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

Shayan Poursoltan
Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, SA 5005, Australia
Email: shayan.poursoltan@adelaide.edu.au

Frank Neumann
Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, SA 5005, Australia
Email: frank.neumann@adelaide.edu.au

***Abstract*—Feature-based analysis has provided new insights into what characteristics make a problem hard or easy for a given algorithms. Studies, so far, considered unconstrained continuous optimisation problem and classical combinatorial optimisation problems such as the Travelling Salesperson problem. In this paper, we present a first feature-based analysis for constrained continuous optimisation. To start the feature-based analysis of constrained continuous optimization, we examine how linear constraints can influence the optimisation behaviour of the well-known ε-constrained differential evolution algorithm. Evolving the coefficients of a linear constraint, we show that even the type of one linear constraint can make a difference of 10-30% in terms of function evaluations for well-known continuous benchmark functions.**

Keywords: Constraints, Continuous Optimisation, Difficulty Prediction, Linear Constraints, Features

I. Introduction

Constrained optimisation problems (COP)s are important and widespread in the real world [1]. This has motivated the development of algorithmic approaches to tackle the constrained optimisation problems. A major component of these algorithms is a mechanism to handle the problem constraints. Various evolutionary algorithms such as differential evolution (DE) [2], particle swarm optimisation (PSO) [3] and evolutionary strategies (ES) [4] have been used to solve COPs. To deal with constraints, these algorithms use constraint handling techniques such as penalty functions, decoder based methods or special operators that separate the treatment of the constraints and objective function. For a comprehensive presentation on the different constraint handling methods, we refer the reader to [5].

There has been always been the question which algorithm from a suite of algorithms is most suitable for a given problem. Mersmann et al. [6] proposed the following steps to answer this question. First, we need to analyse the performance of different algorithms in dependence of problem features. Second, one has to find and extract the problem features that determine problem difficulty.

The idea of testing algorithms on set of test problems with different features was initially proposed in [7] which introduced a benchmark set for constrained continuous optimisation. The aim was to test the ability of different algorithms on a variety of constrained optimisation problems. The features related to these benchmark problems consist of an objective function type (linear, nonlinear), a type of constraints (linear, nonlinear, inequality or equality) and the feasibility ratio $\rho = |F|/|S|$ of the problem. Later, other functions were added to this benchmark set in order to address other features such as the disjoint feasible area and the combination of linear constraints [8]. Furthermore, there have been additional benchmark problems proposed to evaluate the evolutionary algorithm performances [9], [10].

Recently, there has been an increasing interest to understanding the features that make a problem difficult to solve [11]. For continuous problems, test case generators (TCGs) [12], [13] were proposed to study the influence of the combination of problem features on problem difficulty. The TCG's approach is to generate different problems by varying features such as dimensionality, multi modality, size of feasible regions, number and type of constraints. Also, many approaches have been introduced for discrete problems [14], [15], [16], [17]. The example of such a study is the one for the TSP problem [14]. Their idea is to generate hard and easy problem instances by evolving them. This approach is using an evolutionary algorithm to obtain diverse set of hard and easy instances for a certain algorithm [14]. By analysing these instances, it is possible identify features which determine problem hardness

In this paper, we adopt the evolving approach to ensure that the sets of instances are varied from hard to easier ones for constrained continuous optimisation problems. We start by investigating several benchmark functions under one linear constrained. Among several features of the problems, constraints play a vital role in problem hardness. Hence, our method is to evolve easy and hard instances to investigate which features of a linear constraint correlate with instance difficulty. To achieve this information, we use a suitable evolutionary algorithm that handles the constraints. We use the ε-constrained differential evolution with an archive and gradient-based mutation (εDEag) [18] that has better performance than the other competitors based on the CEC 10 special session results [9].

Our results show the effectiveness of linear constraint features on making a problem easy or hard. Analysing these features by testing the evolved instances on (εDEag), provides knowledge on influence of the constraints on the constraint optimisation problems. This type of information is a successful key to design an automated algorithm selection system and sets the basis for studies on the impact of other types of constraints.

The remainder of this paper is as follows: In Section 2 we introduce the concept of constrained continuous optimisation and the algorithms that we use to evolve and solve the problem instances. Our approach to investigate the linear constraints and their effects on problem hardness is presented in Section 3. In Section 4 we carry out the analysis of the experimental results. Finally, we conclude with some remarks and topics for future work.

II. PRELIMINARIES

A. *Constrained continuous optimisation problems*

Constrained continuous optimisation problems are optimisation problems where a function on real-valued variables should be optimized with respect to a given set of constraints. Constraints are usually given by a set of inequalities and/or equalities.

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:

$$\text{Minimize} \quad f(x), \quad x = (x_1, \ldots, x_n) \in \mathbb{R}^n \tag{1}$$

such that $x \in S \cap F$. 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 values of l_i and u_i are lower and upper bounds on the ith variable, respectively. Additional constraints are given by the functions

$$\begin{aligned} g_i(x) &\leq 0 \quad \forall i \in \{1, \ldots, q\}, and \\ h_j(x) &= 0 \quad \forall j \in \{q+1, \ldots, p\} \end{aligned} \tag{3}$$

where both $g_i(x)$ and $h_j(x)$ could be linear or nonlinear. In this research, we consider the constraint as linear inequality.

B. *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 uses the ε-constrained method [19] to transform algorithms for unconstrained problems to constrained ones. The ε-constrained method converts a constrained optimisation problem to an unconstrained one by using ε-level comparison instead of ordinary one. The ε-level comparison is done in lexicographic order in which ϕ (constraint violation) proceeds f (function value) since feasibility has higher priority. For any $\varepsilon \geq 0$, the ε-level comparison of two candidates (f_1, ϕ_1) and (f_2, ϕ_2) is described as the following:

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

```
1) Initialize archive of $A$ with $M$ randomly selected
   individuals from search space $S$.
2) Initialize $\varepsilon$ level Using control level function.
3) Initialize population by selecting top $N$ individuals
   from archive $A$. Individuals are ranked based on the
   $\varepsilon$level comparison.
4) Setting the termination condition: when it exceeds
   Maximum function evaluation number.
5) DE Operation: compare child and parent
   based on $\varepsilon$ level comparison
6) If child is infeasible, it is changed by the gradient
   based mutation with probability P. Go to step 4
7) Control the $\varepsilon$-level
8) Go to step 3
```

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

and

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

By adopting an archive (see Algorithm 1) to the simple (εDEg) [20], the stability, usability and efficiency of the algorithm has been increased [21], [22]. Using an archive improves the diversity of individuals (see Algorithm 1). The offspring generation is adopted in such a way that if the child is not better than its parent, the parent generates another one. This leads to more stability to the algorithm.

III. EVOLVING CONSTRAINTS

In this study, we focus on finding the influence of constraint features on problems. We want to obtain knowledge of what types of constraints or what features of them make a problem difficult to evolutionary algorithm.

While what makes a constrained problem difficult is not a standalone feature, it is worth noting that the most important part of these problems (constraints), need to be studied in detail. Hence, our approach in this study is analysing various effects of constraints on problem hardness. In this research, we consider the linear inequality constraint. A linear inequality constraint is as the form of

$$\begin{aligned} g(x) &= b + a_1 x_1 + \ldots + a_n x_n \\ lc_i \leq a_i &\leq uc_i, \qquad 1 \leq i \leq n \end{aligned} \tag{4}$$

where values of lc_i and uc_i are lower and upper bounds on the coefficients a_i and $x_1, x_2 \ldots, x_n$ are values from equation 1, respectively. Also, we consider $b \leq 0$ so that the objective function optimum is always feasible. In this paper, we investigate the relations of linear constraint coefficients (a_i) and their capacity to control problem difficulty.

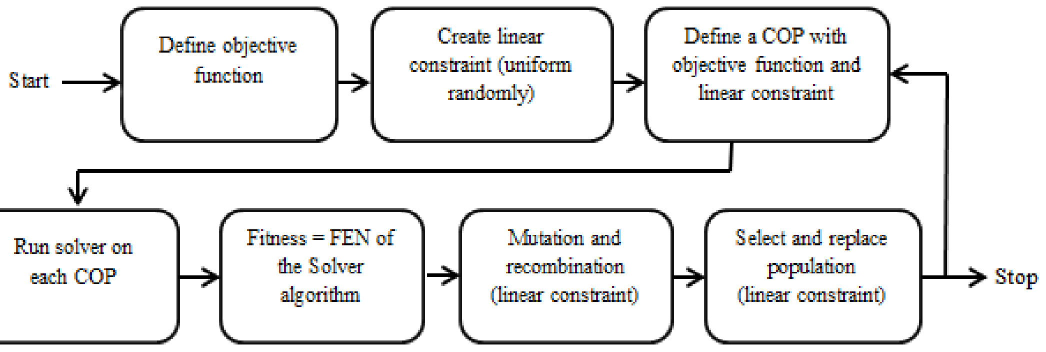

Start
Define objective function
Create linear constraint (uniform randomly)
Define a COP with objective function and linear constraint
Run solver on each COP
Fitness = FEN of the Solver algorithm
Mutation and recombination (linear constraint)
Select and replace population (linear constraint)
Stop

Fig. 1: Evolving constraints process

A. *Methodology*

This experimental study is based on generating easy and hard instances using the performance of another optimisation algorithm (problem solver). Using this technique makes it possible to generate easy and hard instances.

In order to investigate the effects of constraints, first we need to define variety of them with fixed objective function (see Figure 1). Then, the evolutionary algorithm is started by uniform randomly choosing constraint linear coefficients a_i. Next, we define a constraint optimisation problem using a fixed objective function and newly generated constraint. This forms the first step of the evolving process. We then solve the generated COP with the investigated evolutionary algorithm (solver). The function evaluation number (defined in next section) that is required for solving the COP is considered as the fitness value for evolving process. This generational process is repeated to find the hard and easy linear constraint for the fixed objective function.

B. *Algorithms*

For the evolution process, we use differential evolution [2], which is a reliable and versatile function optimiser. DE evolves individuals towards global optimum using mutation, crossover and selection process. The core of DE is based on enhancing the individual differences. The algorithm of differential evolution is described in Algorithm 2,3. The *Cost* function in Algorithm 2 is equivalent to the function evaluation number (FEN) of solver required to solve the COP. As mentioned in Algorithm 2, best solution is easy instances. In order to find hard instances, line 7 needs to be modified as Cost(S_i)>Cost(P_i).

Our chosen algorithm for the solver is εDEag. As we discussed in the previous section, εDEag is a stable optimisation algorithm as evidenced by its results in the CEC 2010 competition [9]. We also, modified the εDEag termination condition in which the solver should be terminated when it reaches FEN_{max} or

$$f(x_{optimum}) - f(x_{best}) \leq e^{-12} \tag{5}$$

Algorithm 2: Differential evolution (DE) algorithm. The Cost function indicates the number of required FEN, that is used to solve the generated instance.

```
1)  inputs: Problem and population size, $Crossover_{rate}$, $weighting_{factor}$
    outputs: $S_{best}$
2)  Population ← InitializePopulation
    EvaluatePopulation(population)
    $S_{best}$ ← GetBestSolution(Population)
3)  Repeat
4)    NewPopulation ← $\phi$
5)    For i starts from 1 to < $Population_{size}$-1
6)      $S_i$ ← Newsample
7)      If Cost($S_i$)≤Cost($P_i$)
8)        NewPopulation ← $S_i$
9)      else
10)       NewPopulation ← $P_i$
11)     Endif
12)   Endfor
13)   Population ← NewPopulation
14)   EvaluatePopulation(population)
15)   $S_{best}$ ← GetBestSolution(Population)
16) Until (stop condition)
```

which means the solution x_{best} is close enough to the optimum solution. Hence, the current function evaluation number (FEN) is considered as the evolving process fitness value. Clearly, the instances that require more FEN are harder to solve comparing to easier ones. This model is repeated until certain number of generations for the DE evolutionary algorithm.

C. *Linear constraint features*

We study statistic based features that lead to generating easy and hard problem instances. In the following the complete features of the linear constraints are discussed.

- **Angle**: This feature is related to the angle of the linear constraint hyperplane and other hyperplane such as objective function and other dimension axes. To

Algorithm 3: Newsample function in Algorithm 2 (DE/rand/1/bin)

```
 1)  inputs: $P_0$, population, NP, F, CR
     outputs: $S$
 2)  **Repeat**
 3)    $P_1 \leftarrow$ RandomMember(population)
 4)  **Untill** $P_1 \neq P_1$
 5)  **Repeat**
 6)    $P_2 \leftarrow$ RandomMember(population)
 7)  **Untill** $P_2 \neq P_0 \vee P_2 \neq P_1$
 8)  **Repeat**
 9)    $P_3 \leftarrow$ RandomMember(population)
10)  **Untill** $P_3 \neq P_0 \vee P_3 \neq P_1 \vee P_3 \neq P_2$
11)  cutpoint $\leftarrow$ RandomMember(population)
12)  *Sample* $\leftarrow 0$
13)  **For** j starts from 1 to NP
14)    **If** j $\equiv$ cutpoint $\wedge$ Rand() $\leq$ CR
15)      $S_j \leftarrow P_{3_j}$ + F*($P_{1_j}$-$P_{2_j}$)
16)    **Else**
17)      $S_j \leftarrow P_{0_j}$
18)    **Endif**
19)  **Endfor**
20)  Return $S$
```

calculate the angle between two hyperplane, we need to find their normal vectors and angle between them using the following

$$\theta = \arccos \frac{n_1.n_1}{|n_1||n_2|} \tag{6}$$

where n_1,n_2 are normal vectors for two hyperplanes.

- **Ratio**: In this research, ratio is the relationship of two coefficients of a constraint (for 2 dimensional problems).

$$Ratio = \frac{a_1}{a_2} \tag{7}$$

where a_1 and a_1 are constraint coefficients (dimension=2). Since we have 2 ratios (based on two coefficients), we only consider the values which are closer to 0.

- **Shortest distance**: This feature reflects the shortest distance between the constraint hyperplane and the optimum solution. To find the shortest distance of optimum point $(x_{01}, x_{02}, \ldots, x_{0n})$ to the hyperplane $a_1x_1 + a_2x_1 + \ldots a_nx_n + b = 0$ we use

$$d_{\perp} = \frac{a_1x_{01} + a_2x_{02} + \ldots a_nx_{0n} + b}{\sqrt{a_1{}^2 + a_2{}^2 + \cdots + a_n{}^2}} \tag{8}$$

- **Constraint coefficients relationship**: Statistics regarding the mean, standard deviation, population standard deviation and variance of linear constraint coefficients. It is likely that this information has the ability to influence the problem difficulty.

IV. Experimental Investigations

We now analyse the linear constraint features for different variety of easy and hard instances. These easy or hard instances are obtained by DE algorithm based on the performance of solver algorithm (εDEag). We start by comparing hard and easy instances for well known objective functions (see Figure 2) to provide an understanding between linear constraint features and problem difficulty. We also, test our approach with a given COP to check out research results.

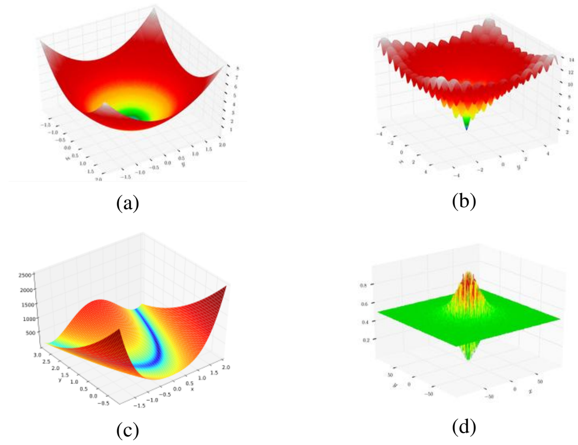


Fig. 2: Objective functions: a)Sphere: bowl shaped b)Ackley: Many Local Minima c)Rosenbrock: Valley-Shaped d)Schaffer N2: Many Local Minima

A. Experimental setup

In our experiment we generate two sets of hard and easy COP with linear constraints. In the following we discuss parameter settings for both algorithm we used:

The parameters for DE algorithm (Algorithm 2,3) are set as follows:

We use DE/rand/1/bin by performing 5000 generations for the evolving algorithm to obtain the proper linear constraint. Other parameters of the DE algorithm are as follows: population size:40, scaling factor (F): 0.9, crossover rate: 0.5. For each problem dimension we run the evolutionary algorithm for 30 independent runs (with different initial population) to evolve hard and easy instances.

The details of εDEag algorithm parameter setting is described bellow: We set the εDEag algorithm to solve generated problem for 1000 generation. Actual parameter values εDEag are: population size: 40, Scaling factor(F): 0.9, crossover rate: 0.5. The parameters for ε-constraint method are: control generation (T_c): 1000, initial ε level (θ): 0.9, Archive size: 100n (n is dimensional number), gradient-based mutation rate (P_g): 0.2 and number of repeating the mutation (R_g): 3.

B. Test problems and experimental results

In this part, we start by examining 2 dimension problems to easily discuss them on figures. Then we continue to extend the analysis on higher dimension problems to investigate the effects of constraint features on them. We first do the experiments on constrained sphere function (see Figure 2) as

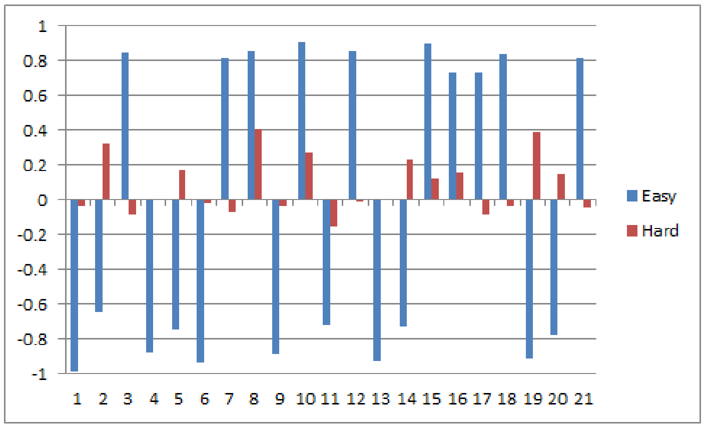


Fig. 3: Linear constraint coefficients ratio for 20 hard and easy instances - Sphere 2D

the form of:

$$\min \quad f(x) := \sum_{i=1}^{n} x_i^2 \qquad -5 \leq x_i \leq 5 \tag{9}$$

$$\text{subject to } g(x) = b + a_1 x_1 + \ldots + a_n x_n \tag{10}$$

where $g(x) \leq 0$ and coefficients (a_i) and b (Equation 4) are considered within the range of -1 $\leq a_i \leq$1 and dimension (n) is 2.

As shown in Figure 4, two different linear constraints create two COP instances. It can be observed that the feasibility ratio of easy instance is greater than hard one. Also, we compared the ratio of coefficients of linear constraints. Since the objective function is symmetric, we calculate the ratio in such a way that numerator is always less than denominator to obtain the results within [-1,1]. We run the test for 30 hard and easy instances. As observed in Figure 3, interestingly, all ratio values for easy problems are close to 1 and roughly 0 for hard one.

In contrast to coefficients ratio, there is no symmetric relationship between angle and problem hardness (FEN). Figure 5 shows the constraint hyperplane has different angles based on the chosen xy,yz,xz plane. In other words, calculating the angle between the constraint plane and various axis planes gives similar or different values according to the chosen plane (see Table I). Choosing the base hyperplane (to measure the angle with) could be more problematic when dealing with higher dimension problems.

Also, the shortest distance between optimum and the linear constraint hyperplane does not have a strong symmetric relationship to FEN of the COP instance. The results on distance feature alone, does not provide any insight into problem difficulty.

In the following, we continue the experiments on higher dimension problems to see if it holds the same pattern as above. We run our evolutionary algorithm (DE) to evolve hard and easy instances for higher dimension problems (30D,50D) with linear constraints. To study about higher dimension constraints, we calculate the standard deviation of coefficients (a_i) and

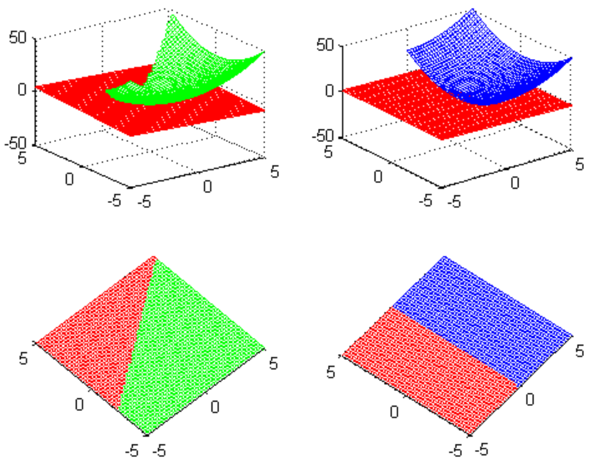


Fig. 4: Hard and easy constrained sphere problem (2D). The left (easy) and right (hard) columns have similar objective function with different linear constraint (red)

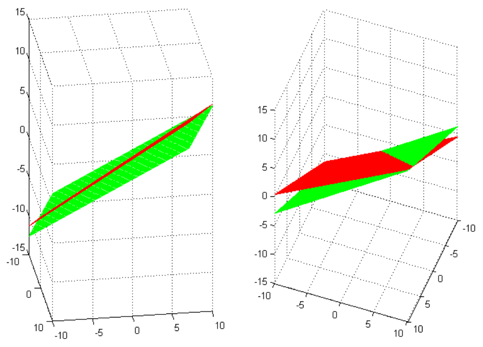
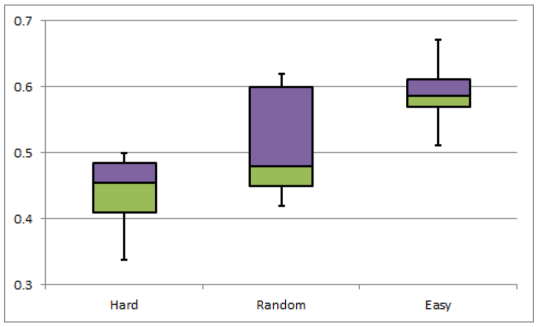


Fig. 5: Different views of easy and hard constraints angle

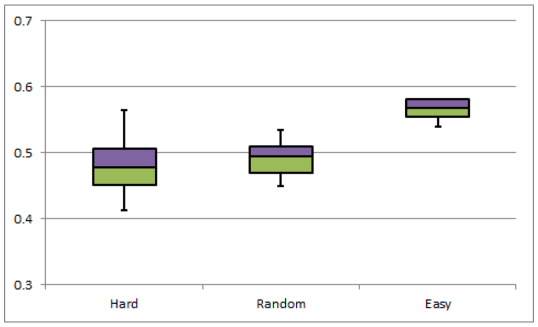
mean value for both groups of hard and easy problem instances (see Figure 6). The box plot indicates that almost in all problems (30D,50D), the standard deviation (including minimum, maximum and mean value) of coefficient a_is in easy instances are greater than hardest ones. For example, standard deviation of a_i in Sphere function (30D) varies from 0.33 to 0.50 with 0.45 median for hard instances, where as the minimum value for easy instance is 0.51 with 0.58 median.

Also it is interesting that all mean values for the constraint coefficients varies roughly from -0.3 to 0.3 (see Table II,III). These feature helps us predicting the given constraint capacity on problem hardness. Some other features such as angle and distance between optimum and constraint plane do not exhibit a symmetric relationship with problem difficulty.

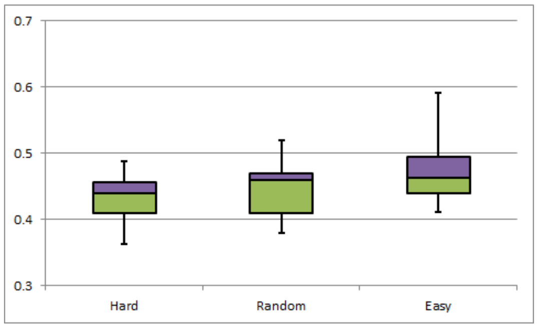
To test our results, we generated 30 random instances and evaluated them in terms of FEN and constraint coefficients. To do this, we used values chosen uniformly at random from $[-1,1]$ for the a_i. As expected the random instances lie between the results for the hard and easy instances in terms



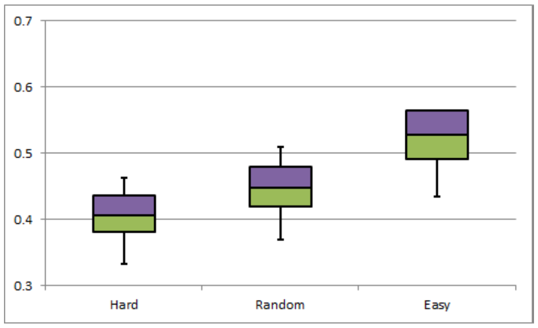
(a)



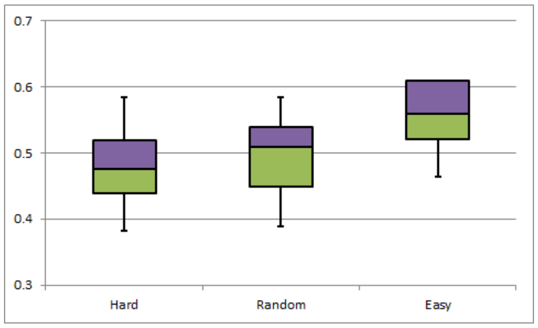
(b)



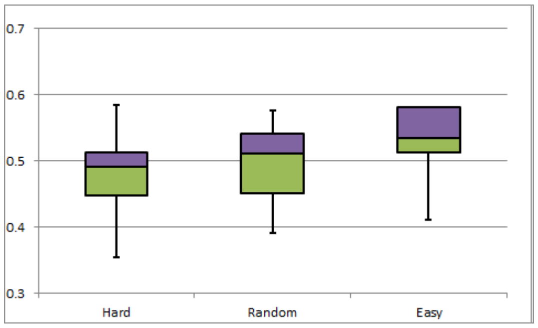
(c)



(d)



(e)



(f)

Fig. 6: Box plots of the standard deviation for constraint coefficients. From top to bottom: a)Sphere-30D b)Sphere-50D c)Rosenbrock-30D d)Rosenbrock-50D e)Ackley-30D f)Ackley-50D.

TABLE I: The angle of easy (lowest FEN) and hard (highest FEN) 2 dimensional sphere function to XY, XZ and YZ plane. The angles are measured in degrees.

COP instance	FEN	XY plane	XZ plane	YZ plane
easy	35605	41	57	67
easy	35609	39	64	68
easy	35524	36	66	64
easy	35520	20	81	72
hard	39925	14	76	89
hard	39504	39	60	66
hard	39695	40	83	51
hard	39871	39	64	62

of fitness evaluations. The results for the coefficient of the random constraints are shown in Figure 6. It can be observed that they lie between the ones for easy and hard instance as shown in the box plots.

To summarize the experiment analysis, the variation of constraint feature values over the problem hardness (function evaluation number) is more prominent in some features than the others. The results exhibit that standard deviation, mean and ratio of linear coefficients have more symmetric relationship with problem hardness than other features. For example, minimum distance of optimum and constraint hyperplane, as well as, the angle between the axis and constraint hyperplane does not provide any useful knowledge. Interestingly, increasing the number of dimension achieve almost the same results (see Figure 6). In general, these experimented values provide suggestion that which linear constraint feature has more contribution to problem difficulty.

V. Conclusions

This paper has contributed the feature-based analysis on constrained continuous optimisation problems to provide insights that which features of the problem make it hard to

TABLE II: FEN, Shortest distance value and the linear constraint coefficients (Mean, standard deviation) of different objective functions (30D)

Function	Easy/Hard	FEN	Coefficients Std	Coefficients Average	Shortest Distance
Sphere 30D	Easy	35563	0.58	0.03	-0.14
Sphere 30D	Hard	39262	0.45	-0.02	-0.06
Rosenbrock 30D	Easy	25870	0.47	0.36	-0.23
Rosenbrock 30D	Hard	39164	0.44	-0.32	-0.24
Ackley 30D	Easy	35450	0.55	-0.3	-0.03
Ackley 30D	Hard	37855	0.47	0.02	-0.06
Schwefel 30D	Easy	26343	0.73	0.44	-0.17
Schwefel 30D	Hard	39946	0.48	-0.30	-0.08
Rastrigin 30D	Easy	32786	0.69	0.11	-0.29
Rastrigin 30D	Hard	38233	0.55	-0.04	-0.02
Sum Squares 30D	Easy	35438	0.57	-0.008	-0.14
Sum Squares 30D	Hard	38206	0.43	0.002	-0.01
Dixon-Price 30D	Easy	28034	0.61	0.26	-0.19
Dixon-Price 30D	Hard	39824	0.52	0.03	-0.16
Zakharov 30D	Easy	27985	0.47	0.40	-0.41
Zakharov 30D	Hard	30340	0.41	0.3	-0.23

TABLE III: FEN, Shortest distance value and the linear constraint coefficients (Mean, standard deviation) of different objective functions (50D)

Function	Easy/Hard	FEN	Coefficients Std	Coefficients Average	Shortest Distance
Sphere 50D	Easy	36072	0.56	0.01	-0.11
Sphere 50D	Hard	39320	0.47	0.02	-0.02
Rosenbrock 50D	Easy	28845	0.52	0.15	-0.04
Rosenbrock 50D	Hard	39234	0.40	-0.26	-0.04
Ackley 50D	Easy	35872	0.54	-0.21	-0.04
Ackley 50D	Hard	39860	0.48	0.11	-0.6
Schwefel 50D	Easy	29645	0.58	0.21	-0.21
Schwefel 50D	Hard	39876	0.51	-0.11	-0.04
Rastrigin 50D	Easy	35435	0.67	0.17	-0.03
Rastrigin 50D	Hard	39854	0.53	-0.33	-0.16
Sum Square 50D	Easy	35654	0.59	0.009	-0.17
Sum Square 50D	Hard	29356	0.47	0.21	-0.14
Dixon-Price 50D	Easy	34362	0.59	-0.03	-0.001
Dixon-Price 50D	Hard	39294	0.46	0.13	-0.12
Zakharov 50D	Easy	28375	0.47	0.23	-0.09
Zakharov 50D	Hard	37287	0.43	0.31	-0.11

evolutionary algorithms. This approach has been used in the field of combinatorial optimisation, however, to the best of our knowledge, this is the first time that it has been applied in the field of constrained continuous optimisation. Hence, for the first step, we investigate that which linear constraint features influence the problem difficulty. This study used an evolutionary algorithm to generate hard and easy instances for the εDEag algorithm. We then analysed variation of linear constraint features with the problem difficulty to understand relationship between constraint features and algorithm performance. The results of this analysis, show that the coefficient of even a linear constraint can make a difference of up to 30% in terms of function evaluations and give a classification of the hardness of these constraints.

Future work will be focused on analysing polynomial constraint features and their capacity to problem difficulty. Extending these features to include other type and combination of constraints is also a future direction of this research which is worth to be carried out.

Acknowledgements

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

References

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

[2] R. Storn and K. Price, “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[3] 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*. IEEE, 1995, pp. 39–43.

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

[5] E. Mezura-Montes and C. A. Coello Coello, "Constraint-handling in nature-inspired numerical optimization: past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[6] 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*. ACM, 2011, pp. 829–836.

[7] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary computation*, vol. 4, no. 1, pp. 1–32, 1996.

[8] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 4, no. 3, pp. 284–294, 2000.

[9] 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.

[10] J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. C. Coello, and K. Deb, "Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization," *Journal of Applied Mechanics*, vol. 41, 2006.

[11] E. Mezura-Montes and C. C. Coello, "What makes a constrained problem difficult to solve by an evolutionary algorithm," Technical Report EVOCINV-01-2004, CINVESTAV-IPN, México, Tech. Rep., 2004.

[12] Z. Michalewicz, K. Deb, M. Schmidt, and T. Stidsen, "Test-case generator for nonlinear continuous parameter optimization techniques," *Evolutionary Computation, IEEE Transactions on*, vol. 4, no. 3, pp. 197–215, 2000.

[13] M. Schmidt and Z. Michalewicz, "Test-case generator tcg-2 for nonlinear parameter optimisation," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1. IEEE, 2000, pp. 728–735.

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

[15] 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*. ACM, 2013, pp. 147–160.

[16] S. Nallaperuma, M. Wagner, and F. Neumann, "Ant colony optimisation and the traveling salesperson problem: hardness, features and parameter settings," in *GECCO (Companion)*, C. Blum and E. Alba, Eds. ACM, 2013, pp. 13–14.

[17] 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*, pp. 1–32, 2013.

[18] 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*. IEEE, 2010, pp. 1–9.

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

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

[21] ——, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE, 2006, pp. 1–8.

[22] ——, "Solving difficult constrained optimization problems by the ε constrained differential evolution with gradient-based mutation," in *Constraint-Handling in Evolutionary Optimization*. Springer, 2009, pp. 51–72.