A Dynamic Penalty Function for Constrained Optimization

Chengyong Si[1(✉)], Jianqiang Shen[1], Xuan Zou[1], Yashuai Duo[1], Lei Wang[2], and Qidi Wu[2]

[1] Shanghai-Hamburg College, University of Shanghai for Science and Technology, Shanghai 200093, China
sichengyong_sh@163.com
[2] College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China
wanglei@tongji.edu.cn

Abstract. Penalty function methods have been widely used for handling constraints, but it's still a challenge about how to set the penalty parameter effectively though many related methods have been proposed. In this paper, the penalty parameter is firstly analyzed systematically by introducing four rules. Based on this analysis, a new Dynamic Penalty Function (DyPF) is proposed by adjusting penalty parameter in three different situations during the evolution (i.e., the infeasible situation, the semi-feasible situation, and the feasible situation). The experiments are designed to verify the effectiveness of our newly proposed DyPF. The results show that DyPF presents a better overall performance than other five dynamic or adaptive state-of-the-art methods in the community of constrained evolutionary optimization.

Keywords: Constrained optimization · Constraint handling techniques · Differential evolution · Dynamic penalty function (DyPF) · Ranking methods

1 Introduction

Constrained Optimization Problems (COPs) are very important and common in real-world applications. The general COPs can be formulated as follows:

$$\text{Minimize} \quad f(\vec{x})$$

$$\text{Subject to:} \quad g_j(\vec{x}) \le 0, \qquad j = 1, \cdots, l$$

$$h_j(\vec{x}) = 0, \qquad j = l+1, \cdots, m$$

where $\vec{x} = (x_1, \cdots, x_n)$ is the decision variable which is bounded by the decision space S. S is defined by the constraints:

$$L_i \le x_i \le U_i, \qquad 1 \le i \le n \tag{1}$$

Y. Tan et al. (Eds.): ICSI-CCI 2015, Part II, LNCS 9141, pp. 261–272, 2015.
DOI: 10.1007/978-3-319-20472-7_28

Here, l is the number of inequality constraints and m-l is the number of equality constraints.

The Evolutionary Algorithms (EAs), as the unconstrained search techniques and solution generating strategies, are not suitable enough to solve COPs without additional mechanisms to deal with the constraints. Consequently, many constrained optimization evolutionary algorithms (COEAs) are proposed [1]-[4].

Penalty function methods are the most widely used methods for handling constraints, in which some penalty parameters are adopted to balance the objective function values and constraint violation (i.e., to bias the search in the constrained search space [5]). It's clear that the performance of these methods is mainly determined by their parameters and the methods can be classified based on the form of these parameters.

If the penalty parameters keep constant throughout the evolution process, this method is called static penalty function method. Alternatively, if the penalty parameters are related with the current generation number, it is called dynamic penalty function method. As many parameters are required in dynamic penalty function method, some adaptive penalty functions which gather information from the search process, or self-adaptive approaches, which evolve both the penalty parameters and solutions, have also been proposed [6].

Some other approaches based on careful comparison among feasible and infeasible solutions are also developed.

For example, Deb [7] proposed a feasibility-based rule to pair-wise compare individuals:

1) Any feasible solution is preferred to any infeasible solution.

2) Among two feasible solutions, the one having better objective function value is preferred.

3) Among two infeasible solutions, the one having smaller constraint violation is preferred.

The stochastic ranking method (SR) proposed by Runarsson and Yao [8] is a very classical constraint handling technique, which tries to achieve a balance between objective function value and constraint violation stochastically. It compares pair-wise solutions using the following criteria: 1) if both individuals are feasible, the ranking of them is determined by the objective function value; else 2) the parameter P_f will determine the probability of ranking by objective function value or constraint violation. Deb's feasibility-based rule can be seen as a special case of SR with P_f =0. The experimental results indicate that an overall better performance can be obtained when P_f =0.45. However, this paper didn't provide the assurance that P_f =0.45 is an optimal value.

And later, these two authors also pointed out that there should be some biases when solving single-objective optimization problems (SCOPs) [9].

Besides, some methods based on multi-objective optimization concepts are also presented. The main idea is to convert the single-objective constrained optimization problem into a bi-objective or multi-objective optimization problem taking the constraints as one or more objectives to be minimized.

After reviewing the three most frequently used constraint handling techniques, some approaches based on the "dynamic or adaptive" idea that are closely related with the work presented in this paper will be introduced in detail.

Wang *et al.* [10] proposed an adaptive tradeoff model (ATM) for constrained evolutionary optimization. In this model, to obtain an appropriate tradeoff between objective function and constraint violation, different tradeoff schemes during different situations in a search process (i.e., infeasible situation, semi-feasible situation and feasible situation) are designed. Based on this idea, an improved adaptive tradeoff model was proposed, in which each constraint violation is first normalized [11], and this model was combined with (μ+λ)-DE with the name (μ+λ)-CDE. To overcome the drawback of dynamic settings for tolerance value δ, Jia *et al.* [12] presented an improved version of (μ+λ)-CDE, named ICDE. Unlike (μ+λ)-CDE, in ICDE, the hierarchical non-dominated individual selection scheme is utilized in the infeasible situation and the feasibility proportion of the population is used to convert the objective function of each individual in the semi-feasible situation.

Besides, some other adaptive approaches or frameworks were also introduced [13]-[16]. As the solution's property (i.e., infeasible or feasible) plays an important role in solving COPs, some adaptive methods based on this were also presented.

Farmani and Wright [17] proposed a self-adaptive fitness formulation in which the infeasibility measure is used to form a two-stage penalty to the infeasible solutions. Venkatraman and Yen [18] presented a generic, two phase framework with the aim to find a feasible solution in the first phase. Based on Yao's stochastic ranking [8], Zhang *et al.* [19] proposed a dynamic stochastic selection (DSS) within the framework of multimember DE (DSS_MDE). Tessema *et al.* [20] introduced another adaptive penalty formulation which uses the number of feasible individuals to determine the amount of penalty added to infeasible individuals (i.e., to guide the search process toward finding more feasible individuals *et al.*).

Many of these methods mentioned above can get a relatively satisfying result, but the parameters used in these approaches are mainly determined by the experiments. The inner mechanism of Constraint Handling Techniques (e.g., the relationship of different CHTs, when and why some CHTs are more efficient) is few studied.

Herein, to overcome this drawback, in this paper we first studied the penalty parameter systematically, and then proposed a new dynamic penalty function (DyPF) based on the analysis.

The rest of this paper is organized as follows. Section 2 systematically analyzes the penalty parameter. Based on this, Section 3 presents a detailed description of the proposed DyPF. The experimental results and the comparison with some similar state-of-the-art methods are given in Section 4. Finally, Section 5 concludes with a brief summary of this paper and some future work.

2 Systematical Analysis of Penalty Parameters

The basic idea of the discussion is that by introducing four rules (i.e., A_1, A_2, B_1, B_2), if the penalty parameter is consistent with some rule's combination in certain

range, then we can conclude that the penalty parameter value in this range has no effect on ranking the solutions [21].

As there are only two kinds of solutions in the evolutionary process (i.e., feasible solutions and infeasible solutions), the ranking or the selection process is mainly between these two kinds of solutions, including feasible solutions versus feasible solutions, feasible solutions versus infeasible solutions, and infeasible solutions versus infeasible solutions. As there is no constraint violation for feasible solutions, only the objective function values are used for the comparison between feasible solutions and feasible solutions. As to the comparison involving infeasible solutions, the following rules are introduced:

A_1: a feasible solution is superior to an infeasible solution.

A_2: a feasible solution is inferior to an infeasible solution.

B_1: two infeasible solutions will be ranked according to the constraint violation, and the one with less constraint violation will be preferred.

B_2: two infeasible solutions will be ranked according to the reciprocal of the constraint violation, and the one with more constraint violation will be preferred.

Additional rule: among two infeasible solutions with the same constraint violation, the one with less objective function value is preferred (for the minimization problems).

It should be noted that here the characteristics of the rules (i.e., good or bad) are not considered as the judgment of a rule is closely related with the problem characteristics (e.g., the location of the optimal solution, the topological properties of the constraints, etc.) and the solving goals (e.g., how to judge the solution's performance if no feasible solutions are found, etc.). And here, the main concern is that the ranking result following some rules' combination (A_1-B_1, A_1-B_2, A_2-B_1, A_2-B_2) is unique.

The introduction of penalty parameter enables us to transform a constrained optimization problem (*A*) into an unconstrained one (*A'*) [8]. Here, we define the evaluation function *L* as follows.

For the given λ , δ >0, let

$$L(\vec{x}_i, \lambda, \delta) = f(\vec{x}_i) + \lambda G(\vec{x}_i, \delta) \quad i = 1, 2, ..., NP \tag{2}$$

where $\vec{x}_i$ stands for the *NP* *n*-dimensional real-valued vectors of the population. λ is the penalty parameter and δ is the tolerance value for the equality constraints. f is the objective function and G is the penalty function with the form as follows.

$$G(\vec{x}_i, \delta) = \sum_{j=1}^{m} G_j(\vec{x}_i, \delta) = \sum_{j=1}^{l} \max(0, g_j(\vec{x}_i)) + \sum_{j=l+1}^{m} \max(0, \left| h_j(\vec{x}_i) \right| - \delta) \tag{3}$$

As the effect of λ is mainly concerned, δ can be supposed as a constant. The formula (2) can be transformed as

$$L(\vec{x}_i, \lambda) = f(\vec{x}_i) + \lambda G(\vec{x}_i) \quad i = 1, 2, ..., NP \tag{4}$$

Given two population members, $\vec{x}_s$ and $\vec{x}_t$, where s and t are randomly selected from [1, NP] and satisfying: $s \neq t$, the difference between their evaluation function values is:

$$\begin{aligned}\Delta(\vec{x}_s, \vec{x}_t, \lambda) &= L(\vec{x}_s, \lambda) - L(\vec{x}_t, \lambda) \\ &= [f(\vec{x}_s) + \lambda G(\vec{x}_s)] - [f(\vec{x}_t) + \lambda G(\vec{x}_t)] \\ &= [f(\vec{x}_s) - f(\vec{x}_t)] + \lambda [G(\vec{x}_s) - G(\vec{x}_t)] \end{aligned} \quad (5)$$

We define $\Delta f_{st} = f(\vec{x}_s) - f(\vec{x}_t)$, $\Delta G_{st} = G(\vec{x}_s) - G(\vec{x}_t)$, then formula (5) can be written as:

$$\Delta(\vec{x}_s, \vec{x}_t, \lambda) = \Delta f_{st} + \lambda \cdot \Delta G_{st} \quad (6)$$

Suppose the number of feasible and infeasible individuals in the population is p and q respectively, with $0 \le p \le NP, 0 \le q \le NP \ \& \ p + q = NP$.

1) Infeasible solution versus feasible solution

Similarly, given two population members, $\vec{x}_s$ and $\vec{x}_t$, where s and t are randomly selected from the p feasible solutions and q infeasible solutions, the formula (5) can be transformed as:

$$\begin{aligned}\Delta(\vec{x}_s, \vec{x}_t, \lambda) &= L(\vec{x}_s, \lambda) - L(\vec{x}_t, \lambda) \\ &= f(\vec{x}_s) - [f(\vec{x}_t) + \lambda G(\vec{x}_t)] \\ &= [f(\vec{x}_s) - f(\vec{x}_t)] - \lambda G(\vec{x}_t) \end{aligned} \quad (7)$$

According to rule A_1, member $\vec{x}_s$ is better than member $\vec{x}_t$. From formula (7), if $\lambda > \dfrac{\Delta f_{st}}{G(\vec{x}_t)}$, then $\Delta(\vec{x}_s, \vec{x}_t, \lambda) < 0$. In this case, rule A_1 and penalty function method have the same effect on ranking these two solutions.

2) *Infeasible solution versus infeasible solution*

According to rule B_1, two infeasible solutions will be ranked according to the constraint violation. So if $\Delta(\vec{x}_s, \vec{x}_t, \lambda)$ and ΔG_{st} have the same symbol (i.e., positive or negative), we can conclude that rule B_1 and penalty function method have the same effect on ranking these two individuals.

There are three different cases in this situation:

a) $\Delta G_{st} > 0$: In this case, if $\lambda > -\dfrac{\Delta f_{st}}{\Delta G_{st}}$, $\Delta(\vec{x}_s, \vec{x}_t, \lambda) = \Delta f_{st} + \lambda \cdot \Delta G_{st} > 0$.

b) $\Delta G_{st} < 0$: In this case, if $\lambda > -\dfrac{\Delta f_{st}}{\Delta G_{st}}$, $\Delta(\vec{x}_s, \vec{x}_t, \lambda) = \Delta f_{st} + \lambda \cdot \Delta G_{st} < 0$.

c) $\Delta G_{st} = 0$: This is the case when the two individuals have the same constraint violation, and $\Delta(\vec{x}_s, \vec{x}_t, \lambda) = \Delta f_{st}$, which is not related with λ. In this case, the ranking will follow additional rule.

In general, when ranking two individuals (e.g., $\vec{x}_s$ and $\vec{x}_t$), if $\lambda > -\frac{\Delta f_{st}}{\Delta G_{st}}, \Delta G_{st} \neq 0$ (here, s and t are randomly selected from the q infeasible solutions), then rule B_1 and penalty function method have the same effect.

Denote I_{fea} and $I_{\inf}$ as the set of the index of p feasible solutions and q infeasible solutions respectively.

Suppose

$$S_{\inf} = \left\{ \begin{array}{l} \lambda_{ij} \mid \lambda_{ij} = -\frac{\Delta f_{ij}}{\Delta G_{ij}}, i = I_{\inf}(1),...,I_{\inf}(q);... \\ j = I_{\inf}(1),...,I_{\inf}(q) \,\&\, G(i) \neq G(j) \end{array} \right\} \; S_{sem} = \left\{ \begin{array}{l} \lambda_{ij} \mid \lambda_{ij} = \frac{\Delta f_{ij}}{G(\vec{x}_j)}, i = I_{fea}(1),...,I_{fea}(p); \\ \qquad\qquad j = I_{\inf}(1),...,I_{\inf}(q) \end{array} \right\}$$

The set of λ can be described as $S = \{S_{\inf}, S_{sem}\}$. As to $S_{\inf}$, we define $\lambda_{\inf}^{\max} = \max(\lambda_{ij})$, $\lambda_{\inf}^{\min} = \min(\lambda_{ij})$, where $i = I_{\inf}(1),...,I_{\inf}(q)$; $j = I_{\inf}(1),...,I_{\inf}(q)$. Likewise, for S_{sem}, we define $\lambda_{sem}^{\max} = \max(\lambda_{ij})$, $\lambda_{sem}^{\min} = \min(\lambda_{ij})$, where $i = I_{fea}(1),...,I_{fea}(p)$; $j = I_{\inf}(1),...,I_{\inf}(q)$. Then the largest value of λ in S is $\lambda_{\max} = \max(\lambda_{\inf}^{\max}, \lambda_{sem}^{\max})$.

There are some different cases in this situation:

a) $\lambda > \lambda_{\max}$: In this case, the ranking results is the same as rule A_1-B_1.

b) $\lambda_{sem}^{\max} < \lambda \leq \lambda_{\max}$: In this case, the results ranked by penalty function method can satisfy rule A_1, but the comparison among infeasible solutions may not satisfy B_1 or B_2.

c) $\lambda_{sem}^{\min} \leq \lambda \leq \lambda_{sem}^{\max}$: In this case, the ranking will not satisfy A_1 or A_2 fully.

d) $\lambda < \lambda_{sem}^{\min}$: the ranking result can satisfy A_2 (i.e., infeasible solutions are better than feasible solutions).

Similarly, following rule B (B_1 and B_2),

e) $\lambda_{\inf}^{\max} < \lambda \leq \lambda_{\max}$: In this case, the results ranked by penalty function method can satisfy rule B_1.

f) $\lambda_{\inf}^{\min} \leq \lambda \leq \lambda_{\inf}^{\max}$: In this case, the ranking will not satisfy B_1 or B_2 fully.

g) $\lambda < \lambda_{\inf}^{\min}$: the ranking result can satisfy B_2 (i.e., infeasible solutions are ranked according to the reciprocal of the constraint violation).

h) $\lambda < \lambda_{\min}$: the ranking result can satisfy A_2-B_2 (i.e., all infeasible solutions are better than feasible solutions, and the infeasible solutions are ranked according to the reciprocal of the constraint violation).

The general results are illustrated in Fig.1, which can provide a limit but effective basis for analyzing and designing adaptive penalty function methods.

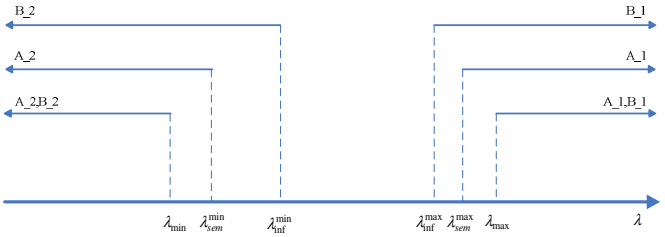

Fig. 1. The corresponding rule for penalty parameter λ

3 A Dynamic Penalty Function

3.1 Basic Idea

As mentioned in Section 2, a too large or too small penalty parameter value will have no influence on ranking the population, and the effect of penalty parameter is highly related with the solutions in the current generation. Based on this, a new DyPF considering the infeasible and semi-feasible situations is proposed.

If the solutions are experiencing the infeasible situation, as one of the main aims is to quickly find a feasible solution, the solutions will be ranked according to the degree of constraint violation, which is consistent with the rule B_1. This strategy is equivalent to the penalty function method when $\lambda > \lambda_{\max}$. And here, λ is set as follows:

$$\lambda = \frac{\max f - \min f}{\min(\Delta G_{ij})} \tag{8}$$

where $i = 1, 2, \ldots, NP; j = 1, 2, \ldots, NP$. Here, f is the objective function value, G is the constraint violation and $\Delta G_{ij} = G(i) - G(j) \,\&\, G(i) \neq G(j)$.

Similarly, if the solutions are experiencing the semi-feasible situation, as there are both feasible and infeasible solutions in the population, apart from considering the comparison between feasible and infeasible solutions, the proportion of feasible solutions should also be taken into account when setting the penalty parameters. And consequently a parameter r_f is introduced. λ is set as follows:

$$\begin{aligned} &\textit{if } r_f < \mu \\ &\quad \lambda = (1+\alpha) \cdot \max\{\lambda_{\inf}^{\max}, \lambda_{sem}^{\max}\}; \\ &\textit{else} \\ &\quad \lambda = (1-r_f) \cdot \lambda_{sem}^{\max} + r_f \cdot \lambda_{sem}^{\min}; \\ &\textit{end} \end{aligned} \tag{9}$$

Here, μ is a threshold, determining using Deb's feasibility-based rule or penalty function method, which can be fixed or a rand number. α is a positive number, and here it's set as 0.01. The value of $\lambda_{\inf}^{\max}$, $\lambda_{sem}^{\min}$, $\lambda_{sem}^{\max}$ are set as Section 2.

From (9), it can be observed that the chance for an infeasible solution to survive into next generation is changing according to the proportion of feasible solutions in last generation. When $r_f < \mu$, the ranking will be based on Deb's feasibility-based rule;

when $r_f \geq \mu$, the ranking will be determined by r_f, $\lambda_{sem}^{\min}$, $\lambda_{sem}^{\max}$, and $\lambda_{sem}^{\min} < \lambda < \lambda_{sem}^{\max}$. In this situation, when r_f is relatively small, the ranking will mainly rely on the constraint violation to select more feasible solutions, as the value of λ is near $\lambda_{sem}^{\max}$; when r_f is relatively large, the ranking will mainly rely on the objective function value (e.g., the feasible solutions are not necessarily superior than the infeasible solutions), and in this case, the chance for infeasible solutions to survive into the next generation will increase.

3.2 Realization

Algorithm 1: Framework of DyPF

Input: *NP*: the size of population at each generation
Max_FES: maximum number of function evaluations
Output: $\vec{x}_{best}$: the best solution in the final population
Step 1 Initialization
Step 1.1 t=0;
Step 1.2 Generate an initial population $P_0 = \{\vec{x}_{1,0}, \cdots, \vec{x}_{NP,0}\}$ randomly.
Step 1.3 Evaluate the objective function values $f(\vec{x}_{i,0})$ and the degree of constraint violations $G(\vec{x}_{i,0})$.
Step 1.4 *FES*=*NP*.
Step 2 Dynamic penalty function model
Step 2.1 Update P_t using DE model to create offspring. These *NP* offspring form the offspring population Q_t.
Step 2.2 Evaluate $f(\vec{x}_{i,t})$ and $G(\vec{x}_{i,t})$ $(i = 1, \cdots, NP)$.
Step 2.3 Compute the feasibility percent r_f of the combined population H_t (i.e., $H_t = P_t \bigcup Q_t$).
Step 2.4 Determinate the current situation of H_t according to r_f.
Step 2.5 Calculate the value of λ according to different situations.
Step 2.6 Compute the fitness function value with λ in **Step 2.5**.
Step 2.7 Rank the population through the value of fitness function and select the best *NP* individuals to constitute the next population P_{t+1}.
Step 2.8 *FES*=*FES*+*NP*.
Step 3 Set t=t+1.
Step 4 Stopping Criterion: If *FES*≥ *Max_FES*, stop and output the best solution $\vec{x}_{best}$, otherwise go to **Step2**

The framework of DyPF is illustrated in Algorithm 1.

4 Experimental Study

4.1 Experimental Settings

This experiment is to verify the effectiveness of proposed DyPF. The details of these benchmark functions are reported in [23]. The evolutionary algorithm used in this paper is *DE/rand/1/bin* [22], and the boundary constraint is reset as [7].The parameters in DE are set as follows: the population size (*NP*) is set to 100; the scaling factor (*F*) is randomly chosen between 0.5 and 0.6, and the crossover control parameter (*Cr*) is randomly chosen between 0.9 and 0.95.

4.2 Experimental Results

25 independent runs were performed for each test function using 5×10^5 FES at maximum, as suggested by Liang *et al.* [23]. Additionally, the tolerance value δ for the equality constraints was set to 0.0001.

1) *The impact of parameter* μ *:* In this part, the impact of μ on the results generated by DyPF is evaluated. Six different values of μ are adopted, i.e., μ=0.1, 0.2, 0.3, 0.4, 0.5 and rand. For page limited, the detailed results are not listed here. The best overall performance can be obtained when $\mu = 0.5$.

2) *Comparison with some "dynamic" approaches in constrained evolutionary optimization:* In this part, we compare DyPF with five other state-of-the-art approaches using the concept of "dynamic" or "adaptive": SR [8]; SMES [5]; ATMES [10]; TPGA [18] and SaFF [17]. The experimental results of these five approaches are directly taken from the references and the comparative results are presented in Table 1.

To statistically compare the performance of different approaches, *t*-test results (h values) are presented in Table 1. Numerical values -1, 0, 1 represent that DyPF is inferior to, equal to and superior to other approaches respectively. It should be noted that the h values is determined considering the overall results, but for page limited, we just list the best value in Table 1.

For all the performance metrics, DyPF performs better than the other five approaches in g02, g05, g07, g09, and g10 as shown in Table 1.

All these six approaches have the same or similar performance in g08 and g12. As for g01 and g11, all approaches except TPGA can always reach the optimal value.

It should be pointed out that DyPF performs not so well on g03 and g13, though the other five approaches also can't obtain a satisfying result.

Besides, from the *t*-test results, it can be seen that DyPF is superior to, equal to and worse than other methods in 37, 21 and 7 cases, respectively out of the 65 cases. The worse cases are mainly from g03. Therefore, the overall performance of DyPF is highly competitive with the other five "dynamic" approaches, especially when considering that DyPF is simple and easy to realize.

Table 1. Comparison of DyPF with "Dynamic" approaches

Fun. & Optimal value		SR [8]	SMES [5]	ATMES [10]	TPGA [18]	SaFF [17]	DyPF
G01 -15.0000	best	-15.000	-15.000	-15.000	-14.9999	-15.0000	-15.0000
	h	0	0	0	1	0	
G02 -0.803619	best	-0.803515	-0.803601	-0.803388	-0.803190	-0.802970	-0.803619
	h	1	1	1	1	1	
G03 -1.0005	best	-1.000	-1.000	-1.000	-1.00009	-1.00000	-0.7412
	h	-1	-1	-1	-1	-1	
G04 30665.5387	best	30665.539	30665.539	30665.539	30665.5312	-30665.50	30665.5387
	h	0	0	0	1	1	
G05 5126.4967	best	5126.497	5126.599	5126.498	5126.5096	5126.9890	5126.4967
	h	1	1	1	1	1	
G06 -6961.8139	best	-6961.814	-6961.814	-6961.814	-6961.1785	-6961.800	-6961.8139
	h	1	1	0	1	1	
G07 24.3062	best	24.307	24.327	24.306	24.410977	24.48	24.3062
	h	1	1	0	1	1	
G08 0.09582504	best	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	0.09582504
	h	0	0	0	0	0	
G09 680.630057	best	680.630	680.632	680.630	680.762228	680.64	680.630057
	h	1	1	1	1	1	
G10 7049.2480	best	7054.316	7051.903	7052.253	7060.55288	7061.34	7049.2480
	h	1	1	1	1	1	
G11 0.7499	best	0.750	0.75	0.75	0.7490	0.7500	0.7499
	h	0	0	0	1	0	
G12 -1.0000	best	-1.000000	-1.000	-1.000	NA	-1	-1.0000
	h	0	0	1	1	0	
G13 0.05394151	best	0.053957	0.053986	0.053950	NA	NA	0.05562855
	h	-1	0	-1	1	1	

5 Conclusion

In this paper, a new Dynamic Penalty Function (DyPF) has been proposed for constrained evolutionary optimization, which is based on the systematical analysis of penalty parameter. Thus this enables DyPF to take advantage of different strategies by adjusting penalty parameters at different situations. To verify the effectiveness of the newly proposed DyPF, an experiment is carried out which is based on the 21 benchmark functions collected in the IEEE CEC2006 special session on constraint real-parameter optimization.

The results show that DyPF has a high effectiveness and is very competitive comparing with other five dynamic or adaptive state-of-the-art methods referred to in this paper in view of simplicity of DyPF. Nevertheless, DyPF can not find a successful solution in g13, which is due to the simplicity of the model.

DE and other evolutionary algorithms have shown a good performance on unconstrained problems, which means they are good at generating satisfying solutions. Thus key point in solving constraint problems is how to select or rank the solutions, especially how to keep the balance between objective function and constraint violations, which is also related with the problem's topological properties. Therefore, this will be our future work.mn.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China under Grants 71371142, 61174183. Chengyong Si would like to thank Prof. Dr. Robert Weigel for his great help in the life and research work, as this work is partially done when he was with the Institute for Electronics Engineering, University of Erlangen-Nuernberg in Germany as a joint doctor, and he is grateful to Dr. Y. Wang for the valuable suggestions. The authors also gratefully acknowledge Dr. T. Lan, Dr. J. Hu, and Dr. G. Gao for improving the presentation of this paper.

References

1. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithm for constrained parameter optimization problems. Evol. Comput. **4**(1), 1–32 (1996)
2. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Comput. Methods Appl. Mech. Eng. **191**(11/12), 1245–1287 (2002)
3. Cai, Z., Wang, Y.: A multiobjective optimization-based evolutionary algorithm for constrained optimization. IEEE Trans. Evol. Comput. **10**(6), 658–675 (2006)
4. Wang, Y., Cai, Z.: A dynamic hybrid framework for constrained evolutionary optimization. IEEE Trans. Syst. Man Cybern. B Cybern. **42**(1), 203–217 (2012)
5. Mezura-Montes, E., Coello Coello, C.A.: A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Trans. Evol. Comput. **9**(1), 1–17 (2005)
6. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. **2**(2), 124–141 (1999)
7. Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Eng. **186**(2–4), 311–338 (2000)
8. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Trans. Evol. Comput. **4**(3), 284–294 (2000)
9. Runarsson, T.P., Yao, X.: Search bias in constrained evolutionary optimization. IEEE Trans. Syst. Man Cybern. C Appl. Rev. **35**(2), 233–243 (2005)
10. Wang, Y., Cai, Z., Zhou, Y., Zeng, W.: An adaptive tradeoff model for constrained evolutionary optimization. IEEE Trans. Evol. Comput. **12**(1), 80–92 (2008)
11. Wang, Y., Cai, Z.: Constrained evolutionary optimization by means of (μ+λ)-differential evolution and improved adaptive trade-off model. Evol. Comput. **19**(2), 249–285 (2011)
12. Jia, G., Wang, Y., Cai, Z., Jin, Y.: An improved (μ+λ)-constrained differential evolution for constrained optimization. Inform. Sci. **222**, 302–322 (2013)
13. Zhan, Z., Zhang, J., Li, Y., Chung, H.S.: Adaptive particle swarm optimization. IEEE Trans. Syst. Man Cybern. B Cybern. **39**(6), 1362–1381 (2009)

14. Gong, W., Cai, Z., Ling, C.X., Li, H.: Enhanced differential evolution with adaptive strategies for numerical optimization. IEEE Trans. Syst. Man Cybern. B Cybern. **41**(2), 397–413 (2011)
15. Minhazul, I.S., Das, S., Ghosh, S., Roy, S., Suganthan, P.N.: An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Trans. Syst. Man Cybern. B Cybern. **42**(2), 482–500 (2012)
16. Zhao, S.Z., Suganthan, P.N., Das, S.: Self-adaptive differential evolution with multi-trajectory search for large scale optimization. Soft Comput. **15**(11), 2175–2185 (2011)
17. Farmani, R., Wright, J.A.: Self-adaptive fitness formulation for constrained optimization. IEEE Trans. Evol. Comput. **7**(5), 445–455 (2003)
18. Venkatraman, S., Yen, G.G.: A generic framework for constrained optimization using genetic algorithms. IEEE Trans. Evol. Comput. **9**(4), 424–435 (2005)
19. Zhang, M., Luo, W., Wang, X.: Differential evolution with dynamic stochasitc selection for constrained optimization. Inform. Sci. **178**(15), 3043–3074 (2008)
20. Tessema, B., Yen, G.G.: An adaptive penalty formulation for constrained evolutionary optimization. IEEE Trans. Syst., Man, Cybern. A Syst. Hum. **39**(3), 565–578 (2009)
21. Si, C., Lan, T., Hu, J., Wang, L., Wu, Q.: Penalty parameter of the penalty function method. Control Decis. **29**(9), 1707–1710 (2014)
22. Das, S., Suganthan, P.N.: Differential evolution: A Survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011)
23. Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello Coello, C.A., Deb, K.: Problem definitions and evaluation criteria for the CEC 2006. Technical report, Special Session on Constrained Real-Parameter Optimization (2006)