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

Shayan Poursoltan(✉) and Frank Neumann(✉)

Optimisation and Logistics, School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia
{shayan.poursoltan,frank.neumann}@adelaide.edu.au

Abstract. Different types of evolutionary algorithms have been developed for constrained continuous optimisation. We carry out a feature-based analysis of evolved constrained continuous optimisation 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 optimisation problems. The aim of these benchmarks is to find out which algorithm is good on which classes of problems. For constrained continuous optimisation problems, there has been an increasing interest to understanding problem features from a theoretical perspective [9]. The feature-based analysis 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 optimisation have recently been carried out in [4,5]. Having enough knowledge on problem properties that make it hard or easy,

S. Arik et al. (Eds.): ICONIP 2015, Part III, LNCS 9491, pp. 344–355, 2015.
DOI: 10.1007/978-3-319-26555-1_39

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 optimisation 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 COPs. 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 COP hard to solve. To provide this knowledge, we need to use a common suitable evolutionary algorithm that handles the constraints. The ε-constrained differential evolution with an archive and gradient-based mutation (εDEag) [12] 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 Sect. 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 Sect. 4, we carry out the analysis of the linear and quadratic constraint features. Finally, we conclude 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 $f: S \to \mathbb{R}$, 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 2010 CEC competition for constrained continuous problems [3]. The εDEag uses ε-constrained method to transform algorithms for unconstrained problems to constrained ones. 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:

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

In order to improve the usability, efficiency and stability of the algorithm, an archive has been applied. Using it improves the diversity of individuals. For a detailed presentation of the algorithm, we refer the reader to [12].

3 Evolving Constraints

It is assumed that the role of constraints in problem difficulty is certainly important for COP. Hence, it is necessary to analyse various effects that constraint can

impose on a constrained problem. 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 COP 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 Fig. 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] 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 εDEag as a solver. The 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.

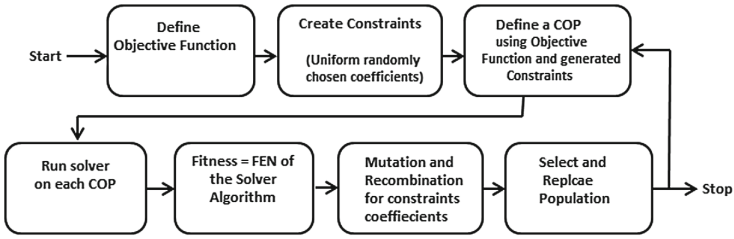

Fig. 1. Evolving constraints process

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 extract features of

constraints and analyse their effect on problem difficulty. The experimented constraints are linear and quadratic as the form of:

$$\text{linear constraint} \quad g(x) = b + a_1 x_1 + \ldots + a_n x_n \tag{4}$$

$$\text{quadratic constraint} \quad g(x) = b + a_1 x_1^2 + a_2 x_1 \ldots + a_{2n-1} x_n^2 + a_{2n} x_n \tag{5}$$

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 Eq. 1 and $a_1, a_2 \ldots, a_n$ are coefficients within the lower and upper bounds (l_c, u_c). We construct COPs where the optimum of the experimented unconstrained problem is feasible. We use quadratic functions of the form of Eq. 5 (univariate) since it is more popular in recent constrained problem benchmarks. Also, the influence of each x_ns can be analysed independently (exponent 2). The optimum of the investigated problems is $x^* = (0, \ldots, 0)$ and we ensure that this point is feasible by requiring $b \leq 0$, when evolving the constraints.

3.3 Constraints Features

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 $(b, a_1, a_2, \ldots, a_n)$ in Eqs. 4 and 5.
- **Shortest Distance:** This feature is related to the shortest distance between the objective function optimum and constraint hyperplane. 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 $(x_{01}, x_{02}, \ldots, x_{0n})$ to the linear constraint hyperplane $(a_1 x_1 + a_2 x_2 + \cdots + a_n x_n + b = 0)$ we use Eq. 6. Also, for quadratic constraint hyperplane $(a_1 x_1^2 + a_2 x_1 \ldots + a_{(2n-1)} x_n^2 + a_{2n} x_n + b = 0)$ we need to find the minimum of Eq. 7.

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

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

  where $d_\perp$ in Eq. 7 is the distance from a point to a quadratic hyperplane. Minimising the distance squared $(d_\perp^2)$ is equivalent to minimising 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 found by Eq. 8.

- **Number of Constraints:** Number of constraints plays an important role in problem difficulty. The 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, the feasibility ratio of generated COP is calculated by choosing 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 (εDEag) 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 pop size = 40, CR = 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 (εDEag) algorithm, its best parameters are chosen based on [12]. These parameters are: generation number = 1500, pop size = 40, CR = 0.5, scaling factor = 0.9. Also, the parameters for ε-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: 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 Eq. 4 with dimension (n) as 30 and all coefficients 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 (εDEag) algorithm as solver to generate more easy and hard instances. In the following we will present our findings based on various features for linear constraints (for each dimension).

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

Figure 2 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 Fig. 2) represents the results for easy and hard instances using all objective function for (εDEag) 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 problem harder or easier to solve. Interestingly, all different objective functions follow the same pattern.

Figure 3 represents variation of shortest distance to optimum feature for easy and hard instances using (εDEag) algorithm. Lower value means a higher distance from the 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. The angle between constraints in easier instances are less than higher ones (see Table 1). So, this feature is contributing in problem difficulty. 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, 10^6 points are generated within the vicinity of optimum (zero in our problems). Later, the ratios 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. Features such as, coefficients standard deviation, shortest distance, angle, 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 objective functions, dimension and coefficient range similar to linear analysis.

Table 2. The FEN for linear constraints

Constraint - Function	DE Easy	DE Hard
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	DE Easy	DE Hard
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

In the following the group of features are studied for easy and hard instances using quadratic constraints.

Observing the Fig. 2, 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 harder 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 Eq. 5). Figure 2 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 (a_{2n-1}) are more contributing than linear ones (a_{2n}) in the same quadratic constraint (see Eq. 5).

Box plots shown in Fig. 3 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. 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

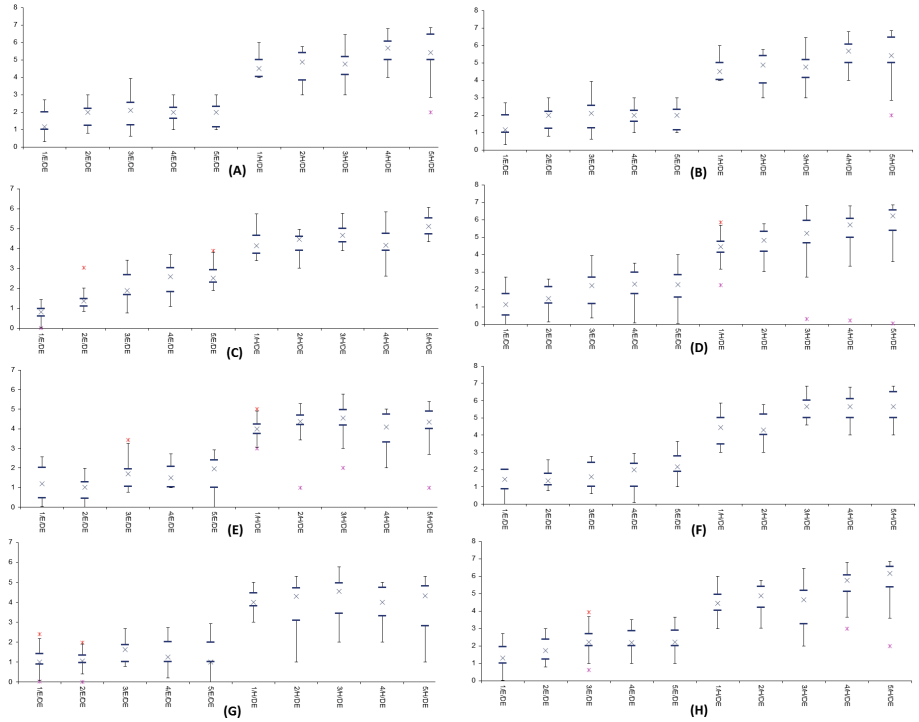

Fig. 2. Box plot for standard deviation of coefficients in linear (A,C,E,G) and quadratic (B,D,F,H) constraints for Sphere (A,B), Ackley (C,D), Rosenbrok (E,F) and Schaffer (G,H). Each sub figure includes 2 sets of hard (H) and Easy (E) instances with 1 to 5 constraints using algorithms (a/b/c denotes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

Table 3, the number of quadratic constraints is contributing to problem difficulty. It is obvious that increasing the number of quadratic constraints makes a problem harder to solve (increases FEN). As observed in Table 5, our investigations on the feasibility ratio show that increasing the number of constraints decreases the problem optimum-local feasibility ratio for easy and hard instances respectively. As it is observed, some groups of features are more contributing to problem difficulty than others. It is shown that the angle feature does not follow any systematic relationship with problem hardness for the considered algorithm in the case of quadratic constraints. On the other hand, the standard deviation, feasibility ratio and number of constraints are more influencing the performance of 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

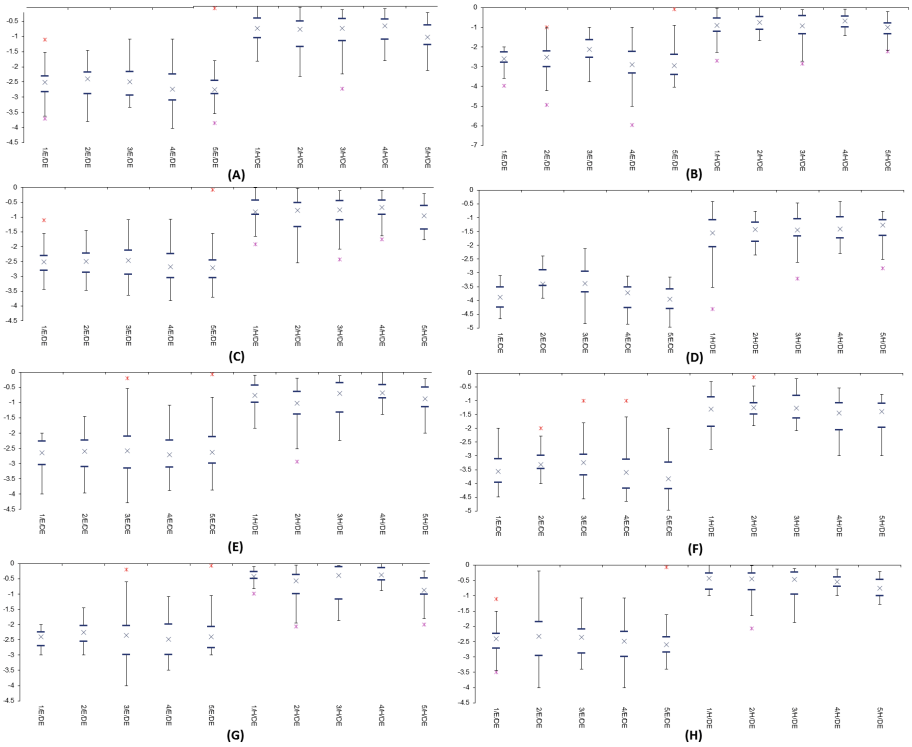

Fig. 3. Box plot for the shortest distance to optimum of linear (A,C,E,G) and quadratic (B,D,F,H) constraints for Sphere (A,B), Ackley (C,D), Rosenbrok (E,F) and Schaffer (G,H). Each sub figure includes 2 sets of hard (H) and Easy (E) instances with 1 to 5 constraints using DE algorithm (a/b/c denotes a: constraint number, b: easy/hard instances and c:algorithm)

(up to 5 constraints). The obtained results show the higher effectiveness of quadratic constraints than linear constraints. In other words, these constraints are more contributing to problem difficulty than linear ones. By analysing the various number of constraints (See Table 6) we can conclude that the 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.

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 %

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

5 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.

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

References

1. Floudas, C.A., Pardalos, P.M.: A Collection of Test Problems for Constrained Global Optimization Algorithms. LNCS, vol. 455. Springer, Heidelberg (1990)
2. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2010: Experimental setup (2010)
3. Mallipeddi, R., Suganthan, P.N.: Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization. Nanyang Technological University, Singapore (2010)
4. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 829–836. ACM (2011)
5. Mersmann, O., Preuss, M., Trautmann, H.: Benchmarking evolutionary algorithms: towards exploratory landscape analysis. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 73–82. Springer, Heidelberg (2010)

6. Mezura-Montes, E., Coello Coello, C.A.: Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm Evol. Comput. **1**(4), 173–194 (2011)
7. Nallaperuma, S., Wagner, M., Neumann, F., Bischl, B., Mersmann, O., Trautmann, H.: 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, pp. 147–160. ACM (2013)
8. Poursoltan, S., Neumann, F.: A feature-based analysis on the impact of linear constraints for ε-constrained differential evolution. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 3088–3095. IEEE (2014)
9. Poursoltan, S., Neumann, F.: Ruggedness quantifying for constrained continuous fitness landscapes. In: Datta, R., Deb, K. (eds.) Evolutionary Constrained Optimization, pp. 29–50. Springer, Heidelberg (2015)
10. Smith-Miles, K., van Hemert, J., Lim, X.Y.: Understanding TSP difficulty by learning from evolved instances. In: Blum, C., Battiti, R. (eds.) LION 4. LNCS, vol. 6073, pp. 266–280. Springer, Heidelberg (2010)
11. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
12. Takahama, T., Sakai, S.: Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1–9. IEEE (2010)