraints only, it is obvious that some features such as angle do not provide useful knowledge for problem difficulty. In general, this experiments point out the relationship of the various constraint features of easy and hard instances with the problem difficulty while moving from easy to hard ones. This improves the understanding of the constraint structures and their ability to make a problem hard or easy for a specific group of evolutionary algorithms.

Conclusions

In this paper, we performed a feature-based analysis on the impact of sets of constraints (linear, quadratic and their combination) on performance of well-known evolutionary algorithm (εDEag). Various features of constraints for easy and hard instances have been analysed to understand which features contribute more to problem difficulty. The sets of constraints have been evolved using an evolutionary algorithm to generate hard and easy problem instances for εDEag. Furthermore, the relationship of the features with the problem difficulty have been examined while moving from easy to hard instances. Later on, these results can be used to design an algorithm prediction model.

Acknowledgements

Frank Neumann has been supported by ARC grants DP130104395 and DP140103400.

Bibliography

[1] C. A. Floudas and P. M. Pardalos. *A collection of test problems for constrained global optimization algorithms*, volume 455. Springer Science & Business Media, 1990.

[2] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2010: Experimental setup. 2010.

[3] R. Mallipeddi and P. N. Suganthan. Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. *Nanyang Technological University, Singapore*, 2010.

[4] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 829–836. ACM, 2011.

[5] O. Mersmann, M. Preuss, and H. Trautmann. Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *Parallel Problem Solving from Nature - PPSN XI, 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I*, volume 6238 of *Lecture Notes in Computer Science*, pages 73–82. Springer, 2010.

[6] E. Mezura-Montes and C. A. Coello Coello. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

[7] S. Nallaperuma, M. Wagner, F. Neumann, B. Bischl, O. Mersmann, and H. Trautmann. A feature-based comparison of local search and the christofides algorithm for the travelling salesperson problem. In *Proceedings of the twelfth workshop on Foundations of genetic algorithms XII*, pages 147–160. ACM, 2013.

[8] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of linear constraints for ε-constrained differential evolution. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 3088–3095. IEEE, 2014.

[9] S. Poursoltan and F. Neumann. Ruggedness quantifying for constrained continuous fitness landscapes. In *Evolutionary Constrained Optimization*, pages 29–50. Springer, 2015.

[10] K. Smith-Miles, J. van Hemert, and X. Y. Lim. Understanding tsp difficulty by learning from evolved instances. In *Learning and intelligent optimization*, pages 266–280. Springer, 2010.

[11] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[12] T. Takahama and Sakai. Constrained optimization by ε constrained differential evolution with dynamic ε-level control. In *Advances in Differential Evolution*, pages 139–154. Springer, 2008.

[13] T. Takahama and S. Sakai. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–9. IEEE, 2010.

[14] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Information characteristics and the structure of landscapes. *Evolutionary Computation*, 8(1):31–60, 2000.

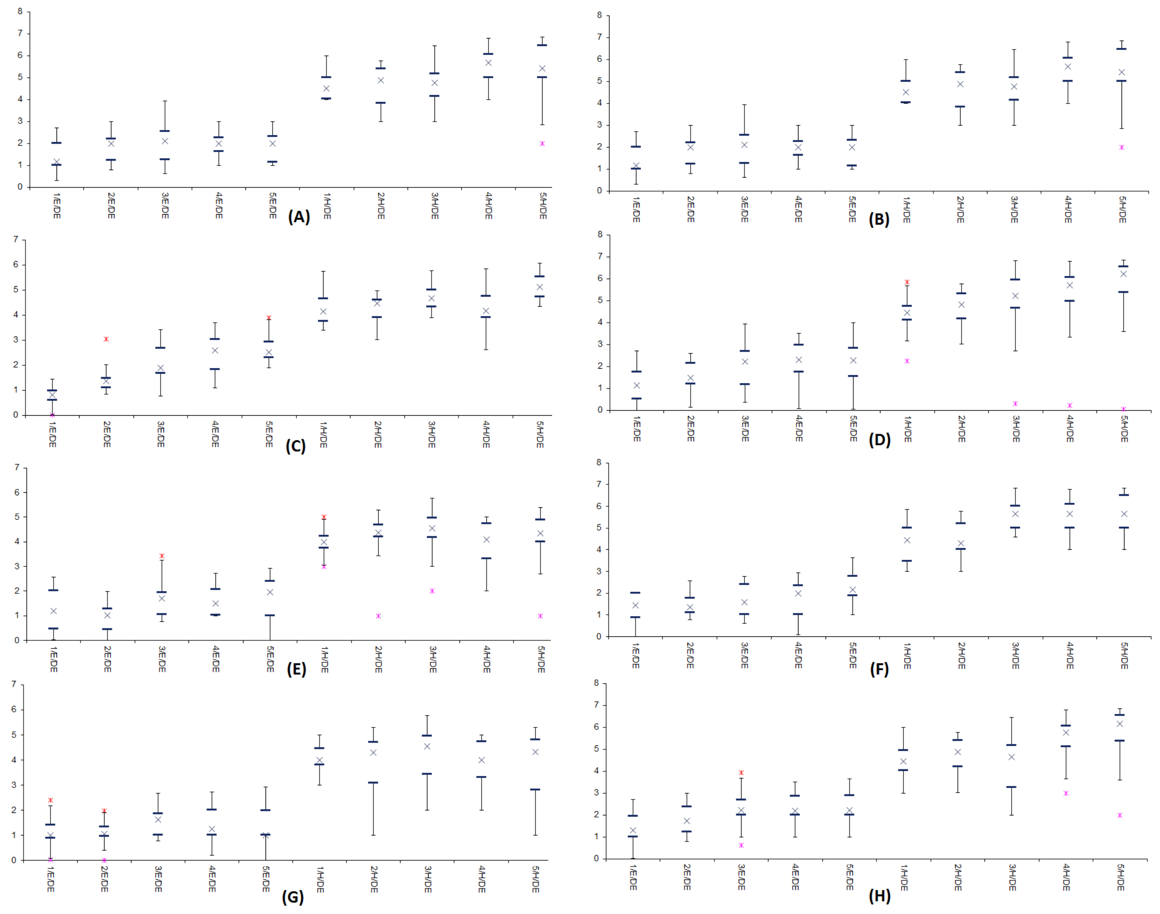

Figure 4: Box plot for standard deviation of coefficients in linear (A,C,E,G) and quadratic (B,D,F,H) constraints for Sphere (A,B), Ackley (C,D), Rosenbrok (E,F) and Schaffer (G,H). Each sub figure includes 2 sets of hard (H) and Easy (E) instances with 1 to 5 constraints using algorithms (a/b/c denotes a: constraint number, b: easy/hard instances and c:algorithm).

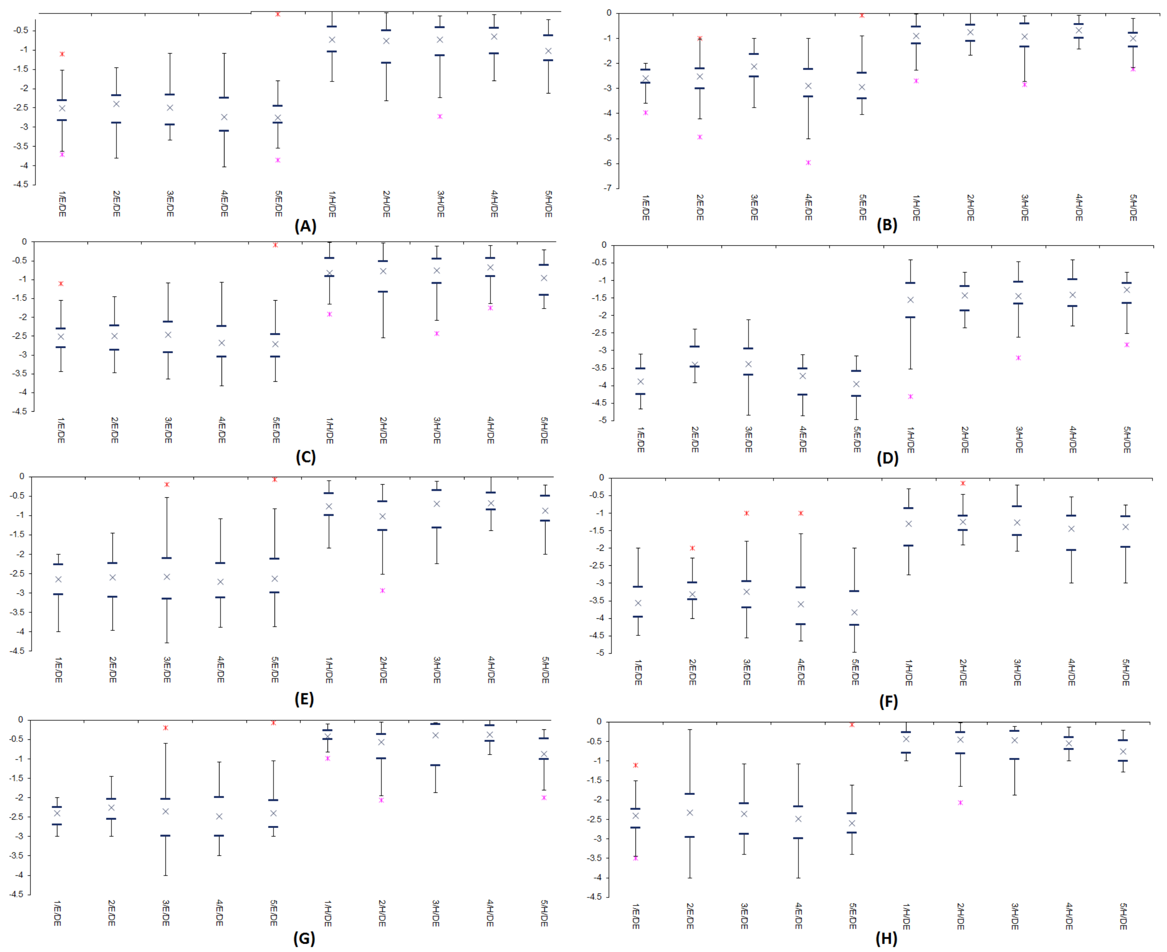

Figure 5: Box plot for the shortest distance to optimum of linear (A,C,E,G) and quadratic (B,D,F,H) constraints for Sphere (A,B), Ackley (C,D), Rosenbrok (E,F) and Schaffer (G,H). Each sub figure includes 2 sets of hard (H) and Easy (E) instances with 1 to 5 constraints using DE algorithm (a/b/c denotes a: constraint number, b: easy/hard instances and c:algorithm).

Chapter 5

A Feature-Based Comparison of Evolutionary Computing Techniques for Constrained Continuous Optimisation

In article included in this chapter [1], we first use single-objective evolver to analyse the features of sets of constraints and their influence on various types of algorithms. We also, for the first time, use multi-objective evolver to generate COP instances that are hard/easy for one and still easy/hard for the other algorithm types. By analysing how well an algorithm performs in conditions where other ones fail, we can derive its strengths and weaknesses over COPS. The results have been published in [1] and its extended version [2] is included in this chapter. This knowledge can help us to improve the efficiency of algorithm prediction model.

References

[1] S. Poursoltan and F. Neumann. A feature-based comparison of evolutionary computing techniques for constrained continuous optimisation. In S. Arik, T. Huang, W. K. Lai, and Q. Liu, editors, *Neural Information Processing*, volume 9491 of *Lecture Notes in Computer Science*, pages 332–343. Springer International Publishing, 2015.

[2] S. Poursoltan and F. Neumann. A feature-based comparison of evolutionary computing techniques for constrained continuous optimisation. *CoRR*, abs/1509.06842, 2015

Statement of Authorship

Title of Paper	A Feature-Based Comparison of Evolutionary Computing Techniques for Constrained Continuous Optimisation		
Publication Status	[✓ Published] Accepted for Publication Submitted for Publication Unpublished and Unsubmitted work written in manuscript style		
Publication Details	S. Poursoltan and F. Neumann. A feature-based comparison of evolutionary computing techniques for constrained continuous optimisation. In S. Arik, T. Huang, W. K. Lai, and Q. Liu, editors, Neural Information Processing, volume 9491 of Lecture Notes in Computer Science, pages 332–343. Springer International Publishing, 2015.		

Principal Author

Name of Principal Author (Candidate)	Shayan Poursoltan		
Contribution to the Paper	Came up with the idea, read the existing articles, categorised them, tested the idea to confirm its efficiency, wrote the first draft and applied comments from Co-author. Also, principal author presented the paper in the conference.		
Overall percentage (%)	85%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	3/2/2016

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Frank Neumann		
Contribution to the Paper	Supervised development of work. Read the paper and provided comment including editorial comments and conceptual feedback to improve the paper.		
Signature		Date	3/2/2016

Name of Co-Author			
Contribution to the Paper			
Signature		Date	

Please cut and paste additional co-author panels here as required.

Statement of Authorship

Title of Paper	A Feature-Based Comparison of Evolutionary Computing Techniques for Constrained Continuous Optimisation
Publication Status	[✓ Published] Accepted for Publication Submitted for Publication Unpublished and Unsubmitted work written in manuscript style
Publication Details	S. Poursoltan and F. Neumann. A feature-based comparison of evolutionary computing techniques for constrained continuous optimisation. CoRR, abs/1509.06842, 2015. This paper is the extended version of published conference paper.

Principal Author

Name of Principal Author (Candidate)	Shayan Poursoltan		
Contribution to the Paper	Came up with the idea, read the existing articles, categorised them, tested the idea to confirm its efficiency, wrote the first draft and applied comments from Co-author. Also, principal author presented the paper in the conference.		
Overall percentage (%)	85%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	3/2/2016

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Frank Neumann		
Contribution to the Paper	Supervised development of work. Read the paper and provided comment including editorial comments and conceptual feedback to improve the paper.		
Signature		Date	3/2/2016

Name of Co-Author			
Contribution to the Paper			
Signature		Date	

Please cut and paste additional co-author panels here as required.

A Feature-Based Comparison of Evolutionary Computing Techniques for Constrained Continuous Optimisation

Shayan Poursoltan
Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, Australia

Frank Neumann
Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, Australia

September 24, 2015

Abstract

Evolutionary algorithms have been frequently applied to constrained continuous optimisation problems. We carry out feature based comparisons of different types of evolutionary algorithms such as evolution strategies, differential evolution and particle swarm optimisation for constrained continuous optimisation. In our study, we examine how sets of constraints influence the difficulty of obtaining close to optimal solutions. Using a multi-objective approach, we evolve constrained continuous problems having a set of linear and/or quadratic constraints where the different evolutionary approaches show a significant difference in performance. Afterwards, we discuss the features of the constraints that exhibit a difference in performance of the different evolutionary approaches under consideration.

1 Introduction

There have been many algorithmic approaches proposed to solve complex optimisation problems, including constrained optimisation problems (COP). Several approaches have been proposed to tackle the constraints in constrained problems. Most of the research has been focused on introducing differential evolution (DE) [14], particle swarm optimisation (PSO) [2] and evolutionary strategies (ES) [13] to solve numerical optimisation problems. In order to deal with these constrained problems, there have been techniques that applied to these algorithms such as penalty functions, special operators (separating the constraint and objective function treatment) and decoder based methods. We refer the reader for a survey of constraint handling techniques in evolutionary computing methods to [9].

In order to compare and evaluate the evolutionary algorithms many approaches have been used. One is finding which algorithm performs better on a set of continuous problems using benchmarks sets [3, 6]. Recently, there has been an increasing interest to analyse the problem features that make it hard to solve. Initial studies have been carried out in the field of continuous optimisation in [8]. Furthermore, there have been techniques that generate a variation of problem instances from easy to hard. Then, the features of this problem instances are analysed in order to find which of them make the problems hard or easy to solve. Generating the variety of problem instances from easy to hard ensures that the knowledge obtained from analysis is reliable.

Although there is not only a standalone feature that makes a problem hard to solve, but it is assumed that constraints are very important in constrained continuous problems. The evolving approach that has been used to analyse the constraint features and their effects on COP's difficulty is discussed in [10, 11]. The idea is to evolve constrained problem instances (by using an evolutionary algorithm) in order to identify the constraint features with more contribution to problem difficulty.

In this paper, by using a single-objective evolutionary algorithm, we generate hard and easy COP instances for DE, ES and PSO algorithms. Later, we solve the generated instances using one algorithm by the other algorithms. The results show that the hardest generated instances using one algorithm are still hard for the other ones. To get better insight, we use multi-objective evolving approach to generate instances that are hard for one algorithm but still easy for the others. By analysing how an algorithm fails in conditions where the rest perform well, we can derive its strengths and weaknesses over constraint features. Our study shows the effectiveness of constraint features that make the problems hard for one and easy for the other algorithms. It can be translated as over which features of constraints, they make the problems hard for a certain algorithm but still easy for the others.

The remainder of this paper is as follows: In Section 2 we introduce the concept of COPs. Then we discuss the evolver (single and multi-objective evolutionary approach) and the solver algorithms (DE, ES and PSO) we use in our experiments. In Section 3 we analyse the performance of various algorithms on each others hard and easy instances (using the single-objective evolver). Section 4 includes the multi-objective approach that generates hard instances for one but easy for the other algorithms. Furthermore, we carry out the analysis of linear and quadratic constraint features that make the problem hard for one and still easy for the rest. Finally, we conclude with some remarks.

2 Preliminaries

2.1 Constrained continuous optimisation problems

In this study, constrained continuous optimisation problems with inequality and equality constraints are investigated. These problems are optimisation problems where a function $f(x)$ should be optimised with respect to a given set of constraints.

Single-objective functions $f\colon S \to \mathbb{R}$ with $S \subseteq \mathbb{R}^n$ are considered in this research. The constraints impose a feasible subset $F \subseteq S$ of the search space S and the aim is finding $x \in S \cap F$ which minimises f. Formally, we state the problems as follows:

$$\begin{array}{ll} \text{minimize} & f(x), \quad x = (x_1, \dots, x_n) \in \mathbb{R}^n \\ \text{subject to} & g_i(x) \leq 0 \quad \forall i \in \{1, \dots, q\} \\ & h_j(x) = 0 \quad \forall j \in \{q+1, \dots, p\} \end{array} \tag{1}$$

where $x = (x_1, x_2, \dots, x_n)$ is an n dimensional vector and $x \in S \cap F$. The $g_i(x)$ (inequality) and $h_j(x)$ (equality) constraints could be linear/nonlinear. Also, the equality constraints are usually replaced by $|h_j(x)| \leq \varepsilon$ where $\varepsilon = 10e^{-4}$ [6]. The feasible region $F \subseteq S$ of the search space S is defined by

$$l_i \leq x_i \leq u_i, \qquad 1 \leq i \leq n \tag{2}$$

where l_i and u_i denote lower and upper bounds respectively for the ith variable in which $1 \leq i \leq n$. In this paper, we focus on the ability of constraints (linear, quadratic) to make a problem hard or easy. The features of these constraints and their effect on problem difficulty is discussed. The constraints are of the following form:

$$\text{linear constraint} \quad g(x) = b + a_1 x_1 + \dots + a_n x_n \tag{3}$$

$$\text{quadratic constraint} \quad g(x) = b + a_1 x_1^2 + a_2 x_1 \dots + a_{2n-1} x_n^2 + a_{2n} x_n \tag{4}$$

or a combination of them, where $x_1, x_2 \dots, x_n$ are values from Equation 1 and $a_1, a_2, \dots, a_n$ are coefficients within lower (l_i) and upper bounds (u_i). We assume univariant quadratic function to analyse each x_n (with exponent 2) independently. Also, unvivarient quadratic constraints are more popular in recent benchmarks [6]. In order to include the optimum of objective function in feasible area, we set $b \leq 0$ (we assume the objective function optimum is zero).

2.2 Algorithms

We now introduce the algorithms for constrained continuous optimisation that are subject to our investigation.

One of the most prominent evolutionary algorithms for COPs is ε-constrained differential evolution with an archive and gradient-based mutation (εDEag). The algorithm is the winner of 2010 CEC competition for continuous COPs [6]. The εDEag uses ε-constrained method to transform algorithms for unconstrained problems to constrained ones. It adopts ε-level comparison to order the possible solutions. In other words, the lexicographic order is used in which constraint violation ($\phi(x)$) has more priority and proceeds the function value ($f(x)$). For more details we refer the reader to [16].

The second algorithm we use in this paper is a $(1+1)$ CMA-ES for constrained optimisation [1]. The $(1+1)$ CMA-ES in [4] is a variant of $(1+1)$-ES which adapts the covariance matrix of its offspring distribution in addition to its global step size. The idea behind the constraint handling approach of this algorithm is to obtain approximations to the normal vectors directions in the vicinity of the current solutions locations by low-pass filtering steps which violates the respective constraints and reducing the variance of the offspring distribution in these directions. Incorporating this constraint handling approach with $(1+1)$ CMA-ES makes an algorithm which is significantly more efficient than other approaches for constrained evolutionary algorithms. Also, the selected algorithm is not sensitive to the rotation of the problem search space. We refer the reader to [1] for more details and implementation.

The third algorithm that is used in our investigation is a particle swarm optimisation. This algorithm (HMPSO) applies a method that uses parallel search operator in which it divides the current swarm into various sub-swarms and locates the solution between them. In each sub-swarm, all particles follow the local best (fittest particle) which improves them to be more fitter. Also, since all sub-swarms are located around different optima (in parallel), then it is more possible to locate multiple optima which improves the diversity of algorithm. Dividing the swarms into sub-swarms improves the diversity of the algorithm. Also, choosing the local best in each sub-swarm can attract the other particles to fitter positions. We refer the reader to [17] for detailed algorithm and implementation.

2.3 Features of Constraints

In this paper we analyse the constraint features of generated problem instances. These features are constraint coefficients relationships such as standard derivation, angle between constraint hyperplanes, feasibility ratio in vicinity of optimum, number of constraints, shortest distance of constraint hyperplane to optimum. The details of these features are discussed in [11].

3 Single-objective Investigations

We first consider different algorithms and compare their relative performance on each other's generated hard and easy instances. We use single-objective evolver to evolve and generate hard and easy instances for all types of algorithms. The detailed procedure and results for DE instances are discussed in [11]. For this experiment, we perform 30 independent runs generating easy and hard instances for PSO and ES solvers. It means, the single-objective evolver only generates instances that are hard/easy for one type of algorithm (PSO, ES and DE). The required function evaluation number (FEN) for solving these instances (PSO, ES and DE) is used as fitness value for single-objective evolver. The parameters for solvers are identical to [1, 16, 17]. Also, we run our experiments on Sphere function (bowl shaped)[3]. We now have three groups of easy and hard instances generated for DE, ES and PSO algorithms. We then compare the DE, ES and PSO algorithms by applying them on each other's easy and hard instances. The analysis is done by comparing the required FEN for an algorithm to solve the other's generated problem instances. Then, it is possible to derive strengths and weaknesses of the considered algorithms by observing how well one algorithm performs in conditions where the other algorithms fail (or it is difficult for them). Table 1 and 2 show different algorithms performance on Sphere objective functions with linear/quadratic constraints (1 to 5 constraints). We also run our experiments on different objective functions such as Ackley and Rosenbrock. The results are shown in Tables 3, 4, 5 and 6. It is interesting that all objective functions follow similar pattern. Considering the required FEN to solve each instances, it is observed that hard instances are still the hardest for their own algorithms and hard for the others. It implies that the hard instances share some common features to make it difficult to solve for all solvers. However, the obtained knowledge is not enough to compare the algorithm capabilities to solve hard problem instances.

4 Multi-objective Investigations

Based on the experiment results in previous section, hard instances for each algorithm are still hard for the others. In order to extract more useful knowledge about the strengths or weaknesses of certain algorithms on constraint algorithms, we need problem instances that are hard for one and easy for the others. Analysing the features of these instances helps us extracting knowledge regarding the strengths and weaknesses of algorithms by examining why an algorithm performs better on some groups of features while the others fails. This will help us developing more efficient prediction model for automated algorithm selection.

To do this, we use a multi-objective DE algorithm (DEMO) described in [12] to minimise the FEN for one algorithm and maximise it for the others. In other words, the FEN for generated problem instances is higher (harder) for a certain algorithm and lower (easier) for the others. In order to find instances that are hard for one algorithm type and easy for the others, we need to find solution as diverse as possible. Also, the solutions need to be close to pareto front. Satisfying these two aims makes us to use multi-objective evolutionary algorithm to generate problem instances. Hence, we use differential evolution for multi-objective optimisation (DEMO) proposed by Robic in [12]. Based on results in [12], the DEMO achieves efficiently the above two goals. In DEMO, the candidate solution replaces parent when it dominates it and if the parent dominates it, the candidate is discarded. Otherwise, if the candidate and parent cannot dominate each other, the candidate is added to the population. The major difference between DEMO and other multi-objective evolutionary algorithms is that the newly generated good candidates are immediately used in creation of the subsequent candidates. This improves fast convergence to the true pareto front, while the use of non-dominated sorting and crowding distance metric in truncation of the extended population promotes the uniform spread of solutions. We refer the reader to [12] for further details and implementation.

In the following, we discuss the results for algorithms performances comparison. We carry out 30 independent runs for each number of constraints that are hard for one algorithm but still easy for the others. We set the evolving algorithm (DEMO) generation number to 5000 and the other parameters of evolving algorithm are set to pop size = 40, CR = 0.5, scaling factor = 0.9 and FEN_{max} is $300K$. Values for these parameters have been obtained by optimising the performance of the evolving algorithm in order to achieve the more easier and harder problem instances. For each of three algorithms, their best parameters are chosen [3, 16, 17]. First, the (εDEg) algorithm parameters are considered as: generation number = 1500, pop size = 100, CR = 0.5, scaling factor = 0.5. Also, the parameters for e-constraint method are described in [11]. Moreover, for evolutionary strategy we perform $(1,7)$-ES algorithm with 1500 generation using $P_f = 0.4$ with tendency to focus on feasible solution. In HMPSO algorithm, the swarm size N is set to 60, each sub-swarm size (N_s) is 8 and all the PSO parameters are considered as Krohling and Coelho's PSO [5]. In order to solve generated COPs, HMPSO generation number is set to 1500. We need to say the parameters for the solvers are identical to those given in [1, 12, 16, 17]

In our all experiments, we generate set of problem instances that are hard to one algorithm and easy to the other ones. Tables 7, 8, 9, 10, 11 and 12 show the function evaluation number (FEN) required for each algorithm to solve DE/ES/PSO hard instances for Sphere, Ackley and Rosenbrock objective functions (with 1 to 5 linear/quadratic constraints). As it is observed, there is more difference between the required FEN of instances generated by multi-objective algorithm evolver than the single-objective one. For instance, the required FEN for solving DE hard instances are higher for DE algorithm than solving it by ES and PSO algorithm. It means the DE hard instances are only hard for DE algorithm and easy for the others. In the following we start analysing constraint features of instances that are hard for one and easy for others.

4.1 Analysis for Linear Constraints

We run our experiments on Sphere, Ackley and Rosenbrock objective functions. The linear constraints are considered as in Equation 3 with all coefficients a_ns that are in the range of $[-5,5]$. Also, the problem dimension is set to 30. As it mentioned before, to analyse and discuss some features such as shortest distance, we assume that the optimum is zero ($b \leq 0$). We use three types of problem instances. DE hard denotes problem instances that are hard for DE algorithm but still easy for PSO and ES algorithms. Also, ES hard instances are easy for DE and PSO algorithm in this section. PSO hard means the instances that are hard for PSO but easy for the rest. Each constraint is generated using multi-objective evolver to generate instances that are hard for one algorithm but easy for others. In the following we discuss the features of linear constraints.

Figure 1 represents some evidence of linear constraint coefficient relationship (standard deviation). It is shown that standard deviation of (1 to 5) linear constraints are higher for DE hard instances than ES and PSO hard ones. This result is similar for all Sphere, Ackley and Rosenbrock objective functions. This means, the instances that are hard for DE algorithm but easy for ES and PSO have higher standard deviation for their constraints coefficients. In other words, this constraint feature has influence on problem difficulty. This improves the prediction ability for algorithm selection framework.

Box plots shown in Figure 2 represent the shortest distance from optimum feature for hard instances. Based on the experiments, hard instances for ES algorithm have higher value (closer to optimum) shortest distance than the other algorithms. It is noteworthy that lower value in Figure 2 means the constraint hyperplane is further from optimum. In other words, the constraints hyperplanes are closer to the optimum in ES hard instances. This relationship holds the pattern for all objective functions in linear constraints. We also study the feasibility ratio in vicinity of the optimum. As observed in Table 14, hard DE instances have lower feasibility ratio comparing to PSO and ES hard instances. This follows the same pattern for all experimented objective functions.

Table 1: The comparison of algorithms performance on each other's easy and hard instances based on required FEN for **Sphere** objective function with **linear** constraints. DE Easy (1 c) means instances that are easy for DE and with 1 constraint.

Instances	DE algorithm	ES algorithm	PSO algorithm
DE Easy (1 c)	25.6K	28.2K	33.2K
ES Easy (1 c)	26.3K	27.1K	33.9K
PSO Easy (1 c)	24.9K	29.1K	72.5K
DE Easy (2 c)	28.9K	21.9K	32.1K
ES Easy (2 c)	25.2K	24.3K	29.4K
PSO Easy (2 c)	24.2K	25.2K	33.5K
DE Easy (3 c)	32.4K	31.2K	33.9K
ES Easy (3 c)	31.8K	29.1K	33.2K
PSO Easy (3 c)	35.1K	28.6K	35.1K
DE Easy (4 c)	34.2K	29.8K	38.2K
ES Easy (4 c)	32.1K	31.5K	36.1K
PSO Easy (4 c)	35.7K	28.9K	39.5K
DE Easy (5 c)	35.3K	42.1K	46.4K
ES Easy (5 c)	31.2K	45.2K	38.2K
PSO Easy (5 c)	35.3K	44.9K	41.2K
DE Hard (1 c)	91.2K	78.3K	76.4K
ES Hard (1 c)	81.3K	86.4K	78.8K
PSO Hard (1 c)	82.5K	72.5K	85.4K
DE Hard (2 c)	93.4K	81.3K	81.4K
ES Hard (2 c)	84.3K	92.6K	79.4K
PSO Hard (2 c)	85.7K	84.1K	89.4K
DE Hard (3 c)	98.3K	93.8K	78.9K
ES Hard (3 c)	91.4K	108.6K	81.2K
PSO Hard (3 c)	89.1K	98.2K	91.6K
DE Hard (4 c)	104.2K	89.4K	82.5K
ES Hard (4 c)	89.4K	115.1K	78.4K
PSO Hard (4 c)	92.9K	93.5K	115.3K
DE Hard (5 c)	123.2K	111.4K	98.4K
ES Hard (5 c)	98.2K	133.2K	94.9K
PSO Hard (5 c)	101.3K	109.2K	118.3K

Table 2: The comparison of algorithms performance on each other's easy and hard instances based on required FEN for **Sphere** objective function with **quadratic** constraints. DE Easy (1 c) means instances that are easy for DE and with 1 constraint.

Instances	DE algorithm	ES algorithm	PSO algorithm
DE Easy (1 c)	24.2K	23.6K	24.9K
ES Easy (1 c)	24.8K	24.2K	25.4K
PSO Easy (1 c)	26.4K	25.4K	26.4K
DE Easy (2 c)	25.3K	28.1K	26.4K
ES Easy (2 c)	24.1K	27.2K	27.4K
PSO Easy (2 c)	23.5K	29.3K	271.K
DE Easy (3 c)	27.9K	31.9K	35.5K
ES Easy (3 c)	29.4K	32.1K	28.5K
PSO Easy (3 c)	28.1K	28.7K	29.4K
DE Easy (4 c)	34.1K	28.9K	36.4K
ES Easy (4 c)	35.2K	35.3K	31.6K
PSO Easy (4 c)	31.8K	29.5K	33.2K
DE Easy (5 c)	38.7K	29.2K	37.2K
ES Easy (5 c)	35.6K	28.2K	39.5K
PSO Easy (5 c)	36.3K	31.5K	36.2K
DE Hard (1 c)	129.3K	102.7K	105.3K
ES Hard (1 c)	104.3K	121.2K	108.2K
PSO Hard (1 c)	108.2K	104.2K	119.8K
DE Hard (2 c)	132.6K	114.2K	114.9K
ES Hard (2 c)	111.2K	127.1K	112.4K
PSO Hard (2 c)	109.4K	112.4K	125.3K
DE Hard (3 c)	136.2K	116.3K	112.4K
ES Hard (3 c)	117.2K	132.1K	109.9K
PSO Hard (3 c)	119.8K	119.2K	132.6K
DE Hard (4 c)	141.2K	119.9K	119.6K
ES Hard (4 c)	113.8K	131.2K	121.9K
PSO Hard (4 c)	115.4K	121.4K	138.9K
DE Hard (5 c)	149.3K	129.7K	122.9K
ES Hard (5 c)	124.4K	149.6K	126.4K
PSO Hard (5 c)	123.9K	124.2K	148.3K

Table 3: The comparison of algorithms performance on each other's easy and hard instances based on required FEN for **Achkley** objective function with **linear** constraints. DE Easy (1 c) means instances that are easy for DE and with 1 constraint.

Instances	DE algorithm	ES algorithm	PSO algorithm
DE Easy (1 c)	39.2K	37.1K	37.4K
ES Easy (1 c)	38.6K	37.8K	38.1K
PSO Easy (1 c)	41.3K	39.2K	41.7K
DE Easy (2 c)	40.3K	41.5K	42.6K
ES Easy (2 c)	41.3K	42.5K	41.5K
PSO Easy (2 c)	42.6K	41.2K	44.7K
DE Easy (3 c)	47.3K	48.2K	46.9K
ES Easy (3 c)	48.9K	47.3K	46.3K
PSO Easy (3 c)	49.2K	51.6K	50.9K
DE Easy (4 c)	48.9K	47.2K	49.2K
ES Easy (4 c)	49.2K	48.2K	50.1K
PSO Easy (4 c)	51.2K	50.5K	52.6K
DE Easy (5 c)	51.3K	52.7K	51.6K
ES Easy (5 c)	52.1K	52.6K	50.7K
PSO Easy (5 c)	55.3K	54.7K	52.3K
DE Hard (1 c)	107.3K	83.2K	85.6K
ES Hard (1 c)	82.3K	105.2K	81.6K
PSO Hard (1 c)	85.2K	83.9K	106.3K
DE Hard (2 c)	114.2K	88.2K	91.5K
ES Hard (2 c)	87.3K	115.3K	88.8K
PSO Hard (2 c)	89.2K	87.3K	116.9K
DE Hard (3 c)	119.8K	94.1K	93.9K
ES Hard (3 c)	95.2K	121.6K	94.2K
PSO Hard (3 c)	93.2K	95.1K	121.5K
DE Hard (4 c)	125.2K	99.2K	101.4K
ES Hard (4 c)	101.3K	126.3K	98.2K
PSO Hard (4 c)	99.4K	97.8K	127.4K
DE Hard (5 c)	132.5K	102.2K	101.5K
ES Hard (5 c)	101.4K	134.7K	103.5K
PSO Hard (5 c)	103.9K	102.4K	131.4K

Table 4: The comparison of algorithms performance on each other's easy and hard instances based on required FEN for **Ackley** objective function with **quadratic** constraints. DE Easy (1 c) means instances that are easy for DE and with 1 constraint.

Instances	DE algorithm	ES algorithm	PSO algorithm
DE Easy (1 c)	38.1K	39.4K	37.2K
ES Easy (1 c)	37.1K	38.1K	39.0K
PSO Easy (1 c)	41.2K	39.9K	40.7K
DE Easy (2 c)	38.9K	43.9K	41.7K
ES Easy (2 c)	40.1K	41.2K	43.2K
PSO Easy (2 c)	39.1K	43.1K	46.1K
DE Easy (3 c)	46.3K	47.9K	45.1K
ES Easy (3 c)	49.1K	48.1K	47.2K
PSO Easy (3 c)	42.1K	45.1K	49.1K
DE Easy (4 c)	49.2K	48.7K	48.4K
ES Easy (4 c)	49.7K	49.9K	52.9K
PSO Easy (4 c)	56.1K	55.1K	54.1K
DE Easy (5 c)	50.0K	51.2K	52.4K
ES Easy (5 c)	51.3K	56.2K	54.1K
PSO Easy (5 c)	61.2K	58.9K	59.1K
DE Hard (1 c)	133.4K	93.1K	94.6K
ES Hard (1 c)	92.1K	135.1K	94.1K
PSO Hard (1 c)	95.2K	94.9K	134.2K
DE Hard (2 c)	139.1K	98.2K	97.1K
ES Hard (2 c)	97.1K	138.2K	99.1K
PSO Hard (2 c)	109.1K	107.2K	145.2K
DE Hard (3 c)	145.3K	112.6K	109.6K
ES Hard (3 c)	111.6K	141.2K	108.9K
PSO Hard (3 c)	111.2K	109.2K	152.K
DE Hard (4 c)	167.2K	132.1K	135.1K
ES Hard (4 c)	131.1K	167.9K	133.9K
PSO Hard (4 c)	132.1K	133.2K	169.1K
DE Hard (5 c)	177.1K	143.2K	131.3K
ES Hard (5 c)	141.2K	181.2K	144.9K
PSO Hard (5 c)	139.1K	142.9K	182.1K

Table 5: The comparison of algorithms performance on each other's easy and hard instances based on required FEN for **Rosenbrock** objective function with **linear** constraints. DE Easy (1 c) means instances that are easy for DE and with 1 constraint.

Instances	DE algorithm	ES algorithm	PSO algorithm
DE Easy (1 c)	38.1K	37.4K	39.1K
ES Easy (1 c)	39.2K	38.9K	36.2K
PSO Easy (1 c)	44.6K	40.1K	43.1K
DE Easy (2 c)	41.2K	42.4K	47.1K
ES Easy (2 c)	40.2K	45.2K	40.4K
PSO Easy (2 c)	43.9K	42.6K	44.8K
DE Easy (3 c)	41.2K	42.3K	45.4K
ES Easy (3 c)	47.7K	48.1K	47.4K
PSO Easy (3 c)	48.3K	52.4K	53.1K
DE Easy (4 c)	44.1K	42.7K	43.3K
ES Easy (4 c)	48.4K	49.7K	52.6K
PSO Easy (4 c)	53.5K	53.1K	54.9K
DE Easy (5 c)	55.5K	53.2K	52.1K
ES Easy (5 c)	52.8K	55.4K	52.1K
PSO Easy (5 c)	58.2K	53.4K	52.8K
DE Hard (1 c)	110.2K	83.7K	85.2K
ES Hard (1 c)	81.5K	107.2K	82.1K
PSO Hard (1 c)	86.9K	85.3K	108.2K
DE Hard (2 c)	116.6K	89.3K	92.1K
ES Hard (2 c)	88.2K	117.5K	87.0K
PSO Hard (2 c)	90.8K	88.1K	114.5K
DE Hard (3 c)	121.2K	92.1K	92.5K
ES Hard (3 c)	94.1K	124.8K	93.2K
PSO Hard (3 c)	94.7K	96.5K	125.2K
DE Hard (4 c)	126.1K	101.6K	104.2K
ES Hard (4 c)	104.8K	127.2K	96.1K
PSO Hard (4 c)	100.2K	92.1K	123.7K
DE Hard (5 c)	135.1K	109.5K	106.8K
ES Hard (5 c)	105.2K	136.1K	105.1K
PSO Hard (5 c)	102.1K	106.8K	131.4

Table 6: The comparison of algorithms performance on each other's easy and hard instances based on required FEN for **Rosenbrock** objective function with **quadratic** constraints. DE Easy (1 c) means instances that are easy for DE and with 1 constraint.

Instances	DE algorithm	ES algorithm	PSO algorithm
DE Easy (1 c)	41.2K	40.2K	38.7K
ES Easy (1 c)	39.2K	39.3K	36.1K
PSO Easy (1 c)	43.1K	39.6K	42.2K
DE Easy (2 c)	39.1K	44.4K	43.2K
ES Easy (2 c)	42.6K	44.5K	41.8K
PSO Easy (2 c)	41.2K	45.6K	47.2K
DE Easy (3 c)	47.1K	48.5K	49.2K
ES Easy (3 c)	46.8K	49.5K	48.8K
PSO Easy (3 c)	46.2K	42.7K	48.4K
DE Easy (4 c)	48.7K	49.1K	51.2K
ES Easy (4 c)	50.2K	52.4K	55.2K
PSO Easy (4 c)	59.2K	54.5K	51.9K
DE Easy (5 c)	52.5K	56.1K	55.7K
ES Easy (5 c)	56.0K	55.7K	53.8K
PSO Easy (5 c)	66.3K	59.8K	60.4K
DE Hard (1 c)	137.2K	93.7K	92.1K
ES Hard (1 c)	93.7K	138.2K	99.2K
PSO Hard (1 c)	93.1K	96.2K	138.0K
DE Hard (2 c)	142.7K	99.7K	98.4K
ES Hard (2 c)	98.8K	141.6K	101.4K
PSO Hard (2 c)	112.7K	109.5K	148.1K
DE Hard (3 c)	148.2K	115.3K	112.8K
ES Hard (3 c)	114.1K	144.8K	113.2K
PSO Hard (3 c)	113.6K	113.1K	157.3K
DE Hard (4 c)	171.7K	136.7K	133.4K
ES Hard (4 c)	134.7K	168.2K	135.3K
PSO Hard (4 c)	134.7K	139.3K	172.6K
DE Hard (5 c)	179.6K	146.1K	144.8K
ES Hard (5 c)	143.8K	185.1K	147.4K
PSO Hard (5 c)	143.7K	141.4K	186.4K

Table 7: The FEN required for each algorithm to solve DE/ES/PSO hard instances (Sphere for 1 to 5 **linear constraints**)

Instances	DE algorithm	ES algorithm	PSO algorithm
DE hard (1 c)	**86.3K**	41.5K	43.2K
ES hard (1 c)	45.7K	**84.2K**	48.3K
PSO hard (1 c)	37.2K	41.8K	**80.1K**
DE hard (2 c)	**88.8K**	43.9K	44.2K
ES hard (2 c)	45.9K	**85.4K**	46.3K
PSO hard (2 c)	43.2K	42.5K	**82.9K**
DE hard (3 c)	**91.4K**	44.6K	45.3K
ES hard (3 c)	49.2K	**87.8K**	48.1K
PSO hard (3 c)	46.2K	47.7K	**85.5K**
DE hard (4 c)	**94.2K**	47.5K	47.8K
ES hard (4 c)	51.7K	**89.1K**	50.1K
PSO hard (4 c)	48.7K	49.9K	**87.3K**
DE hard (5 c)	**96.2K**	48.2K	49.5K
ES hard (5 c)	52.4K	**90.4K**	53.5K
PSO hard (5 c)	49.6K	51.4K	**91.6K**

Table 8: TThe FEN required for each algorithm to solve DE/ES/PSO hard instances (Sphere for 1 to 5 **quadratic constraints**)

Instances	DE algorithm	ES algorithm	PSO algorithm
DE hard (1 c)	**92.3K**	50.2K	51.9K
ES hard (1 c)	48.8K	**91.3K**	49.3K
PSO hard (1 c)	44.5K	46.8K	**93.1K**
DE hard (2 c)	**93.5K**	52.9K	54.2K
ES hard (2 c)	50.9K	**95.9K**	51.2K
PSO hard (2 c)	50.2K	53.2K	**96.3K**
DE hard (3 c)	**95.9K**	54.3K	55.3K
ES hard (3 c)	53.9K	**97.4K**	52.4K
PSO hard (3 c)	57.3K	56.3K	**98.9K**
DE hard (4 c)	**98.3K**	56.4K	57.3K
ES hard (4 c)	56.3K	**102.3K**	52.1K
PSO hard (4 c)	59.2K	58.2K	**101.6K**
DE hard (5 c)	**102.1K**	58.3K	59.4K
ES hard (5 c)	59.2K	**103.2K**	60.2K
PSO hard (5 c)	62.6K	63.8K	**105.2K**

Table 9: The FEN required for each algorithm to solve DE/ES/PSO hard instances (Ackley for 1 to 5 **linear constraints**)

Instances	DE algorithm	ES algorithm	PSO algorithm
DE hard (1 c)	**102.3K**	46.1K	51.4K
ES hard (1 c)	51.2K	**104.7K**	50.2K
PSO hard (1 c)	47.4K	49.8K	**107.4K**
DE hard (2 c)	**112.1K**	56.1K	54.1K
ES hard (2 c)	53.9K	**115.9K**	48.6K
PSO hard (2 c)	55.5K	55.3K	**117.2K**
DE hard (3 c)	**126.1K**	63.7K	65.2K
ES hard (3 c)	59.1K	**128.3K**	58.7K
PSO hard (3 c)	61.7K	62.8K	**134.2K**
DE hard (4 c)	**124.9K**	68.4K	63.1K
ES hard (4 c)	64.1K	**129.8K**	59.2K
PSO hard (4 c)	67.5K	69.2K	**135.2K**
DE hard (5 c)	**138.8K**	75.2K	74.1K
ES hard (5 c)	71.2K	**137.1K**	76.7K
PSO hard (5 c)	73.1K	74.1K	**141.2K**

Table 10: The FEN required for each algorithm to solve DE/ES/PSO hard instances (Ackley for 1 to 5 **quadratic constraints**)

Instances	DE algorithm	ES algorithm	PSO algorithm
DE hard (1 c)	**142.5K**	60.1K	62.5K
ES hard (1 c)	58.5K	**148.2K**	61.4K
PSO hard (1 c)	53.2K	53.9K	**147.7K**
DE hard (2 c)	**153.3K**	58.1K	58.1K
ES hard (2 c)	59.2K	**155.5K**	59.2K
PSO hard (2 c)	57.8K	56.3K	**157.2K**
DE hard (3 c)	**167.3K**	65.2K	68.1K
ES hard (3 c)	63.2K	**169.2K**	69.8K
PSO hard (3 c)	65.7K	67.9K	**167.6K**
DE hard (4 c)	**174.8K**	71.2K	75.1K
ES hard (4 c)	66.8K	**169.1K**	72.9K
PSO hard (4 c)	69.1K	68.3K	**172.9K**
DE hard (5 c)	**179.5K**	75.1K	76.1K
ES hard (5 c)	72.8K	**174.9K**	77.4.2K
PSO hard (5 c)	75.1K	74.9K	**175.9K**

Table 11: The FEN required for each algorithm to solve DE/ES/PSO hard instances (Rosenbrock for 1 to 5 **linear constraints**)

Instances	DE algorithm	ES algorithm	PSO algorithm
DE hard (1 c)	**103.1K**	48.2K	53.9K
ES hard (1 c)	53.1K	**107.2K**	52.7K
PSO hard (1 c)	45.7K	48.1K	**109.2K**
DE hard (2 c)	**115.1K**	57.8K	55.7K
ES hard (2 c)	54.7K	**113.4K**	46.1K
PSO hard (2 c)	54.8K	54.9K	**119.5K**
DE hard (3 c)	**124.3K**	65.2K	62.1K
ES hard (3 c)	58.8K	**127.1K**	59.7K
PSO hard (3 c)	62.3K	65.5K	**136.1K**
DE hard (4 c)	**125.5K**	69.1K	65.2K
ES hard (4 c)	67.5K	**128.1K**	58.7K
PSO hard (4 c)	64.9K	70.6K	**137.1K**
DE hard (5 c)	**135.1K**	74.1K	74.7K
ES hard (5 c)	73.7K	**135.8K**	75.1K
PSO hard (5 c)	72.3K	76.5K	**140.9K**

Table 12: The FEN required for each algorithm to solve DE/ES/PSO hard instances (Rosenbrock for 1 to 5 **quadratic constraints**)

Instances	DE algorithm	ES algorithm	PSO algorithm
DE hard (1 c)	**143.1K**	61.4K	63.7K
ES hard (1 c)	59.6K	**149.7K**	62.5K
PSO hard (1 c)	54.3K	54.2K	**143.8K**
DE hard (2 c)	**155.2K**	59.2K	59.2K
ES hard (2 c)	61.3K	**154.8K**	57.9K
PSO hard (2 c)	59.2K	57.1K	**158.0K**
DE hard (3 c)	**168.9K**	63.7K	66.8K
ES hard (3 c)	65.2K	**170.1K**	68.1K
PSO hard (3 c)	63.7K	68.1K	**168.9K**
DE hard (4 c)	**175.1K**	73.8K	76.9K
ES hard (4 c)	68.2K	**172.7K**	75.2K
PSO hard (4 c)	67.7K	69.1K	**176.2K**
DE hard (5 c)	**180.2K**	74.2K	77.8K
ES hard (5 c)	74.2K	**175.1K**	79.2K
PSO hard (5 c)	73.6K	74.4K	**179.4K**

Also increasing the number of constraints decreases the problem optimum-local feasibility for all algorithm problem instances. The angle between linear constraints feature is analysed for linear constraints. As it is observed in Table 13, ES hard instances have lower angle values for all Sphere, Ackley and Rosenbrock objective functions. This means, instances that are hard for ES have less angle value between their constraint hyperplanes. Interestingly, all objective function that we use in this experiment follow the same relationship.

As it is observed, to compare the instances, DE hard instances have higher linear constraint coefficient standard deviation. It can be translated as DE algorithm has more difficulty to coefficients standard deviation feature than PSO and ES algorithms. Also, the local-optimum feasibility ratio value is higher in ES and PSO hard instances than DE hard ones. This means, ES and PSO algorithms are more effective to problems with higher optimum feasibility ratio feature. The shortest distance and angle features for ES is less than DE and PSO hard instances. Interestingly, this features are similar for all used objective functions. The linear constraint feature based analysis gives us helpful knowledge to implement algorithm selection framework.

4.2 Analysis for Quadratic Constraints

In this section, we carry out our experiments on Sphere, Ackley and Rosenbrock objective function with quadratic constraints (see Equation 4) using same setup as previous section. In the following we do feature based analysis of constraints in hard DE, PSO and ES instances (that are easy for the other algorithms).

Figure 1 shows some evidence of quadratic constraint coefficients relationship. Based on our experiments, in each constraint, the quadratic coefficient has more ability than linear coefficients to make problem harder to solve. In other words, in Equation 4, a_1 is more contributing than a_2 to problem difficulty. As it is shown in the box plots, the standard deviation of 1 to 5 quadratic constraints in DE hard instances are higher comparing the other two algorithm hard instances. In contrast, our results show no systematic relationship between problem difficulty and linear coefficients in each quadratic constraints and quadratic coefficients have more contribution in problem difficulty.

As it is observed in Figure 2, the shortest distance feature for DE, PSO and ES hard instances are compared. In instances that are hard for ES and easy for the other algorithms, the quadratic constraint hyperplanes are closer to optimum (zero). This applies to all experimented objective functions. Also, calculating the angle feature for quadratic constraint does not show any systematic relationship to problem difficulty. The feasibility ratio near the optimum is analysed for DE, ES and PSO hard instances. As it is shown in Table 15, the feasibility ratio in DE hard instances are lower than the other algorithms hard instances. All objective functions have the same pattern. Also, the number of constraint has a systematic relationship with feasibility ratio.

Based on the results, to compare COP instances with quadratic constraints, DE hard instances have higher coefficient standard deviation value than the other algorithm hard ones. It is translated as the DE algorithm has more difficulty solving instances with higher standard deviation value for their quadratic constraints than ES and PSO. Also, the quadratic constraints are closer to optimum in ES instances than the other experi-

Table 13: The angle feature for Sphere objective function for linear constraints

	Cons 1,2	Cons 1,3	Cons 1,4	Cons 1,5	Cons 2,3	Cons 2,4	Cons 2,5	Cons 3,4	Cons 3,5	Cons 4,5
DE Hard	74	64	63	58	74	71	68	59	62	86
ES Hard	33	21	37	24	44	46	39	46	48	51
PSO Hard	75	63	82	68	71	73	72	69	81	86

Table 14: Optimum-local feasibility ratio of search space near the optimum for 1,2,3,4 and 5 linear constraint

	1 cons	2 cons	3 cons	4 cons	5 cons
DE Hard	6%	5%	3%	3 %	2%
ES Hard	16%	11%	10 %	6%	5%
PSO Hard	17%	12%	11%	8%	5%

mented algorithms. In other words, ES algorithm is more influenced by constraint with closer to optimum instances. Moreover, the optimum feasibility ratio in DE instances are lower than PSO and ES.

5 Conclusion

In this paper, we carried out an algorithm performance comparison on each others constrained problem instances. We then analysed the features and characteristics of constraints that make them hard to solve for certain algorithm but easy for the others. It is observed that some constraint features are more contributing to problem difficulty for certain algorithms. In linear constraints, some features such as coefficient relationship, angle, local-optimum feasibility ratio and shortest distance play an important role in problem difficulty to DE and ES algorithms. Considering quadratic instances, angle does not show any relationship to problem difficulty.

By analysing how well one algorithm performs in conditions where other algorithms fail, we can derive its strengths and weaknesses over constrained problems. These results can help us to improve the efficiency of algorithm prediction model.

Acknowledgements

Frank Neumann has been supported by ARC grants DP130104395 and DP140103400.

References

[1] D. V. Arnold and N. Hansen. A (1+ 1)-cma-es for constrained optimisation. In *Proceedings of the 14th annual conference on Genetic and evolutionary compu-*

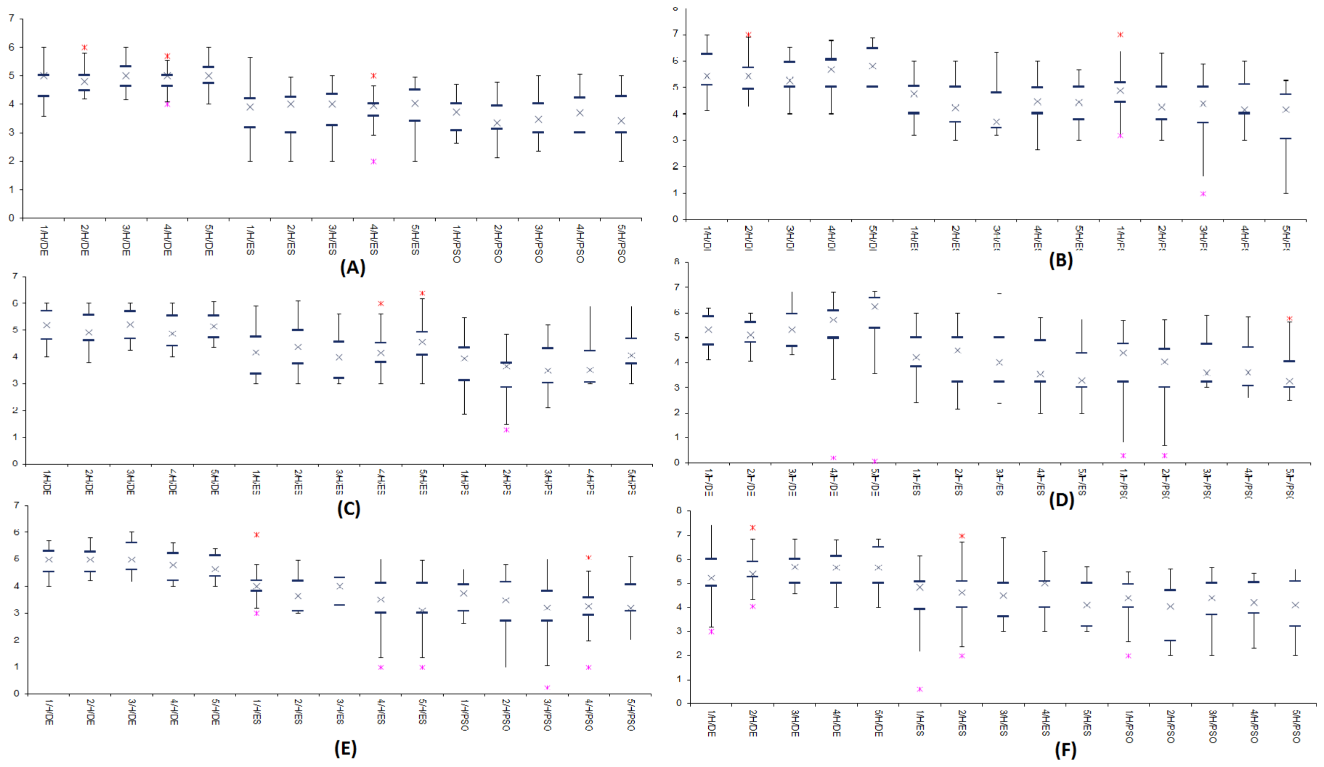

Figure 1: Box plot for standard deviation of coefficients in linear constraints with objective functions: Sphere (A), Ackley (C) and Rosenbrock (E) and quadratic constraints Sphere (B), Ackley (D) and Rosenbrock (F). Each sub figure includes hard instances (H) with 1 to 5 constraints using algorithms (a/b/c denotes a: number of constraints, b: hard instances and c: hard instances for DE/ES/PSO algorithm).

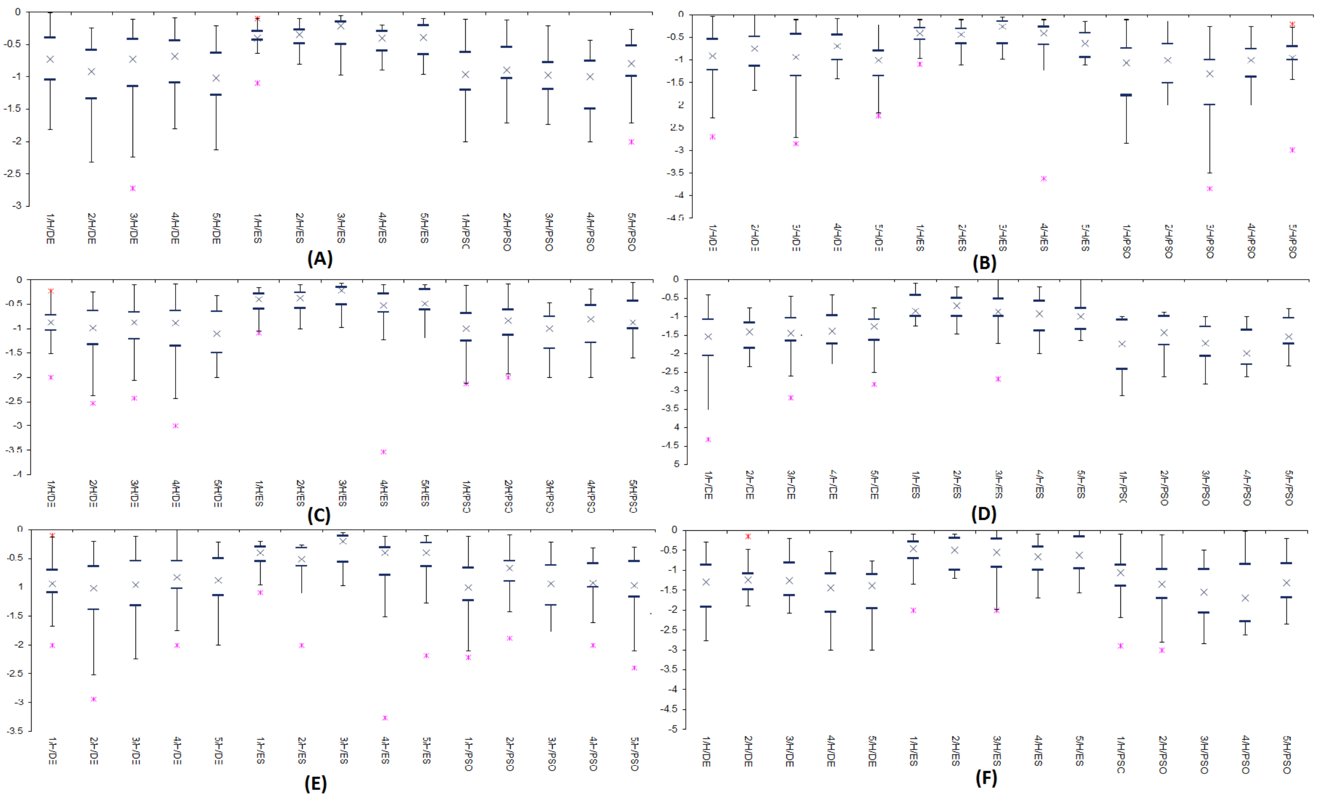

Figure 2: Box plot for shortest distance feature in linear constraints with objective functions: Sphere (A), Ackley (C) and Rosenbrock (E) and quadratic constraints Sphere (B), Ackley (D) and Rosenbrock (F). Each sub figure includes hard instances (H) with 1 to 5 constraints using algorithms (a/b/c denotes a: number of constraints, b: hard instances and c: hard instances for ES/PSO/DE algorithm).

Table 15: Optimum-local feasibility ratio of search space near the optimum for 1,2,3,4 and 5 quadratic constraint

	1 cons	2 cons	3 cons	4 cons	5 cons
DE Hard	4%	4%	3%	2 %	2%
ES Hard	14%	10%	8 %	7%	5%
PSO Hard	15%	10%	9%	8%	7%

tation, pages 297–304. ACM, 2012.

[2] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE, 1995.

[3] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2010: Experimental setup. 2010.

[4] C. Igel, T. Suttorp, and N. Hansen. A computational efficient covariance matrix update and a (1+ 1)-cma for evolution strategies. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 453–460. ACM, 2006.

[5] R. A. Krohling and L. dos Santos Coelho. Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(6):1407–1416, 2006.

[6] R. Mallipeddi and P. N. Suganthan. Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. *Nanyang Technological University, Singapore*, 2010.

[7] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 829–836. ACM, 2011.

[8] O. Mersmann, M. Preuss, and H. Trautmann. *Benchmarking evolutionary algorithms: Towards exploratory landscape analysis*. Springer, 2010.

[9] E. Mezura-Montes and C. A. Coello Coello. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

[10] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of linear constraints for ε-constrained differential evolution. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 3088–3095. IEEE, 2014.

[11] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of set of constraints for e-constrained differential evolution. *CoRR*, abs/1506.06848, 2015.

[12] T. Robič and B. Filipič. Demo: Differential evolution for multiobjective optimization. In *Evolutionary Multi-Criterion Optimization*, pages 520–533. Springer, 2005.

[13] H.-P. P. Schwefel. *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.

[14] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[15] T. Takahama and Sakai. Constrained optimization by ε constrained differential evolution with dynamic ε-level control. In *Advances in Differential Evolution*, pages 139–154. Springer, 2008.

[16] T. Takahama and S. Sakai. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–9. IEEE, 2010.

[17] Y. Wang and Z. Cai. A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems. *Frontiers of Computer Science in China*, 3(1):38–52, 2009.

Chapter 6

A Feature-Based Prediction Model of Algorithm Selection for Constrained Continuous Optimisation

The article in this chapter investigates the impact of different sets of evolved instances for building prediction models in the area of algorithm selection [1]. This article is an extended version of paper [2] that is already submitted for review. We used evolved instances to implement a prediction model that predict the best suited algorithm type for a given COP based on its constraint features. The training data is tested between various subsets of evolved instances from multi-objective evolver in previous chapter. In other words, the prediction model performance is analysed to find the best customisation of training data.

References

[1] S. Poursoltan and F. Neumann. A feature-based prediction model of algorithm selection for constrained continuous optimisation. *CoRR*, abs/1602.02862, 2016

[2] S. Poursoltan and F. Neumann. A feature-based prediction model of algorithm selection for constrained continuous optimisation. Accepted at The IEEE Congress on Evolutionary Computation 2016 (IEEE CEC 2016)

Statement of Authorship

Title of Paper	A Feature-Based Prediction Model of Algorithm Selection for Constrained Continuous Optimisation
Publication Status	☐ Published [✓ Accepted for Publication] ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	S. Poursoltan and F. Neumann. A feature-based prediction model of algorithm selection for constrained continuous optimisation. Accepted at The IEEE Congress on Evolutionary Computation 2016 (IEEE CEC 2016)

Principal Author

Name of Principal Author (Candidate)	Shayan Poursoltan		
Contribution to the Paper	Came up with the idea, read the existing articles, categorised them, tested the idea to confirm its efficiency, wrote the first draft and applied comments from Co-author.		
Overall percentage (%)	85%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	5/6/2016

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Frank Neumann		
Contribution to the Paper	Supervised development of work. Read the paper and provided comment including editorial comments and conceptual feedback to improve the paper.		
Signature		Date	5/6/2016

Name of Co-Author			
Contribution to the Paper			
Signature		Date	

Please cut and paste additional co-author panels here as required.

Statement of Authorship

Title of Paper	A Feature-Based Prediction Model of Algorithm Selection for Constrained Continuous Optimisation
Publication Status	[✓ Published] ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	S. Poursoltan and F. Neumann. A feature-based prediction model of algorithm selection for constrained continuous optimisation. CoRR, abs/1602.02862, 2016. This paper is the extended version of submitted paper.

Principal Author

Name of Principal Author (Candidate)	Shayan Poursoltan		
Contribution to the Paper	Came up with the idea, read the existing articles, categorised them, tested the idea to confirm its efficiency, wrote the first draft and applied comments from Co-author.		
Overall percentage (%)	85%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	3/2/2016

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Frank Neumann		
Contribution to the Paper	Supervised development of work. Read the paper and provided comment including editorial comments and conceptual feedback to improve the paper.		
Signature		Date	3/2/2016

Name of Co-Author			
Contribution to the Paper			
Signature		Date	

Please cut and paste additional co-author panels here as required.

A Feature-Based Prediction Model of Algorithm Selection for Constrained Continuous Optimisation

Shayan Poursoltan
Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, Australia

Frank Neumann
Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, Australia

Abstract

With this paper, we contribute to the growing research area of feature-based analysis of bio-inspired computing. In this research area, problem instances are classified according to different features of the underlying problem in terms of their difficulty of being solved by a particular algorithm. We investigate the impact of different sets of evolved instances for building prediction models in the area of algorithm selection. Building on the work of Poursoltan and Neumann [11, 10], we consider how evolved instances can be used to predict the best performing algorithm for constrained continuous optimisation from a set of bio-inspired computing methods, namely high performing variants of differential evolution, particle swarm optimization, and evolution strategies. Our experimental results show that instances evolved with a multi-objective approach in combination with random instances of the underlying problem allow to build a model that accurately predicts the best performing algorithm for a wide range of problem instances.

1 Introduction

Throughout the history of heuristic optimisation, various methods have been proposed to solve constrained optimisation problems (COPs), specially non-linear ones. The main idea behind these algorithms is to tackle the constraints. Important approaches in this area are differential evolution (DE), particle swarm optimisation (PSO) and evolutionary strategies (ES). To handle the constraints, there have been many techniques applied to these algorithms such as penalty functions, special operators (separating the constraint and objective function treatment) and decoder based methods. We refer the reader to [7] for a survey of constraint handling techniques in evolutionary computation. Given a range of different algorithms for constrained continuous optimisation, we consider algorithm selection problem (ASP) [12] which consists of selection the best performing algorithm from a suite of algorithms for a given problem instance. In most circumstances, it is difficult to answer the following question: "Can we estimate the likelihood that algorithm A will be successful on a given constrained optimisation problem P?". Recent works in the field show that it is possible to select the algorithm most likely to be best suited for a given problem [8, 14, 2]. Based on these studies, it is possible to find the links between problem characteristics and algorithm performance. The key to these investigations is problem features which can be used to predict the most suited algorithm from a set of algorithms.

It is widely assumed that constraints play a vital role in COP's difficulty. Therefore, in this study we use the meta-learning framework outlined in [12, 16] to build a prediction model for a given COP. Our model predicts the best algorithm type (DE, ES and PSO) for a given COP based on their constraint features. The model inputs include the features of constraints in a given problem. It is shown in [11, 9] that by using an evolving approach, it is possible to generate problem instances covering a wide range of problem/algorithm difficulty. Such instances can be used to extract and analyse the features that make a problem hard or easy to solve for a given algorithm. For a detailed discussion on these constraints (linear, quadratic and their combination) we refer the reader to [11].

To build a reliable prediction model, we need to train it with variety of problem instances that are hard or easy for algorithm(s). Based on the investigations in [10], a multi-objective evolutionary algorithm can be used to generate constrained problem instances that are hard/easy for one algorithm but still easy/hard for the others. The authors show which features of the constraints make the problems hard for certain algorithm but still easy for the others. Hence, we use the same approach to generate problem instances to use in our model training phase. This can improve the accuracy of prediction model since the training instances are used to show the strengths and weaknesses of various algorithm types over constraint features. To illustrate the model's efficiency on constraints, we examine our model with generated testing problems such as hard/easy for one but easy/hard for the others and more general random instances. To show the model prediction ability over constraints (linear, quadratic and their combination), we also experiment given problems with various objective functions.

The remainder of this paper is organised as follows. In section 2, we discuss constrained continuous optimisation problems. Later, we introduce the evolutionary algorithms that are suggested by our prediction model. Moreover, the background materials related to multi-objective evolver, algorithm selection problem and meta-learning prediction model are discussed in detail. Section 3 describes and compares all models trained with different subsets of instances from multi-objective evolver population set. By choosing the best training data preference in Section 3, the experimental analysis on various benchmark problems is described in Section 4. We then conclude with some remarks in Section 5.

2 Preliminaries

2.1 Constrained Continuous Optimisation Problems

A constrained optimisation problem (COP) in a continuous space is formulated as follows:

$$\begin{aligned} \text{Find} \quad & x \in S \subseteq R^D \\ & f(x) = min(f(y); y \in S), \\ \text{subject to} \quad & g_i(x) \leq 0 \quad \forall i \in \{1, \ldots, q\} \\ & h_j(x) = 0 \quad \forall j \in \{q+1, \ldots, p\} \end{aligned} \tag{1}$$

In this formulation, f, g_i and h_j are real-valued functions on the search space S, q is the number of inequalities and $p-q$ is the number of equalities. The search space S is defined as a D dimensional rectangle in R^D. These equality and inequality constraints could be linear or nonlinear. The set of all feasible points $F \subseteq S$ which satisfy all equality and inequality constraints is formulated as:

$$l_i \leq x_i \leq u_i, \qquad 1 \leq i \leq D \tag{2}$$

where l_i and u_i denote lower and upper bounds for the ith variable respectively. Usually, to simplify COP, the equalities are replaced by the following inequalities [18] as follows:

$$|h_j(x)| \leq \varepsilon \quad \text{for} \;\; j = q+1 \;\; \text{to} \;\; p \tag{3}$$

where ε is a small positive value. In all experiments in this paper, the value of ε is considered as 1E-4, the same as it was in [5].

2.2 Algorithms

In this section we discuss the basic ideas about algorithms for constrained optimisation problems such as differential evolution, evolutionary strategies and particle swarm optimisation.

The ε-constrained differential evolution with an archive and gradient-based mutation (εDEag) is the winner of 2010 CEC competition for continuous constrained optimisation problems [5]. This algorithm uses ε-constrained method technique to transform algorithms for unconstrained problems to constrained ones. Also, possible solutions are ordered by ε-level comparison. This means, the lexicographic order is used in which constraint violation ($\phi(x)$) has more priority and proceeds the function value ($f(x)$). A detailed description of this algorithm can be found in [17].

For evolutionary strategy algorithms, $(1+1)$ CMA-ES for constrained optimisation [1] is included in our experiment. This algorithm is a variant of $(1+1)$-ES which adapts the covariance matrix of its offspring distribution in addition to its global step size. The $(1+1)$ CMA-ES for constrained optimisation obtains approximations to the normal vectors directions in the vicinity of the current solution locations by applying low-pass filtering steps that violates the constraints and reducing the variance of the offspring distribution in these directions. Adopting this method makes $(1+1)$ CMA-ES as one of the most efficient algorithms for constrained optimisation problems. We refer the reader to [1] for detailed description and implementation.

The next algorithm that is used in our investigation from particle swarm optimisation algorithms is hybrid multi-swarm particle swarm optimisation (HMPSO). This algorithm divides the current swarms into sub-swarms and search the solution between them in parallel. All particles in each sub-swarms locate their fittest local particle which attracts the particles to fitter positions. Also, having multiple sub-swarms near different optima increase the diversity of the algorithm. A detailed description and implementation of HMPSO can be found in [19].

2.3 Multi-objective Investigations

In order to extract information about the strengths and weaknesses of certain algorithms on constrained optimisation problems, we need problem instances with different kinds of difficulties for the considered algorithms. The reason behind this idea is that using instances that are randomly generated are not efficient to cover the full spectrum of difficulty analysis. To do this, we evolve instances to find the ones that are hard/easy for one algorithm and easy/hard for the others. Analysing the features of these instances helps us extracting knowledge regarding the strengths and weaknesses of the considered algorithms and give reasons of why an algorithm performs well on one problem while the others have difficulties. Insights from this analysis can be used to develop more efficient prediction model for automated algorithm selection.

As mentioned above, we generate problem instances of different difficulties. To do this, a multi-objective DE algorithm (evolver) [13] is used to evolve constraints that make problems hard for one algorithm type and easy for the others in the algorithm suite following the approach in [11, 10]. The feature-based comparison of various algorithm types has been carried out in these papers. The authors show the constraint (linear and/or quadratic and their combination) features that are more contributing to problem difficulty for certain algorithms. We refer the readers to [11, 10] for a detailed description and implementation.

2.4 Algorithm Selection Problem

There are many algorithms that are proposed for constrained continuous optimisation problems. These algorithms are categorised as different types such as differential evolution (DE), evolutionary strategy (ES) and particle swarm optimisation (PSO). So, as a direct consequence of this, it is difficult to understand which algorithms or types of algorithms are more efficient to solve given COPs. To determine the best algorithm to solve a problem is referred to "Algorithm Selection Problem" term in [12] by Rice. In his work, Rice proposed a model with four main characteristics: a set of problem instances F, a set of algorithms A, measures for the cost of performing algorithms on particular problem (Y) and set of characteristics of problem instances (C). The illustration for Rice general algorithm selection framework is shown in Figure 1 which predicts the performance $y(a(f))$ of a given algorithm a on a problem f by extracted features c. If a solution is found, it is possible to extract features from a given problem

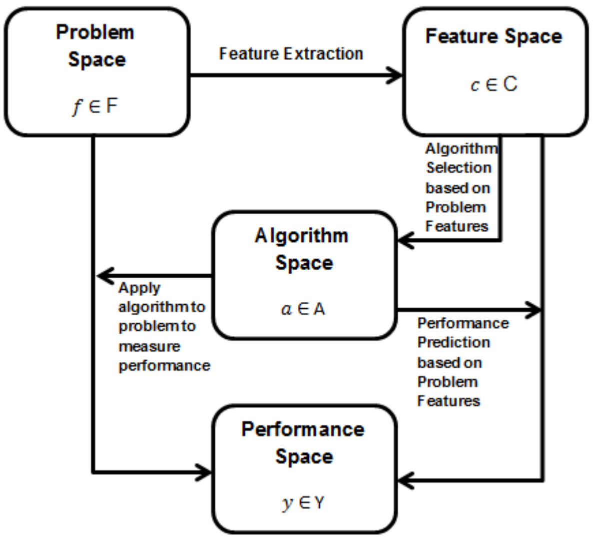
Problem Space
f ∈ F
Feature Extraction
Feature Space
c ∈ C
Algorithm Selection based on Problem Features
Algorithm Space
a ∈ A
Apply algorithm to problem to measure performance
Performance Prediction based on Problem Features
Performance Space
y ∈ Y

Figure 1: The framework of general problem of algorithm selection and performance prediction using problem features based on [12].

and select the most appropriate algorithm or predict the performance of the algorithm based on these features. This framework has been extended by [16, 3, 4] in a variety of computational problem domains using meta-learning framework. So, if the values are features of problems with algorithm performance measure are known beforehand, then it is possible to use a learning strategy to predict the algorithm performance based on the problem features.

2.5 Prediction Model

Our prediction model is based on the [8]. Note that this model is used for unconstrained continuous optimisation problems using landscapes features. Inputs to the model are independent problem feature variables (C) and algorithm parameters and output is the performance measure as an dependant variable which is required function evaluation number (FEN) of the suggested algorithm. This model can be used to predict the algorithm behaviour on a given problem. To achieve this we use popular basic technique for model building. A high-level overview of the regression model is shown in Figure 2.

As discussed above, there have been many attempts to train these prediction models with random generated or benchmark problem instances which could not fully include all problem instances with difficulty variations. To improve this, we cover the full spectrum of difficulty by evolving two sets of instances with extreme problem difficulties. These extreme difficulty instances are the ones which are hard for one algorithm and easy for the others or easy for one and still hard for the rest.

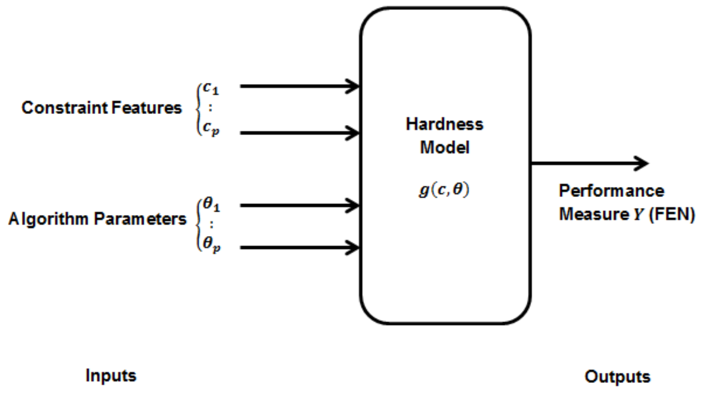

Figure 2: Meta-learning prediction model for a constrained continuous optimisation problem

To build our regression model, we implement a multi-layered feed-forward neural network with 2 hidden layers and 10 neurons in each layer as the regression model. For training the model, we use a Levenbeg-Marqurd back-propagation algorithm [6] package using Matlab R2014b. To train this model, we use evolved instances that are generated from multi-objective evolver in [10]. The prediction model inputs are given COPs constraint features and algorithms parameter values (the parameters for the experimented algorithms are identical to [17, 19, 1].)

3 Prediction Model based on Evolved Instances

As mentioned earlier, our goal is to propose a reliable prediction model using constraint features. This reliability can be improved by choosing proper set of learning data. The accuracy of this prediction model depends on many factors such as the relevance of constraint features, the diversity of instances used to train the model and its training method. Therefore, to improve this, we train our prediction model obtained from multi-objective evolver described in [10]. These instances are hard for one algorithm but still easy for the other (or easy for one and hard for the other algorithms). Analysing these instances shows the effectiveness of constraint features that make a problem hard or easy for certain algorithms. This set up improves the accuracy of prediction which is evaluated on the capability to provide realistic ranking of different algorithm types performances. This can be done by comparing the required function evaluation numbers (FEN) needed by various algorithms to solve a given COP.

The prediction model uses constraint features (first input) to predict the best algorithm type for a given constrained optimisation problem. These constraint features are constraint coefficients relationships such as standard deviation, angle between constraint hyperplanes, feasibility ratio in vicinity of optimum and number of constraints. The details of these features are discussed in [11]. Also, for the second input, since selecting various algorithm parameters has different impact on algorithm ability to solve a given problem, we conduct an experimentally driven meta-learning approach which has been proposed by Smith-Miles in [15]. For our model, we use the parameters for DE, CMA-ES and HMPSO suggested in [17, 19, 1].

Our goal of building a prediction model is to identify the best algorithm type for a given problem. Therefore, the model output predicts the most suited algorithm and required FEN to solve a given constrained problem. The suggested FENs denote numbers of function evaluation which are needed by different algorithms to solve a given COP.

In the following we train our prediction model with variety of instance subsets generated by the multi-objective evolver. We choose different combination of subsets of instances to maximise our prediction model accuracy upon a given constrained problem. These training phase instance subsets are selected from extreme points, Pareto front line, more random (general) solutions and combination of Pareto front and random points from multi-objective evolver solution population in [10]. We then compare the prediction accuracy for these prediction model with various training data preferences.

3.1 Extreme Instances

We first train our prediction model with an extreme instance subset which covers the extreme points of Pareto front in the multi-objective evolver population set. These extreme solutions are selected from evolved instances that are easiest/hardest for one and hardest/easiest for the other algorithms at the same time. The reason behind this selection is to assess the ability of our model to find the most suited algorithm for a given COP which is fairly hard for one or multiple algorithms. In other words, it is more beneficial to choose a best algorithm for a given COP in which it cannot be solved easily by certain other algorithms.

To determine the actual accuracy of our extreme point prediction model (EP-PM), we select 1500 extreme instances that are hard/easy for one and easy/hard for the other algorithms for its training phase. To analyse and test the quality of this prediction model, we use two sets of testing problem instances that we already know their best algorithm and required FEN. The first one is the set of problem instances that are hard/easy for one algorithm and easy/hard for the others. This set can improve the accuracy of EP-PM for given problems that fall into extreme-like evolved problem instances. However, it is very likely that the real world given COP is similar to other evolved instance subset types. Therefore, as a second set, we use random (general) testing problem instances to analyse the EP-PM with a potential real world given problem. We need to mention that we already know about their best algorithm and required FEN.

Our result for EP-PM is summarised in Table 1 for instances that are hard/easy for one and easy/hard for the other algorithms. Also, the results for testing random instances are shown in this table. Moreover, DE hard (1 C) denotes testing Sphere problem instance with 1 linear constraint that is hard for DE algorithm but still easy for the others. The information about actual and predicted algorithm and required FEN to solve a problem instance is indicated. The model not only suggests the best algorithm with its required FEN but also predicts the FENs required to solve a given problem with other algorithms (not the best ones).

Based on the results, EP-PM performs acceptable on extreme-like testing instances. This performance is acceptable on predicted best algorithm type and required FEN. Also, the error rates for using other algorithms (not the best ones) are still acceptable. But, analysing the testing random instances, it is obvious the model is not capable of predicting algorithm and FEN for these more general form instances. This means, the proposed model (EP-PM) is not accurate enough for randomly generated real world instances and the difference between predicted and actual FEN is considerable. Also, the error rate of other algorithm choices (not the best algorithm) is still high.

To summarize, although the EP-PM model performs fairly accurate on instances that are grouped into extreme points evolved instances, still needs improvement to handle other subsets (such as random generated instances). The likelihood of given COP which is more similar to random evolved instances are higher. Thus, this motivates us to examine other subsets from evolving algorithm population instances for our training phase. This could be moving along the Pareto front line and choosing more instances from this category.

3.2 Pareto Front Instances

It is shown that in order to improve the accuracy of our prediction model we need to include or select different varieties of evolved instance subsets for its training step. The idea behind this choice is to obtain a model that can predict more general forms of given constrained problems. Of course it is vital to predict best algorithm for a given problem which is considerably hard/easy for one and still easy/hard for the other algorithms, but we also need to include more forms of generality to our prediction model. So, we need to move along Pareto front line in multi-objective evolver population set for our learning phase. This could increase the ability of our model to predict algorithms for more general given COP which is not similar to extreme point instances.

Given a total number of 3000 instances from evolver Pareto front line, we train our Pareto front prediction model (PF-PM). This preference could increase PF-PM ability to predict more general forms of given COPs. To compare the quality of our prediction model and other models with different learning phases, we use same extreme and random generated testing instances used in previous section. Results shown in Table 2 indicate an improving accuracy for random testing instances used for previous model EP-PM (using extreme instances). Looking at the Table 2, we see that moving towards Pareto front line in evolver population set for choosing learning instances increases the accuracy of predicted FEN for predicted algorithm. Also, predicting FEN using other algorithms (not the best one) represents that the PF-PM is more accurate than EP-PM for testing random instances. This improvement is acceptable in algorithm type prediction but we still need to improve the predicted required FEN.

As discussed in this section, to improve the accuracy of prediction model, we chose instances of evolver from its Pareto front line solution population. In other words, Pareto front prediction model (PF-PM) has some strengths and weaknesses. Although its error rate for predicting correct algorithms is improved, there is still considerable difference between the actual required FENs and predicted ones. As testing instances are selected mostly from more general instances (not close to extreme points), we need to experiment other training instances subsets. In order to address more general form, based on results so far, our next move is to choose more random instances from evolver population set for our training set. This could result in increasing the accuracy of our model for more general forms of given COPs.

3.3 Random Instances

The initial prediction models discussed earlier (EP-PM and PF-PM) has some limitations. In other words, they are not accurate enough to predict more general forms of given COPs, specially when they are similar to evolver population instances except extreme and Pareto front points. The results for PF-PM show an increase in accuracy of prediction for testing COPs which are not similar to extreme points, but the predicted required FEN still needs an improvement. Our goal is to design a prediction model with an ability to predict all possible given COPs. These COPs are within the range of extreme to random like instances. Based on previous results, it is shown that moving from extreme to Pareto front line instances increases the model accuracy (see Section 3.2). Hence, to decrease the error rate for required FEN for more general testing COPs we choose only random instances for testing phase. These random instances are selected from evolver population set.

We use 3000 random instances from multi-objective evovler population set for our random only prediction model (RO-PM). To assess the accuracy of our random only prediction model we use the same testing instances applied to EP-PM and PF-PM. Table **??** indicates the actual and predicted FEN and algorithm of hard/easy and random testing instances for RO-PM. As it is observed, the random only prediction model (RO-PM) fails to predict suitable algorithm for a given COP. This failure include both predicted algorithm and required FEN. Results show that moving through random instances in evolver population and choose only random instances increase the number of incorrect predictions. Also, comparing to previous models, RO-PM accuracy is decreased for testing instances similar to extreme points (easy/hard instances).

It is shown that the accuracy of resulting model with Pareto front instances (PF-PM) is improved by selecting different subsets (Pareto front line) than extreme points. This improvement is analysed by experimenting more general form of testing COPs. Therefore, this motivated us to experiment only random solutions in order to build more accurate prediction model for given COPs. By selecting only random instances to train the new model, it is observed the predicted algorithm and required FEN is not accurate as Pareto front (PF-PM) and extreme points (EP-PM) models. It can be translated as excluding instances from Pareto front line for training step decreases the accuracy of prediction model. Also, the RO-PM model failed to predict testing instances which are similar to evolver extreme points (easy/hard instances). So, other possibility is to use a combination of both Pareto front and random instances from evolver population for model training.

3.4 Pareto Front with Random Instances

In previous sections we experimented three types of prediction models (EP-PM, PF-PM and RO-PM). These models have different prediction accuracy upon choosing different varies of evolver subsets for their training phases. Our goal of building a prediction model is to minimise its error rate in both algorithm and required FEN. So far, we understand that moving from extreme points (EP-PM) to Pareto front (PF-PM) for training step increases the accuracy of prediction model. However, moving further and choosing only random points from evolver (RO-PM) is not the solution for covering all possible given COPs (extreme and random like instances). In other words, there should be a trade-off relation between moving towards random points from extreme and random instances in multi-objective evolver population. Therefore, our preference for training phase is a combination set of Pareto front and random instances from multi-objective evolver. Not only it covers instances that are hard/easy for one and easy/hard for the other algorithms, but also it can predict general given COPs more accurately.

To train our Pareto front with random instances prediction model (PFR-PM), we use 3000 points from evolver population set (1500 each). To compare and assess the prediction model accuracy we experiment our PFR-PM with the same testing COPs for the former models. The results for Pareto front and random prediction model (PFR-PM) are shown in Table 4. It is observed that including both Pareto front line and random only instances can be effective in accuracy improvement. Analysing the results, it is obvious the error rate for both predicted algorithm and required FEN are decreased. Also, the model is able to predict required FEN using other algorithms (not the best one) more accurate. This can be concluded by having lower error rates for FENs of other algorithms (not the best one) for PFR-PM. The reason behind this is that to cover all possible given COPs, we use both Pareto front and random points from evolver population for out training phase. In other words, our model is trained with constraint characteristics and features of both types of COPs (Pareto front and random instances).

It is shown that choosing the proper subsets for training phase is effective in prediction model accuracy. The results for four prediction models suggest that moving from extreme points to random instances can improve the quality of prediction model. It is found that there is a trade-off relationship in choosing instances that are close to random or extreme points (from evolver) for training phase. Results analysis shows by selecting a combination subsets from both random and Pareto front instances (PFR-PM) for training step, we can improve our model prediction quality. This improvement is in both selected algorithm and also its required FEN. By selecting the best model (PFR-PM), in the following, we examine it in a more detailed approach.

4 Experiments on Benchmarks

Our goal is to design highly accurate prediction model for a given COP based on its constraint features. As mentioned earlier, in order to improve the model accuracy, it needs to be trained with COP instances that are generated using multi-objective evolver. So far, we analysed the results for all four types of prediction models trained with extreme points (EP-PM), Pareto front (PF-PM), random only (RO-PM) and combination of Pareto front with random (PFR-PM) instances. We have experimented our prediction model with various subsets of training data to analyse the best preference. As results indicate, the most accurate prediction model is the one which is trained with combination of Pareto front and random subset of evolver population (PFR-PM). This model is capable of predicting algorithms for almost all possible given testing COPs such as the ones similar to extreme (hard/easy) or ordinary instances (random) in a multi-objective evolver population. Also, the prediction ability for required function evaluation number (FEN) has been significantly increased. Therefore, in order to assess our optimised prediction model (see Section 3.4), we decide to experiment it with our newly designed benchmark. In order to analyse the capability of the prediction model (PFR-PM) on constraints, we use fixed objective function with various numbers of linear, quadratic (and their combination) constraints. Then, we test other well-known objective function to see the relationship of constraints and our prediction model.

For this experiment, we train our PFR-PM with 3000 instances from both Pareto front and random instances of evolving algorithm population set (1500 each). We also use optimised algorithm parameter settings for each algorithm suggested in [17, 19, 1]. In order to show the accuracy of model on prediction over constraints, we examine the model on various well-known objective functions such as Sphere (bowl-shaped), Ackley (many local optima) and Rosenbrock (valley-shaped). Also, to evaluate the effectiveness of our model on constrained problems, we use various numbers and types of constraints. Tables 5, 6 and 7 show the prediction results for Sphere, Ackley and Rosenbrock objective functions respectively. The results show the number of correct algorithm types prediction (success rate) from 30 different tests. Also, one step further, the average deviations of required FEN (the correct and predicted one) for the predicted algorithms are calculated.

Table 5 compares the prediction results for our proposed model (PFR-PM) and random only model (RO-PM) for Sphere COPs. The results indicate the effectiveness of choosing the proper subset training instances. It is observed that the prediction algorithm success rate for our proposed model (PFR-PM) is significantly better than RO-PM for all Sphere COPs using various combinations of constraints. The success rate (out of 30 tests) for newly testing given COP is significantly higher for PFR-PM comparing to RO-PM. Also, the low value average deviation of predicted FEN and actual one for PFR-PM represents its higher accuracy in predicting the algorithm performance in terms of function evaluation number.

By observing the Tables 6 and 7, we realise that our prediction model (PFR-PM) is reliable in predicting with only constraints. In other words, experimenting other types of objective functions (Bowl-shaped, many local-optima and valley shaped) with accurate results shows the ability of the model to predict based on constraints. Based on the Table 6, the lower value of FEN average deviation indicates the higher accuracy of PFR-PM for Ackley COPs. Also, the results for Rosenbrock COPs with 1 to 5 linear, quadratic constraints (and their combination) shows the accuracy of PFR-PM comparing to RO-PM. The average deviation of FEN for Rosenbrock problems denotes the significantly close predicted FEN with PFR-PM (see Table 7).

As mentioned before, the output of our proposed prediction model (PFR-PM) includes predicted algorithm with its required FEN. It is observed that the prediction model is capable of suggesting the best algorithm and required FEN based on constraint features of given COP. Due to the stochastic nature of evolutionary optimisation, the above benchmark tests are repeated 30 times and the two-tail t-test significance is performed for average deviations of FEN. The significant level α is considered as 0.05. The p-values for significance of a difference between FEN average deviation of Pareto front with random (PFR-PR) and random only (RO-PR) models for each Sphere, Ackley and Rosenbrock are shown in Tables 5, 6 and 7 respectively. The results show that the difference in FEN Average deviation are significant and less than 0.05.

As discussed earlier, the idea of designing a prediction model based on instance features is rather a novel approach in algorithm selection problem. Training a model with COP instances from multi-objective evolver improves the prediction accuracy. The performance prediction (FEN) and suggested algorithm can be used to produce the final output of our prediction model. As we know selecting a suitable algorithm for a given problem requires substantial amount of time. In contrast, in our approach, we only need to extract features of a problem once and the model produces the final output. It is observed that selecting different sets of training instances improves the prediction model success rate. We designed and examined various prediction model using different subsets of problem instances from evolver population set. In order to show the ability of the prediction model only based on constraints features we use various objective functions. Results for these COPs with different combinations of objective functions and constraints indicate that the model is highly accurate in algorithm and required FEN prediction.

5 Conclusion

In this study, we examined the impact of different types of problem instances that can be used in prediction models for constrained continuous optimisation.

Our resulting prediction model captures the links between constraint features, algorithm type performance and the required function evaluation number. The model inputs are considered as constraint features and selected parameter settings. The outputs includes the required function evaluation number and most suited algorithm type to solve the given COP.

The model was trained (using NN learning strategy) with evolved COP instances. To improve the accuracy of the model we used evolved instances that are hard/easy for one and easy/hard for the other algorithms. These training instances are generated with multi-objective evolver. We first, chose various subsets of instances from multi-objective evolver population set. It is observed that the a model using combination of Pareto front line and random points from population set has the highest accuracy in predicting best algorithm types for a given COP. We then, tested our prediction model with different objective functions and constraints types. The results indicate our prediction model is reliable to suggest and predict most suited algorithm and required FEN using problem's constraint features. Our approach shows the relationship between constraint features and various algorithm performances. The results clearly demonstrate the ability of prediction model to predict the algorithm and required FEN using only constraint features.

References

[1] D. V. Arnold and N. Hansen. A (1+ 1)-cma-es for constrained optimisation. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 297–304. ACM, 2012.

[2] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuß. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 313–320. ACM, 2012.

[3] F. Hutter, Y. Hamadi, H. H. Hoos, and K. Leyton-Brown. Performance prediction and automated tuning of randomized and parametric algorithms. In *Principles and Practice of Constraint Programming-CP 2006*, pages 213–228. Springer, 2006.

[4] K. Leyton-Brown, E. Nudelman, and Y. Shoham. Empirical hardness models: Methodology and a case study on combinatorial auctions. *Journal of the ACM (JACM)*, 56(4):22, 2009.

[5] R. Mallipeddi and P. N. Suganthan. Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. *Nanyang Technological University, Singapore*, 2010.

[6] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441, 1963.

[7] E. Mezura-Montes and C. A. Coello Coello. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

[8] M. A. Muñoz, M. Kirley, and S. K. Halgamuge. A meta-learning prediction model of algorithm performance for continuous optimization problems. In *Parallel Problem Solving from Nature-PPSN XII*, pages 226–235. Springer, 2012.

[9] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of linear constraints for ε-constrained differential evolution. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 3088–3095. IEEE, 2014.

[10] S. Poursoltan and F. Neumann. A feature-based analysis on the impact of set of constraints for ε-constrained differential evolution. In S. Arik, T. Huang, W. K. Lai, and Q. Liu, editors, *Neural Information Processing*, volume 9491 of *Lecture Notes in Computer Science*, pages 344–355. Springer International Publishing, 2015.

[11] S. Poursoltan and F. Neumann. A feature-based comparison of evolutionary computing techniques for constrained continuous optimisation. In S. Arik, T. Huang, W. K. Lai, and Q. Liu, editors, *Neural Information Processing*, volume 9491 of *Lecture Notes in Computer Science*, pages 332–343. Springer International Publishing, 2015.

[12] J. R. Rice. The algorithm selection problem. 1975.

[13] T. Robič and B. Filipič. Demo: Differential evolution for multiobjective optimization. In *Evolutionary Multi-Criterion Optimization*, pages 520–533. Springer, 2005.

[14] K. Smith-Miles. Towards insightful algorithm selection for optimisation using meta-learning concepts. In *WCCI 2008: IEEE World Congress on Computational Intelligence*, pages 4118–4124. IEEE, 2008.

[15] K. A. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41(1):6, 2008.

[16] K. A. Smith-Miles, R. J. James, J. W. Giffin, and Y. Tu. A knowledge discovery approach to understanding relationships between scheduling problem structure and heuristic performance. In *Learning and intelligent optimization*, pages 89–103. Springer, 2009.

[17] T. Takahama and S. Sakai. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–9. IEEE, 2010.

[18] T. Takahama and S. Sakai. Efficient constrained optimization by the ε constrained adaptive differential evolution. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[19] Y. Wang and Z. Cai. A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems. *Frontiers of Computer Science in China*, 3(1):38–52, 2009.

Table 1: Predicted and actual most suited algorithm type/required FEN for Sphere function with linear constraint(s). The prediction model is trained with extreme points (EP-PM) from multi objective evolver population. DE hard/easy (1 C) is a problem instance that is hard/easy for DE algorithm but easy/hard for the others.

Instances name	Predicted alg.	Actual alg.	Error	Predicted FEN for DE	Actual FEN for DE	Error for DE	Predicted FEN for ES	Actual FEN for ES	Error for ES	Predicted FEN for PSO	Actual FEN for PSO	Error for PSO
DE hard (1 c)	ES	ES	NO	83.2K	86.3K	-3.1K	**43.7K**	**41.5K**	+2.2K	46.8K	43.2K	+3.6K
ES hard (1 c)	PSO	DE	YES	43.4K	**45.7K**	-2.3K	80.9K	84.2K	-3.3K	**41.6K**	48.3K	-6.7K
PSO hard (1 c)	DE	DE	NO	**38.9K**	**37.2K**	+1.7K	43.2K	41.8K	+1.4K	76.2K	80.1K	-3.9K
DE hard (2 c)	ES	ES	NO	83.5K	87.4K	-3.9K	**42.5K**	**45.2K**	-2.7K	44.2K	43.6K	+1.4K
ES hard (2 c)	DE	DE	NO	**47.2K**	**46.4K**	+0.8K	84.3K	88.3K	-4.0K	49.4K	46.8K	+2.6K
PSO hard (2 c)	DE	DE	NO	**41.5K**	**43.6K**	-2.1K	46.3K	45.1K	+1.2K	85.5K	83.2K	+2.2K
DE hard (3 c)	ES	ES	NO	87.3K	92.4K	-5.1K	**43.8K**	**45.2K**	-1.4K	45.6K	47.2K	-1.6K
ES hard (3 c)	PSO	PSO	NO	45.6K	51.2K	-5.6K	95.5K	91.2K	+4.3K	**43.5K**	**48.2K**	-4.7K
PSO hard (3 c)	DE	DE	NO	**47.2K**	**49.6K**	-2.4K	51.2K	53.5K	-2.3K	92.1K	89.5K	+2.6K
DE hard (4 c)	PSO	ES	YES	95.2K	93.7K	-1.5K	52.4K	**47.2K**	+5.2K	**49.2K**	50.2K	-1.0K
ES hard (4 c)	PSO	PSO	NO	51.3K	49.2K	+2.1K	85.3K	89.1K	-3.8K	**46.3K**	**48.9K**	-2.6K
PSO hard (4 c)	DE	ES	YES	**49.2K**	52.7K	-3.5K	53.4K	**50.8K**	+2.6K	89.3K	87.5K	+1.8K
DE hard (5 c)	ES	ES	NO	94.7K	96.3K	-1.6K	**49.3K**	**53.3K**	-4.0K	57.1K	55.3K	+1.8K
ES hard (5 c)	DE	DE	NO	**51.4K**	**53.8K**	-2.4K	94.2K	92.6K	+1.6K	55.2K	54.7K	-2.5K
PSO hard (5 c)	DE	DE	NO	**49.1K**	**51.3K**	-2.2K	57.9K	55.2K	+2.7K	89.5K	93.2K	-3.7K
DE easy (1 c)	DE	DE	NO	**45.3K**	**48.9K**	-3.6K	81.5K	85.2K	-3.7K	75.4K	79.1K	-3.7K
ES easy (1 c)	ES	ES	NO	87.3K	81.4K	+5.9K	**48.3K**	**54.7K**	-6.4K	79.4K	81.7K	-2.3K
PSO easy (1 c)	PSO	PSO	NO	92.5K	87.1K	+5.4K	81.4K	84.2K	-2.8K	**48.2K**	**41.9K**	+6.3K
DE easy (2 c)	DE	DE	NO	**49.4K**	**44.5K**	+4.9K	89.3K	83.1K	+6.2K	78.9K	81.6K	-2.7K
ES easy (2 c)	ES	ES	NO	78.4K	83.6K	-5.2K	**51.4K**	**55.3K**	-3.9K	79.3K	78.9K	+0.4K
PSO easy (2 c)	PSO	PSO	NO	84.2K	81.4K	+2.8K	78.3K	83.9K	-5.6K	**51.6K**	**49.4K**	2.2K
DE easy (3 c)	DE	DE	NO	**41.8K**	**48.4K**	-6.6K	85.8K	89.4K	-3.6K	86.3K	84.5K	+1.8K
ES easy (3 c)	ES	ES	NO	91.4K	87.8K	+3.6K	**48.2K**	**46.2K**	+2.0K	51.3K	48.0K	+3.3K
PSO easy (3 c)	PSO	PSO	NO	88.4K	89.5K	-1.1K	93.2K	87.3K	+5.9K	**46.8K**	**49.1K**	-2.3K
DE easy (4 c)	DE	DE	NO	**51.6K**	**53.2K**	-1.6K	83.5K	85.1K	-1.6K	91.4K	93.8K	-2.4K
ES easy (4 c)	ES	ES	NO	85.3K	89.4K	-4.1K	**51.2K**	**48.9K**	+2.3K	89.4K	87.2K	+2.2K
PSO easy (4 c)	PSO	DE	YES	57.2K	**55.5K**	+1.7K	84.1K	92.4K	-8.3K	**54.2K**	68.9K	-14.7K
DE easy (5 c)	DE	DE	NO	**59.2K**	**57.8K**	+1.4K	93.2K	95.3K	-2.1K	81.3K	81.5K	-0.2K
ES easy (5 c)	ES	ES	NO	88.3K	91.9K	-3.6K	**53.2K**	**55.7K**	-2.5K	86.3K	81.1K	5.2K
PSO easy (5 c)	PSO	PSO	NO	89.3K	91.3K	-2.0K	84.3K	82.9K	+1.4K	**55.8K**	**57.6K**	-1.8K
Rnd. 1 (1 c)	DE	PSO	YES	**53.2K**	65.7K	-12.5K	65.3K	56.9K	+8.4K	62.4K	**53.2K**	+9.2K
Rnd. 2 (1 c)	ES	DE	YES	77.2K	**61.9K**	15.3K	**43.5K**	64.2K	-20.7	51.9K	65.2K	-13.3K
Rnd. 3 (2 c)	ES	DE	YES	71.4K	**59.8K**	+11.6K	**48.1K**	65.3K	-17.2K	60.2K	69.6K	-9.4K
Rnd. 4 (2 c)	PSO	ES	YES	61.2K	72.2K	-11.0K	68.8K	**59.4K**	+9.4K	**55.3K**	71.4K	-16.1K
Rnd. 5 (3 c)	DE	DE	NO	**45.2K**	**65.4K**	-20.2K	77.2K	66.9K	+10.3K	64.5K	74.6K	-10.1K
Rnd. 6 (3 c)	ES	PSO	YES	71.9K	82.4K	-10.5K	**45.6K**	78.3K	-32.7K	50.3K	**61.4K**	-11.1K
Rnd. 7 (4 c)	DE	ES	YES	**71.6K**	83.2K	-11.6K	80.3K	**65.8K**	+14.5K	89.2K	78.9K	+10.3K
Rnd. 8 (4 c)	PSO	DE	YES	84.7K	**71.1K**	+13.6K	68.8K	79.2K	-10.4K	**62.7K**	73.2K	-10.5K
Rnd. 9 (5 c)	ES	DE	YES	91.8K	**82.7K**	+9.1K	**83.9K**	93.8K	-9.9K	89.0K	97.3K	-8.3K
Rnd. 10 (5 c)	DE	PSO	YES	**78.4K**	93.2K	-14.8K	79.2K	89.5K	-10.3K	89.3K	**68.4K**	+20.9K
Rnd. 11 (1 c)	ES	DE	YES	56.3K	**44.6K**	+11.7K	**49.3K**	59.2K	-9.9K	65.2K	64.2K	-8.0K
Rnd. 12 (1 c)	ES	DE	YES	54.2K	**45.2K**	+9.0K	**48.2K**	58.9K	-10.7K	61.2K	69.4K	-8.2K
Rnd. 13 (2 c)	DE	PSO	YES	**48.2K**	61.2K	-13.0K	78.5K	64.3K	+14.2K	50.9K	**59.1K**	-8.2K
Rnd. 14 (2 c)	ES	PSO	YES	70.2K	79.0K	-8.8K	**61.2K**	75.3K	-14.1K	71.2K	**63.4K**	+7.8K
Rnd. 15 (3 c)	DE	PSO	YES	**63.4K**	71.9K	-8.5K	71.3K	79.3K	-8.0K	73.1K	**65.3K**	+7.8K
Rnd. 16 (3 c)	PSO	DE	YES	77.5K	**69.2K**	+8.3K	89.5K	79.3K	+10.2K	**74.3K**	70.1K	+6.2K
Rnd. 17 (4 c)	ES	ES	NO	69.3K	73.9K	-4.6K	**68.5K**	**75.9K**	-7.4K	69.K	77.4K	-7.7K
Rnd. 18 (4 c)	DE	PSO	YES	**71.4K**	85.3K	-13.9K	81.2K	78.3K	+2.9K	75.3K	**68.3K**	+7.0K
Rnd. 19 (5 c)	DE	ES	YES	**81.2K**	93.9K	-12.7K	83.2K	**91.2K**	-8.0K	87.2K	95.9K	-8.7K
Rnd. 20 (5 c)	PSO	PSO	NO	81.2K	90.9K	-9.7K	81.3K	93.2K	-11.9K	**78.2K**	**87.3K**	-9.1K

Table 2: Predicted and actual most suited algorithm type/required FEN for Sphere function with linear constraint(s). The prediction model is trained with pareto front line (PF-PM) from multi objective evolver population. DE hard/easy (1 C) is a problem instance that is hard/easy for DE algorithm but easy/hard for the others.

Instances name	Predicted alg.	Actual alg.	Error	Predicted FEN for DE	Actual FEN for DE	Error for DE	Predicted FEN for ES	Actual FEN for ES	Error for ES	Predicted FEN for PSO	Actual FEN for PSO	Error for PSO
DE hard (1 c)	ES	ES	NO	82.7K	86.3K	-3.6	**43.5K**	**41.5K**	2.0K	47.3K	43.2K	4.1K
ES hard (1 c)	PSO	DE	YES	42.3K	**45.7K**	-3.4K	86.4K	84.2K	2.2K	**40.3K**	48.3K	-8.0K
PSO hard (1 c)	DE	DE	NO	**35.1K**	**37.2K**	-2.1K	43.8K	41.8K	2.0K	77.4K	80.1K	-2.7K
DE hard (2 c)	ES	ES	NO	89.4K	87.4K	2.0K	**43.1K**	**45.2K**	-2.1K	45.7K	43.6K	2.1K
ES hard (2 c)	DE	DE	NO	**48.5K**	**46.4K**	2.1K	86.9K	88.3K	-1.4K	48.2K	46.8K	1.4K
PSO hard (2 c)	DE	DE	NO	**42.3K**	**43.6K**	-1.3K	47.2K	45.1K	2.1K	84.6K	83.2K	1.4K
DE hard (3 c)	ES	ES	NO	87.6K	92.4K	-4.8K	**42.9K**	**45.2K**	-2.3K	46.2K	47.2K	-1.0
ES hard (3 c)	DE	PSO	YES	**49.3K**	51.2K	-1.9K	94.5K	91.2K	3.3K	50.7K	**48.2K**	2.5K
PSO hard (3 c)	DE	DE	NO	**47.5K**	**49.6K**	-2.1K	55.7K	53.5K	2.2K	93.1K	89.5K	3.6K
DE hard (4 c)	ES	ES	NO	94.9K	93.7K	1.2K	**49.2K**	**47.2K**	2.0K	53.6K	50.2K	3.4K
ES hard (4 c)	PSO	PSO	NO	52.4K	49.2K	3.2K	86.9K	89.1K	-2.2K	**45.1K**	**48.9K**	-3.8K
PSO hard (4 c)	ES	ES	NO	55.1K	52.7K	2.4K	**51.9K**	**50.8K**	1.1K	85.2K	87.5K	-2.3K
DE hard (5 c)	ES	ES	NO	94.1K	96.3K	-2.2	**55.9K**	**53.3K**	2.6K	58.3K	55.3K	3.0K
ES hard (5 c)	DE	DE	NO	**51.9K**	**53.8K**	-1.9K	95.1K	92.6K	2.5K	52.9K	54.7K	-1.8K
PSO hard (5 c)	ES	DE	YES	54.7K	**51.3K**	3.4K	**53.1K**	55.2K	-2.1K	89.4K	93.2K	-3.8K
DE easy (1 c)	PSO	DE	YES	72.3K	**48.9K**	23.4K	80.4K	85.2K	-4.8K	**71.2K**	79.1K	-7.9K
ES easy (1 c)	ES	ES	NO	78.4K	81.4K	-3.0K	**47.2K**	**54.7K**	-7.5K	78.9K	81.7K	-2.8K
PSO easy (1 c)	PSO	PSO	NO	91.4K	87.1K	4.3K	80.7K	84.2K	-3.5K	45.8K	**41.9K**	3.9K
DE easy (2 c)	DE	DE	NO	**48.1K**	**44.5K**	3.6K	85.9K	83.1K	2.8K	85.1K	81.6K	3.5K
ES easy (2 c)	ES	ES	NO	79.4K	83.6K	-4.2K	**52.9K**	**55.3K**	-2.4K	80.7K	78.9K	1.8K
PSO easy (2 c)	PSO	PSO	NO	83.5K	81.4K	2.1K	79.3K	83.9K	-4.6K	**47.1K**	**49.4K**	-2.3K
DE easy (3 c)	DE	DE	NO	**42.4K**	**48.4K**	-6.0K	86.2K	89.4K	-3.2K	87.5K	84.5K	3.0K
ES easy (3 c)	ES	ES	NO	90.4K	87.8K	2.6K	**47.9K**	**46.2K**	1.7K	50.5K	48.0K	2.5K
PSO easy (3 c)	PSO	PSO	NO	86.4K	89.5K	-3.1K	94.1K	87.3K	6.8K	**47.9K**	**49.1K**	-1.2K
DE easy (4 c)	DE	DE	NO	**50.8K**	**53.2K**	-2.4K	84.2K	85.1K	-0.9K	95.3K	93.8K	1.5K
ES easy (4 c)	ES	ES	NO	86.3K	89.4K	-3.1K	**46.2K**	**48.9K**	-2.7K	90.3K	87.2K	3.1K
PSO easy (4 c)	PSO	DE	YES	60.4K	**55.5K**	4.9K	88.2K	92.4K	-4.2K	**58.8K**	68.9K	-10.1K
DE easy (5 c)	DE	DE	NO	**56.7K**	**57.8K**	-1.1K	92.5K	95.3K	-2.7K	83.2K	81.5K	1.7K
ES easy (5 c)	ES	ES	NO	87.3K	91.9K	-4.6K	**52.8K**	**55.7K**	-2.9K	85.4K	81.1K	4.3K
PSO easy (5 c)	PSO	PSO	NO	88.2K	91.3K	-3.1K	85.2K	82.9K	2.3K	**56.2K**	**57.6K**	-1.4K
Rnd. 1 (1 c)	PSO	PSO	NO	60.8K	65.7K	-4.9K	60.2K	56.9K	+3.3K	**48.8K**	**53.2K**	-4.4K
Rnd. 2 (1 c)	DE	DE	NO	**67.3K**	**61.9K**	+5.4K	68.1K	64.2K	+3.9K	71.3K	65.2K	+6.1K
Rnd. 3 (2 c)	DE	DE	NO	**51.5K**	**59.8K**	-8.3K	59.9K	65.3K	-5.2K	63.8K	69.6K	-5.8K
Rnd. 4 (2 c)	ES	ES	NO	65.9K	72.2K	-6.3K	**64.1K**	**59.4K**	+4.7K	64.7K	71.4K	-6.7K
Rnd. 5 (3 c)	ES	DE	YES	59.2K	**65.4K**	-6.2K	**57.1K**	66.9K	-9.8K	67.3K	74.6K	-7.3K
Rnd. 6 (3 c)	PSO	PSO	NO	74.9K	82.4K	-7.5K	69.2K	78.3K	-9.1K	**55.1K**	**61.4K**	-6.3K
Rnd. 7 (4 c)	ES	ES	NO	87.6K	83.2K	+4.4K	**71.3K**	**65.8K**	+5.5K	72.1K	78.9K	-6.8K
Rnd. 8 (4 c)	ES	DE	YES	78.3K	**71.1K**	+7.2K	**70.5K**	79.2K	-8.7K	77.9K	73.2K	+4.7K
Rnd. 9 (5 c)	DE	DE	NO	**87.9K**	**82.7K**	+5.2K	88.2K	93.8K	-5.6K	91.9K	97.3K	-5.4K
Rnd. 10 (5 c)	PSO	PSO	NO	79.1K	93.2K	-14.1K	85.6K	89.5K	-3.9K	**75.6K**	**68.4K**	+7.2K
Rnd. 11 (1 c)	DE	DE	NO	**51.2K**	**44.6K**	+6.6K	52.1K	59.2K	-7.1K	57.1K	64.2K	-7.1K
Rnd. 12 (1 c)	DE	DE	NO	**49.2K**	**45.2K**	+4.0K	52.1K	58.9K	-6.8K	62.4K	69.4K	-7.0K
Rnd. 13 (2 c)	PSO	PSO	NO	53.2K	61.2K	-8.0K	60.5K	64.3K	-3.8K	**52.7K**	**59.1K**	-6.4K
Rnd. 14 (2 c)	PSO	PSO	NO	72.9K	79.0K	-6.1K	65.9K	75.3K	-9.4K	**59.6K**	**63.4K**	-3.8K
Rnd. 15 (3 c)	ES	PSO	YES	65.1K	71.9K	-6.8K	**59.9K**	79.3K	-19.4K	60.9K	**65.3K**	-4.4K
Rnd. 16 (3 c)	DE	DE	NO	**63.1K**	**69.2K**	-7.9K	72.5K	79.3K	-6.8K	74.2K	70.1K	6.1K
Rnd. 17 (4 c)	ES	ES	NO	69.9K	73.9K	-4.0K	**68.4K**	**75.9K**	-7.5K	70.3K	77.4K	-7.1K
Rnd. 18 (4 c)	PSO	PSO	NO	74.8K	85.3K	-10.5K	84.0K	78.3K	+5.7K	**72.6K**	**68.3K**	+4.3K
Rnd. 19 (5 c)	DE	ES	YES	**85.1K**	93.9K	-8.1K	**85.8K**	91.2K	-6.1K	98.9K	95.9K	+3.0K
Rnd. 20 (5 c)	PSO	PSO	NO	85.1K	90.9K	-5.8K	87.9K	93.2K	-5.3K	**82.5K**	**87.3K**	-4.8K

Table 3: Predicted and actual most suited algorithm type/required FEN for Sphere function with linear constraint(s). The prediction model is trained with random points (RO-PM) from multi objective evolver population. DE hard/easy (1 C) is a problem instance that is hard/easy for DE algorithm but easy/hard for the others.

Instances name	Predicted alg.	Actual alg.	Error	Predicted FEN for DE	Actual FEN for DE	Error for DE	Predicted FEN for ES	Actual FEN for ES	Error for ES	Predicted FEN for PSO	Actual FEN for PSO	Error for PSO
DE hard (1 c)	PSO	ES	YES	93.6K	86.3K	+7.3K	53.5K	**41.5K**	+12K	**38.1K**	43.2K	-5.1K
ES hard (1 c)	DE	DE	NO	**51.8K**	**45.7K**	+6.1K	78.1K	84.2K	-6.1K	53.7K	48.3K	+5.4K
PSO hard (1 c)	PSO	DE	YES	58.8K	**37.2K**	+21.6K	73.5K	41.8K	+31.7K	**57.3K**	80.1K	-22.8K
DE hard (2 c)	ES	ES	NO	66.8K	87.4K	-20.6K	**57.9K**	**45.2K**	+12.7K	72.8K	43.6K	+29.2K
ES hard (2 c)	PSO	DE	YES	67.2K	**46.4K**	+20.8K	73.2K	88.3K	-15.2K	**53.7K**	46.8K	+6.9K
PSO hard (2 c)	DE	DE	NO	**55.7K**	**43.6K**	+12.1K	73.6K	45.1K	+28.5K	77.8K	83.2K	-5.4K
DE hard (3 c)	PSO	ES	YES	73.4K	92.4K	-19.0K	59.2K	**45.2K**	+14.0K	**57.3K**	47.2K	+10.1K
ES hard (3 c)	DE	PSO	YES	**62.7K**	51.2K	+11.5K	73.6K	91.2K	-17.6K	64.3K	**48.2K**	+16.1K
PSO hard (3 c)	ES	DE	YES	59.2K	**49.6K**	+9.6K	**43.2K**	53.5K	-10.3K	103.6K	89.5K	+14.1K
DE hard (4 c)	PSO	ES	YES	88.2K	93.7K	-5.5K	51.2K	**47.2K**	+4.0K	**49.1K**	50.2K	-1.1K
ES hard (4 c)	ES	PSO	YES	**39.8K**	49.2K	-9.4K	96.3K	89.1K	+7.2K	41.2K	**48.9K**	-7.7K
PSO hard (4 c)	DE	ES	YES	**63.9K**	52.7K	+11.2K	64.6K	**50.8K**	+13.8K	83.8K	87.5K	-3.7K
DE hard (5 c)	PSO	ES	YES	102.6K	96.3K	+6.3K	62.3K	**53.3K**	+9.0K	**49.2K**	55.3K	-6.1K
ES hard (5 c)	PSO	DE	YES	47.9K	**53.8K**	-5.9K	84.8K	92.6K	-7.8K	**47.1K**	54.7K	-7.6K
PSO hard (5 c)	ES	DE	YES	59.9K	**51.3K**	+8.6K	**45.2K**	55.2K	-10.0K	82.7K	93.2K	-10.5K
DE easy (1 c)	DE	DE	NO	**41.2K**	**48.9K**	-7.7K	78.2K	85.2K	-7K	67.2K	79.1K	-11.9K
ES easy (1 c)	ES	ES	NO	68.2K	81.4K	-13.2K	**64.2K**	**54.7K**	9.6K	72.9K	81.7K	-8.8K
PSO easy (1 c)	ES	PSO	YES	81.2K	87.1K	4.1K	**62.1K**	84.2K	-22.1K	63.4K	**41.9K**	21.5K
DE easy (2 c)	DE	DE	NO	**54.2K**	**44.5K**	9.7K	68.4K	83.1K	-14.7K	71.4K	81.6K	-10.2K
ES easy (2 c)	PSO	ES	YES	75.3K	83.6K	-8.3K	69.2K	**55.3K**	13.9K	**65.1K**	78.9K	-13.8K
PSO easy (2 c)	PSO	PSO	YES	68.3K	81.4K	-13.1K	94.1K	83.9K	10.2K	**58.1K**	**49.4K**	8.7K
DE easy (3 c)	DE	DE	NO	**59.1K**	**48.4K**	10.7K	78.2K	89.4K	-11.2K	80.4K	84.5K	-4.1K
ES easy (3 c)	PSO	ES	YES	73.9K	87.8K	-13.9K	55.2K	**46.2K**	9K	**42.4K**	48.0K	-5.6K
PSO easy (3 c)	PSO	PSO	NO	69.2K	89.5K	-20.3K	76.4K	87.3K	-10.9K	**58.2K**	**49.1K**	9.1K
DE easy (4 c)	DE	DE	NO	**64.2K**	**53.2K**	11K	78.3K	85.1K	-6.8K	87.3K	93.8K	-6.5K
ES easy (4 c)	ES	ES	NO	**78.3K**	89.4K	-11.1K	59.1K	**48.9K**	10.2K	82.5K	87.2K	-4.7K
PSO easy (4 c)	PSO	DE	YES	61.3K	**55.5K**	5.8K	85.9K	92.4K	-6.5K	**60.4K**	68.9K	-8.5K
DE easy (5 c)	PSO	DE	YES	71.4K	**57.8K**	13.6K	99.1K	95.3K	3.8K	**70.9K**	81.5K	-10.6K
ES easy (5 c)	ES	ES	NO	86.9K	91.9K	-5.0K	62.3K	**55.7K**	6.6K	78.9K	81.1K	-2.2K
PSO easy (5 c)	PSO	PSO	NO	84.1K	91.3K	-7.2K	76.9K	82.9K	-6.0K	63.4K	**57.6K**	5.8K
Rnd. 1 (1 c)	PSO	PSO	NO	59.1K	65.7K	-6.6K	51.3K	56.9K	-5.6K	**50.9K**	**53.2K**	-2.3K
Rnd. 2 (1 c)	DE	DE	NO	**55.2K**	**61.9K**	-6.7K	67.4K	64.2K	+3.2K	69.8K	65.2K	+4.6K
Rnd. 3 (2 c)	DE	DE	NO	**49.2K**	**59.8K**	-10.6K	57.9K	65.3K	-7.4K	73.4K	69.6K	+3.8K
Rnd. 4 (2 c)	ES	ES	NO	64.1K	72.2K	-8.1K	**63.2K**	**59.4K**	+3.8	65.3K	71.4K	-6.1K
Rnd. 5 (3 c)	ES	DE	YES	68.3K	**65.4K**	+2.9K	**63.5K**	66.9K	-3.4K	69.9K	74.6K	-4.7K
Rnd. 6 (3 c)	PSO	PSO	NO	73.9K	82.4K	-8.5K	66.2K	78.3K	-12.1K	**56.8K**	**61.4K**	-4.6K
Rnd. 7 (4 c)	PSO	ES	YES	78.3K	83.2K	-4.9K	73.5K	**65.8K**	+7.7K	**71.2K**	78.9K	-7.7K
Rnd. 8 (4 c)	ES	DE	YES	76.9K	**71.1K**	+5.8K	**68.3K**	79.2K	-10.9K	76.3K	73.2K	+3.1K
Rnd. 9 (5 c)	DE	DE	NO	**88.3K**	**82.7K**	+5.6K	90.9K	93.8K	-2.9K	90.2K	97.3K	-7.1K
Rnd. 10 (5 c)	PSO	PSO	NO	80.8K	93.2K	-12.4K	83.5K	89.5K	-6K	**75.3K**	**68.4K**	+6.9K
Rnd. 11 (1 c)	ES	DE	YES	52.9K	**44.6K**	+8.3K	**52.4K**	59.2K	-6.8K	54.3K	64.2K	-9.9K
Rnd. 12 (1 c)	DE	DE	NO	**50.2K**	**45.2K**	+5K	51.2K	58.9K	-7.7K	62.5K	69.4K	-6.9K
Rnd. 13 (2 c)	PSO	PSO	NO	67.3K	61.2K	6.1K	60.8K	64.3K	-3.5K	**49.2K**	**59.1K**	-9.9K
Rnd. 14 (2 c)	PSO	PSO	NO	70.3K	79.0K	-8.7K	68.3K	75.3K	-7K	**57.9K**	**63.4K**	-5.5K
Rnd. 15 (3 c)	ES	PSO	YES	66.9K	71.9K	-5K	**55.2K**	79.3K	-24.1K	58.3K	**65.3K**	-7K
Rnd. 16 (3 c)	DE	DE	NO	**61.2K**	**69.2K**	-8K	70.5K	79.3K	-8.8K	73.9K	70.1K	3.8K
Rnd. 17 (4 c)	ES	ES	NO	67.9K	73.9K	-6K	**65.2K**	**75.9K**	-10.7K	69.2K	77.4K	-8.2K
Rnd. 18 (4 c)	PSO	PSO	NO	72.7K	85.3K	-12.6K	80.3K	78.3K	+2K	**70.2K**	**68.3K**	+1.9K
Rnd. 19 (5 c)	DE	ES	YES	**83.1K**	93.9K	-10.8K	83.9K	**91.2K**	-7.3K	99.1K	95.9K	+3.2K
Rnd. 20 (5 c)	PSO	PSO	NO	84.1K	90.9K	-6.8K	85.9K	93.2K	-7.3K	**83.5K**	**87.3K**	-3.8K

Table 4: Predicted and actual most suited algorithm type/required FEN for Sphere function with linear constraint(s). The prediction model is trained with combination of pareto front and random points (PFR-PM) from multi objective evolver population. DE hard/easy (1 C) is a problem instance that is hard/easy for DE algorithm but easy/hard for the others.

Instances name	Predicted alg.	Actual alg.	Error	Predicted FEN for DE	Actual FEN for DE	Error for DE	Predicted FEN for ES	Actual FEN for ES	Error for ES	Predicted FEN for PSO	Actual FEN for PSO	Error for PSO
DE hard (1 c)	ES	ES	NO	83.8K	86.3K	-2.5K	**40.2K**	**41.5K**	-1.3K	45.8K	43.2K	2.6K
ES hard (1 c)	DE	DE	NO	**43.6K**	**45.7K**	-2.1K	80.2K	84.2K	-4.0K	45.1K	48.3K	-3.2K
PSO hard (1 c)	DE	DE	NO	**39.1K**	**37.2K**	1.9K	42.4K	41.8K	0.6K	78.1K	80.1K	-2.0K
DE hard (2 c)	PSO	ES	YES	85.2K	87.4K	-2.2K	43.2K	**45.2K**	-2.0K	**41.9K**	43.6K	-1.7K
ES hard (2 c)	DE	DE	NO	**45.8K**	**46.4K**	-0.6K	85.3K	88.3K	-3.0K	48.9K	46.8K	2.1K
PSO hard (2 c)	DE	DE	NO	**41.8K**	**43.6K**	-1.8K	47.9K	45.1K	2.8K	85.3K	83.2K	2.1K
DE hard (3 c)	ES	ES	NO	89.5K	92.4K	-2.9K	**43.1K**	**45.2K**	-2.1K	44.2K	47.2K	-3.0K
ES hard (3 c)	PSO	PSO	NO	47.8K	51.2K	-3.4K	94.7K	91.2K	3.5K	**42.9K**	**48.2K**	-5.3K
PSO hard (3 c)	DE	DE	NO	**46.8K**	**49.6K**	-2.8K	50.2K	53.5K	-3.3K	92.5K	89.5K	3.0K
DE hard (4 c)	ES	ES	NO	96.2K	93.7K	2.5K	**49.2K**	**47.2K**	2.0K	51.2K	50.2K	1.0K
ES hard (4 c)	PSO	PSO	NO	50.4K	49.2K	1.2K	84.7K	89.1K	-4.4K	**46.8K**	**48.9K**	-2.1K
PSO hard (4 c)	ES	ES	NO	**50.3K**	52.7K	-2.4K	53.6K	**50.8K**	2.8K	90.2K	87.5K	2.7K
DE hard (5 c)	PSO	ES	YES	97.2K	96.3K	0.9K	55.9K	**53.3K**	2.6K	**52.4K**	55.3K	-2.9K
ES hard (5 c)	DE	DE	NO	**52.5K**	**53.8K**	-1.3K	94.2K	92.6K	1.6K	53.8K	54.7K	-0.9K
PSO hard (5 c)	DE	DE	NO	**50.2K**	**51.3K**	-1.1K	56.3K	55.2K	1.1K	90.5K	93.2K	-2.7K
DE easy (1 c)	DE	DE	NO	**46.3K**	**48.9K**	-2.6K	82.7K	85.2K	-2.5K	76.2K	79.1K	-2.9K
ES easy (1 c)	ES	ES	NO	86.6K	81.4K	5.2K	**48.2K**	**54.7K**	-6.5K	80.2K	81.7K	-1.5K
PSO easy (1 c)	PSO	PSO	NO	86.3K	87.1K	-0.8K	82.4K	84.2K	-1.8K	**45.7K**	**41.9K**	3.8K
DE easy (2 c)	DE	DE	NO	**48.1K**	**44.5K**	3.6K	87.4K	83.1K	4.3K	79.4K	81.6K	-2.2K
ES easy (2 c)	ES	ES	NO	77.2K	83.6K	-6.4K	**52.3K**	**55.3K**	-3.0K	77.4K	78.9K	-1.5K
PSO easy (2 c)	PSO	PSO	NO	83.2K	81.4K	1.8K	78.4K	83.9K	-5.5K	**48.1K**	**49.4K**	-1.3K
DE easy (3 c)	DE	DE	NO	**42.5K**	**48.4K**	-5.9K	86.8K	89.4K	-2.6K	87.4K	84.5K	2.9K
ES easy (3 c)	PSO	ES	YES	90.9K	87.8K	3.1K	48.9K	**46.2K**	2.7K	**46.9K**	48.0K	-1.1K
PSO easy (3 c)	PSO	PSO	NO	87.8K	89.5K	-1.7K	92.4K	87.3K	5.1K	47.0K	**49.1K**	-2.1K
DE easy (4 c)	DE	DE	NO	**50.2K**	**53.2K**	-3.0K	83.4K	85.1K	-1.7K	91.8K	93.8K	-2.0K
ES easy (4 c)	ES	ES	NO	86.7K	89.4K	-2.7K	**50.8K**	**48.9K**	1.9K	89.8K	87.2K	2.6K
PSO easy (4 c)	DE	DE	NO	56.7K	**55.5K**	1.2K	87.9K	92.4K	-4.5K	**55.2K**	68.9K	-13.7K
DE easy (5 c)	DE	DE	NO	**55.9K**	**57.8K**	-1.9K	93.1K	95.3K	-2.2K	82.4K	81.5K	0.9K
ES easy (5 c)	ES	ES	NO	89.4K	91.9K	-2.5K	**56.3K**	**55.7K**	0.6K	85.9K	81.1K	4.8K
PSO easy (5 c)	PSO	PSO	NO	92.4K	91.3K	1.1K	84.1K	82.9K	1.2K	**56.1K**	**57.6K**	-1.5K
Rnd. 1 (1 c)	PSO	PSO	NO	63.4K	65.7K	-2.3K	55.4K	56.9K	-1.5K	**51.7K**	**53.2K**	-1.5K
Rnd. 2 (1 c)	DE	DE	NO	**64.6K**	**61.9K**	+2.7K	64.7K	64.2K	+0.5K	65.7K	65.2K	+0.5K
Rnd. 3 (2 c)	DE	DE	NO	**55.0K**	**59.8K**	-4.8K	62.3K	65.3K	-3.0K	67.1K	69.6K	-2.5K
Rnd. 4 (2 c)	ES	ES	NO	68.6K	72.2K	-3.6K	**61.4K**	**59.4K**	+2.0K	68.2K	71.4K	-3.2K
Rnd. 5 (3 c)	DE	DE	NO	**61.9K**	**65.4K**	-3.5K	62.5K	66.9K	-4.4K	70.8K	74.6K	-3.8K
Rnd. 6 (3 c)	PSO	PSO	NO	78.3K	82.4K	-4.1K	73.9K	78.3K	-4.4K	**57.8K**	**61.4K**	-3.6K
Rnd. 7 (4 c)	ES	ES	NO	86.4K	83.2K	+3.2K	**69.4K**	**65.8K**	+3.6K	74.7K	78.9K	-4.2K
Rnd. 8 (4 c)	ES	DE	YES	74.6K	**71.1K**	+3.5K	**73.7K**	79.2K	-5.5K	75.8K	73.2K	+2.6K
Rnd. 9 (5 c)	DE	DE	NO	**84.2K**	**82.7K**	+1.5K	87.6K	93.8K	-6.2K	94.2K	97.3K	-3.1K
Rnd. 10 (5 c)	PSO	PSO	NO	83.7K	93.2K	-9.5K	84.3K	89.5K	-5.2K	**65.8K**	**68.4K**	-2.6K
Rnd. 11 (1 c)	DE	DE	NO	**48.5K**	**44.6K**	+3.9K	55.8K	59.2K	-0.4K	60.9K	64.2K	-3.3K
Rnd. 12 (1 c)	DE	DE	NO	**48.7K**	**45.2K**	+3.5K	53.6K	58.9K	-5.3K	63.6K	69.4K	-5.8K
Rnd. 13 (2 c)	PSO	PSO	NO	55.7K	61.2K	-5.5K	65.9K	64.3K	+1.6K	**54.9K**	**59.1K**	-4.2K
Rnd. 14 (2 c)	PSO	PSO	NO	76.0K	79.1K	-3.1K	66.6K	75.3K	-8.7K	**61.7K**	**63.4K**	-1.7K
Rnd. 15 (3 c)	ES	PSO	YES	68.8K	71.9K	-3.1K	**62.7K**	79.3K	-16.6K	63.1K	**65.3K**	-2.2K
Rnd. 16 (3 c)	DE	DE	NO	**66.5K**	**69.2K**	-2.7K	72.2K	79.3K	-7.1K	71.5K	**70.1K**	+3.4K
Rnd. 17 (4 c)	ES	ES	NO	70.4K	73.9K	-3.5K	**70.3K**	**75.9K**	-5.6K	71.6K	77.4K	-5.8K
Rnd. 18 (4 c)	PSO	PSO	NO	75.4K	85.3K	-9.9K	83.5K	78.3K	+5.2K	**71.2K**	**68.3K**	+2.9K
Rnd. 19 (5 c)	DE	ES	YES	**86.9K**	93.9K	-7.0K	88.5K	**91.2K**	-2.7L	97.4K	95.9K	-1.5K
Rnd. 20 (5 c)	PSO	PSO	NO	87.4K	90.9K	-3.5K	90.3K	93.2K	-2.9K	**85.9K**	**87.3K**	-1.4K

Table 5: Comparison of PFR-PM with RO-PM models for Sphere function with various types of constraints (linear, quadratic and their combination). Average deviation of FEN denotes the average of differences between actual and predicted required FEN for PFR-PM and RO-PM.

Problem	Success Rate RO-PM	Success Rate FR-PM	Average deviation of FEN for RO-PM	Average deviation of FEN for PFR-PM	P value
Sphere, 1lin	2	26	7.8K	2.4K	0.004
Sphere, 2lin	1	27	8.6K	1.8K	0.013
Sphere, 3lin	1	26	12.6K	3.2K	0.018
Sphere, 4lin	5	28	13.6K	3.5K	0.006
Sphere, 5lin	1	28	17.4K	2.7K	0.028
Sphere, 1Quad	1	27	11.9K	2.1K	0.035
Sphere, 2Quad	1	26	13.4K	2.6K	0.038
Sphere, 3Quad	3	28	15.8K	3.7K	0.043
Sphere, 4Quad	1	29	19.3K	3.1K	0.026
Sphere, 5Quad	5	28	21.6K	4.3K	0.035
Sphere, 4Lin, 1Quad	2	24	13.4K	2.5K	0.007
Sphere, 3Lin, 2Quad	2	26	13.8K	2.8K	0.004
Sphere, 2Lin, 3Quad	3	28	16.1K	3.8K	0.031
Sphere, 1Lin, 4Quad	5	27	18.9K	3.7K	0.016

Table 6: Comparison of PFR-PM with RO-PM models for Ackley function with various types of constraints (linear, quadratic and their combination). Average deviation of FEN denotes the average of differences between actual and predicted required FEN for PFR-PM and RO-PM.

Problem	Success Rate RO-PM	Success Rate FR-PM	Average deviation of FEN for RO-PM	Average deviation of FEN for PFR-PM	P value
Ackley, 1lin	0	27	9.3K	2.5K	0.043
Ackley, 2lin	1	27	11.5K	3.2K	0.016
Ackley, 3lin	2	25	10.3K	2.7K	0.004
Ackley, 4lin	1	28	14.7K	3.6K	0.008
Ackley, 5lin	6	29	13.8K	4.5K	0.025
Ackley, 1Quad	2	29	16.3K	3.5K	0.046
Ackley, 2Quad	1	27	17.7K	4.1K	0.026
Ackley, 3Quad	0	25	18.3K	3.7K	0.043
Ackley, 4Quad	3	27	16.9K	5.1K	0.048
Ackley, 5Quad	2	29	21.9K	5.8K	0.034
Ackley, 4Lin, 1Quad	4	24	15.8K	4.2K	0.032
Ackley, 3Lin, 2Quad	2	26	16.7K	4.6K	0.012
Ackley, 2Lin, 3Quad	3	24	16.8K	4.9K	0.006
Ackley, 1Lin, 4Quad	0	28	19.8K	4.3K	0.021

Table 7: Comparison of PFR-PM with RO-PM models for Rosenbrock function with various types of constraints (linear, quadratic and their combination). Average deviation of FEN denotes the average of differences between actual and predicted required FEN for PFR-PM and RO-PM.

Problem	Success Rate RO-PM	Success Rate FR-PM	Average deviation of FEN for RO-PM	Average deviation of FEN for PFR-PM	P value
Rosenbrock, 1lin	2	26	10.3K	3.3K	0.038
Rosenbrock, 2lin	0	26	11.5K	4.6K	0.035
Rosenbrock, 3lin	3	25	12.7K	3.6K	0.002
Rosenbrock, 4lin	4	27	15.8K	5.2K	0.035
Rosenbrock, 5lin	5	28	19.4K	5.1K	0.028
Rosenbrock, 1Quad	2	27	17.4K	4.1K	0.017
Rosenbrock, 2Quad	1	29	21.5K	4.7K	0.043
Rosenbrock, 3Quad	4	26	21.3K	5.7K	0.037
Rosenbrock, 4Quad	3	28	18.5K	5.2K	0.043
Rosenbrock, 5Quad	2	28	24.6K	6.9K	0.004
Rosenbrock, 4Lin, 1Quad	1	28	14.7K	3.6K	0.004
Rosenbrock, 3Lin, 2Quad	0	24	17.4K	4.7K	0.024
Rosenbrock, 2Lin, 3Quad	4	25	19.5K	3.6K	0.029
Rosenbrock, 1Lin, 4Quad	2	27	21.3K	5.2K	0.006

Chapter 7

Conclusions

In this thesis, we have presented new insights into feature-based analysis for bio-inspired algorithms for constrained continuous optimisation problems. We introduced a new feature-based analysis for constrained optimisation problems in which the relations between constraint features and algorithms are studied. Then, by analysing these features, we have designed an automated feature-based algorithm selection model to predict the performance of the most suited algorithm for a given problems.

In Chapter 2, we conducted a survey of existing theoretical analysis techniques for fitness landscapes. We discussed ruggedness quantifying and its application in combinatorial or unconstrained real-valued optimisation problems. However, applying the existing techniques on constrained problems provides an inaccurate ruggedness value. Hence, we implemented a technique with a biased walking method to measure the fitness landscape ruggedness for constrained continuous optimisation problems.

Taking a more practical analysis perspective, we provided new insight into real-valued constrained optimisation problems. Constraints are very important in constrained optimisation problems and this motivation led us to implement the first feature-based analysis on COPs using an evolving approach. By applying this approach we generated easy and hard COP instances to analyse the features of a set of linear and/or quadratic (and their combination) constraints. By analysing the constraints for the diverse set of evolved COP instances, we identified the features of the constraints that influence the problem difficulty (in Chapter 3, 4). We extended the study by conducting a feature-based comparison of various types of evolutionary algorithms such as DE, ES and PSO. In Chapter 5, we then used a multi-objective algorithm as an evolver to generate instances that are hard/easy for one algorithm, but still easy/hard for the others. Analysing the constraint features of these evolved instances helps us extracting the knowledge regarding the strengths and weaknesses of various algorithm types by examining why an algorithm fails while the other performs better.

Building on the evolved instances from evolving approach, we consider how a prediction model can predict a most suited algorithm type for a given COP. Experimenting various subsets of evolved instances from multi-objective evolver population for the model training phase, we identified its best customisation. Based on the results in Chapter 6, using a combination of Pareto front and more random instances from a multi-objective evolver in the learning phase increased the reliability of the prediction model. The experimental evaluation showed the accuracy of the proposed prediction model and its ability to predict the right algorithm type, based on the constraint features of a given COP.

In summary, the feature-based analysis provides new insights into constrained optimisation problems. The novel feature-based analysis of COPs not only provides guidance to implement a prediction model based on the knowledge, but also can help practitioners to study the strengths and weaknesses of the evolutionary algorithms. This study was the first step in COP feature-based analysis with its ability to find the relationships between algorithm performances and constraint features. By using knowledge from evolved instances, the outcome of the study shows how to use an evolving approach and features of COPs to design a successful algorithm selection framework. Finally, the prediction model provides valuable insights into the algorithm themselves, in terms of the kinds of problems that an algorithm would potentially struggle to solve.