J Comb Optim (2006) 12:409–419
DOI 10.1007/s10878-006-9004-x

Dynamic-objective particle swarm optimization for constrained optimization problems

Haiyan Lu · Weiqi Chen

Published online: 20 September 2006

Abstract This paper firstly presents a novel constraint-handling technique , called dynamic-objective method (DOM), based on the search mechanism of the particles of particle swarm optimization (PSO). DOM converts the constrained optimization problem into a bi-objective optimization problem, and then enables each particle to dynamically adjust its objectives according to its current position in the search space. Neither Pareto ranking nor user-defined parameters are involved in DOM. Secondly, a new PSO-based algorithm—restricted velocity PSO (RVPSO)—is proposed to specialize in solving constrained optimization problems. The performances of DOM and RVPSO are evaluated on 13 well-known benchmark functions, and comparisons with some other PSO algorithms are carried out. Experimental results show that DOM is remarkably efficient and effective, and RVPSO enhanced with DOM exhibits greater performance. In addition, besides the commonly used measures, we use histogram of the test results to evaluate the performance of the algorithms.

Keywords Constrained optimization · Particle swarm optimization · Constraint-handling · Evolutionary computation

1 Introduction

This paper considers the constrained Optimization problems (COPs) formulated as

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

This research is supported by National Natural Science Foundation of China (No. 10371028) and Scientific Research Fund of Southern Yangtze University (No. 0003182).

H. Lu (✉)
Department of Mathematics, Zhejiang University, Hangzhou 310027, P. R. China;
Department of Information & Computing Science, Southern Yangtze University, Wuxi 214122, P. R. China
e-mail: kangting88@hotmail.com; e-mail: luhaiyan_xingyu@zju.edu.cn

W. Chen
Department of Computer Science & Engineering, Southern Yangtze University, Wuxi 214122,
P. R. China; The Sixth Department, China Ship Scientific Research Center, Wuxi 214082, P. R. China

where $f(\vec{x})$ is the objective function, $\vec{x} \in \mathcal{S} \cap \mathcal{F}, \mathcal{S} \subseteq \mathcal{F}, \mathcal{S} \subseteq \mathcal{R}^n$ defines the *search space* by the parametric constraints $l_d \leq x_d \leq u_d$ and the *feasible region* $\mathcal{F}$ is defined by

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, q \tag{2}$$

$$h_j(\vec{x}) = 0, \quad j = q+1, \ldots, m \tag{3}$$

Usually equality constraints are transformed into inequalities of the form

$$|h_j(\vec{x})| - \epsilon \leq 0, \quad \text{for } j = q+1, \ldots, m \tag{4}$$

A solution $\vec{x}$ is regarded as *feasible* if $g_i(\vec{x}) \leq 0$, for $i = 1, \ldots, q$ and $|h_j(\vec{x})| - \epsilon \leq 0$, for $j = q+1, \ldots, m$. In this paper ϵ is set to 0.001.

Particle swarm optimization (PSO) is one of the population-based evolutionary algorithms (EAs) that use ideas and get inspirations from the social behavior of a flock of birds or fish schooling. Since its introduction by Kennedy and Eberhart (1995), PSO has gained increasing popularity due to its effectiveness in performing a wide variety of difficult optimization problems (Coello, 2004; Parsopoulos and Vrahatis, 2004; Shi and Krohling, 2002; Clerc and Kennedy, 2002). However, like other EAs, PSO in its original form lacks a mechanism for handling constraints to the problems. Although a number of constraint-handling methods have been proposed for other EAs (Coello, 2002; Michalewicz and Schoenauer, 1996), there has been little work related to the incorporation of constraints into PSO (Hu and Eberhart, 2002; Coath and Halgamuge, 2003; Hu and Eberhart, 2003; Toscano and Coello, 2004).

Motivated by the above fact, in this paper we propose a new constraint-handling method, called dynamic-objective method (DOM), based on the search mechanism of PSO. In DOM, we transform the original single-objective constrained optimization problem into a bi-objective unconstrained optimization problem, and then a dynamic-objective strategy is applied to bias the search towards a feasible solution.

Moreover, we believe that incorporating a constraint-handing mechanism into an EA is only a remedy in solving COPs, the inherent search mechanism of EAs should also be modified to reflect the impact of feasible region on the search mechanism of the algorithm. Therefore, we propose a novel PSO, called restricted velocity PSO (RVPSO), which utilizes the information about the feasible region that the particle learned from its fly experience during the search process.

The rest of this paper is organized as follows. Section 2 briefly introduces the PSO algorithm. Section 3 presents our main contribution, including DOM and RVPSO. The experimental results are reported in Section 4. The DOM is presented in Section 3. Finally, we establish some conclusions and put forward our future paths of research in Section 5.

2 Particle swarm optimization (PSO)

PSO employs a population of individuals to probe the promising region of the search space. In this paradigm, the population is called a *swarm* and the individuals are called *particles*. Each particle i ($\vec{x}_i$) is flown with a velocity ($\vec{v}_i$) through the search space, and retains in its memory the best position it ever experienced ($\vec{p}_i$). In the *global* version of PSO, each particle knows the best position ever attained by all particles of the swarm ($\vec{p}_g$). In the *local* version, each particle has a knowledge of the best position obtained within its own neighborhood.

The update equations of the canonical PSO (CPSO) is as follows:

$$\vec{v}_i^{k+1} = \omega\vec{v}_i^k + c_1 r_1\big(\vec{p}_i^k - \vec{x}_i^k\big) + c_2 r_2\big(\vec{p}_g^k - \vec{x}_i^k\big) \tag{5}$$

$$\vec{x}_i^{k+1} = \vec{x}_i^k + \vec{v}_i^{k+1} \tag{6}$$

where $i = 1, 2, \ldots, N$, k is the iteration counter, c_1 and c_2 are *cognitive* and *social* parameters, respectively; r_1 and r_2 are random numbers uniformly distributed within the range $[0, 1]$, i.e., $r_1,\ r_2 \in U[0, 1]$; ω is the *inertia weight*, and is set to $0.5 + \frac{\text{rand}(\cdot)}{2}$ where $\text{rand}(\cdot) \in U[0, 1]$. c_1 and c_2 are both set to 1.49445. The above parameter settings (Hu et al., 2003) are used for comparison with our proposed algorithm in this paper.

3 Our approaches

3.1 Dynamic-objective method (DOM) for constraint-handling

The idea underlying our proposed constraint-handling method is very natural and simple. When applying PSO to a COP, each particle moves towards a promising region within the search space. It is reasonable to conceive that, only after a particle enters the feasible region, could it have the opportunity to approach the optimal solution located in the feasible region. In other words, for particle outside the feasible region, their objective is to enter the feasible, rather than searching for the optimal solution; while for those already inside the feasible region, their objective is to find the optimal solution. Therefore, the original COP can be viewed as a bi-objective unconstrained optimization problem, one objective (the first objective) is to enter the feasible region, and the other one (the second objective) is to optimize the original objective function of the COP. During the search process of PSO, the search mechanism is devised such that each particle have the ability to dynamically adjust its objectives according to whether it resides inside or outside the feasible region. This is why we refer to this constraint-handling mechanism as dynamic-objective method (DOM).

As the aforementioned, in DOM the first objective is essentially to optimize a function whose optimal solutions are exactly all the feasible solutions of the original COP. There are several ways to construct such kind of function. One simple form of this first objective is as follows:

$$\Phi(\vec{x}) = \sum_{i=1}^{q} \max(0, g_i(\vec{x})) + \sum_{j=q+1}^{m} \max(0, |h_j(\vec{x})| - \epsilon) \tag{7}$$

Clearly, $\Phi(\vec{x})$ is the sum of constraint violations, $\Phi(\vec{x}) \geq 0$, and thus $\Phi(\vec{x}) = 0$ for $\forall\vec{x} \in \mathcal{F}$. Moreover, all the optimal solutions of $\Phi(\vec{x})$ constitute the feasible region $\mathcal{F}$ of the original problem. In DOM, $\Phi(\vec{x})$ is seemingly similar to but essentially different from penalty function, it is just used as a "distance" measure of each particle apart from the feasible region. Consequently, the original COP is converted into the following unconstrained bi-objective optimization problem:

$$\min F(\vec{x}) = (\Phi(\vec{x}), f(\vec{x})), \quad \vec{x} = (x_1, \ldots, x_n) \subset \mathcal{S} \in \mathcal{R}^n \tag{8}$$

Theoretically, only when $\Phi(\vec{x}) = 0$ could the particle begin to minimize $f(\vec{x})$, since the optimum value of $\Phi(\vec{x})$ is 0. In practice, however, we can start optimizing $f(\vec{x})$ in advance by prescribing a threshold $\delta \geq 0$ such that if $\Phi(\vec{x}) \leq \delta$ then the particle begin to optimize $f(\vec{x})$.

Here δ serves as a switch which controls when to start the process of optimizing $f(\vec{x})$. The value (large or small) of δ will not prevent the particle from entering the feasible region. It is evident that when the particle lies outside the feasible region, $\Phi(\vec{x}) > 0$, the particle does not need to calculate $f(\vec{x})$ at all, resulting in a decreasing in the computational cost.

It should be noted that after the particle starts optimizing $f(\vec{x})$, it is still likely for this particle to fly out of the feasible region. In this situation, the particle would give up optimizing $f(\vec{x})$ and turn again to minimize $\Phi(\vec{x})$. Therefore, each particle has the ability to dynamically adjust its objectives according to its current position in the search space. Moreover, the selection of objectives and the optimization process of each particle are independent and parallel. In addition, DOM do not need to operate Pareto ranking or to compare the quality of particles between one another, this is what our proposed DOM differs from ordinary multi-objective method for constraint-handling (Coello, 2000).

The updating of $\vec{p}_i$ and $\vec{p}_g$ in (5) is a crucial step in a PSO algorithm. From the aforementioned search mechanism of particles in DOM, when incorporating DOM into a PSO algorithm, we can readily obtain the update strategies of $\vec{p}_i$ and $\vec{p}_g$. That is, if the current position of a particle is within the feasible region, then $\vec{p}_i$ is defined as **the fittest feasible solution (with respect to $f(\vec{x})$) it has encountered so far; otherwise, $\vec{p}_i$ is defined as** its encountered position that is either closest to the feasible region or the fittest feasible solution (if any feasible solution has been found). As for $\vec{p}_g$, if all the positions ever encountered by the swarm lie outside the feasible region, then $\vec{p}_g$ is taken as the position closest to the feasible region; otherwise, $\vec{p}_g$ is defined as the fittest feasible solution ever encountered by the swarm. The pseudo-code of DOM on how to update $\vec{p}_i$ and $\vec{p}_g$ is described in Fig. 1.

DOM is a generic constraint-handling method suitable for incorporating into many variants of PSO algorithms. The effectiveness and efficiency of DOM will be verified in Section 4.

3.2 Restricted velocity particle swarm optimization (RVPSO)

PSO is originally devised for solving unconstrained optimization problems, hence it does not take into account of the impact of the feasible region (or the constraints) on the search mechanism of PSO when solving COPs. We believe that if we could modify the inherent search mechanism of PSO so as to incorporate into it could the impact of the feasible region, then the performance of the resulting algorithm would be improved in solving COPs. Therefore, we propose in this subsection a novel PSO algorithm–restricted velocity PSO (RVPSO), which incorporates the impact of feasible region on the velocity of particles of the swarm.

Fig. 1 Pseudo-code of DOM

$\Phi_{ibest} = \Phi(\vec{p}_i)$, $f_{ibest} = f(\vec{p}_i)$, $\Phi_i = \Phi(\vec{x}_i^k)$
if $\Phi_i < \Phi_{ibest}$ **then** $\vec{p}_i \leftarrow \vec{x}_i^k$, $\Phi_{ibest} \leftarrow \Phi_i$ **end**
if $\Phi_i = \Phi_{ibest}$ and $\Phi_{ibest} \leq \delta$ **then**
  $f_i = f(\vec{x}_i^k)$
  if $f_i \leq f_{ibest}$ **then** $\vec{p}_i^k \leftarrow \vec{x}_i^k$, $f_{ibest} \leftarrow f_i$ **end**
end
$\Phi_{gbest} = \Phi(\vec{p}_g)$, $f_{gbest} = f(\vec{p}_g)$,
if $\Phi_i < \Phi_{gbest}$ then $\vec{p}_g \leftarrow \vec{x}_i^k$, $\Phi_{gbest} \leftarrow \Phi_i$ **end**
if $\Phi_g = \Phi_{gbest}$ and $\Phi_{gbest} \leq \delta$ **then**
  if $f_i \leq f_{gbest}$ **then** $\vec{p}_g \leftarrow \vec{x}_i^k$, $f_{gbest} \leftarrow f_i$ **end**
end

Consider the velocity update Eq. (5) of the CPSO, which can be divided into two parts. The first part $\omega\vec{v}_i^k$ represents the weighted previous velocity of the particle. The second part $c_1r_1(\vec{p}_g^k - \vec{x}_i^k) + c_2r_2(\vec{p}_i^k - \vec{x}_i^k)$ represents the velocity towards the potential region around $\vec{p}_i^k$ and $\vec{p}_g^k$. In order to reflect the impact of the feasible region, we make the following two modifications on (5):

(a) Set $c_1 = c_2 = 1$. This will make the potential position $\vec{x}_i^k + r_1(\vec{p}_g^k - \vec{x}_i^k) + r_2(\vec{p}_i^k - \vec{x}_i^k)$ not to leave far away from $\vec{p}_g^k$ and $\vec{p}_i^k$, particularly when $\vec{p}_g^k$ and $\vec{p}_i^k$ are already within the feasible region, the probability that the above potential position falls within the feasible region would be greatly increased, this would improve efficiency of particles' probing the feasible region.

(b) Replace $\omega\vec{v}_i^k$ by $\omega(\vec{p}_g^k - \vec{p}_j^k)$, where $\vec{p}_j^k$ is the best position of a randomly selected particle, and j is an random integer number uniformly distributed in the range $[1, N]$. This is because, on the one hand, $\omega\vec{v}_i^k$ in the CPSO does not reflect the impact of the feasible region, or it does not utilize the information about the feasible region. For example, in solving COPs, $\omega\vec{v}_i^k$ should not be too large, otherwise the particle would be very likely to leave too far away from the feasible region. Meanwhile, it should not be too small, otherwise the feasible region would not be sufficiently explored. Therefore, we should replace $\omega\vec{v}_i^k$ by an appropriate or "restricted" velocity term that can reflect in some way the impact (or information) of the feasible region. On the other hand, when $\vec{p}_g^k$ and $\vec{p}_j^k$ lie within the feasible region, $(\vec{p}_g^k - \vec{p}_j^k)$ reflects a portion of information about the size of the approximate region around $\vec{p}_g^k$ in the feasible region; Moreover, $(\vec{p}_g^k - \vec{p}_j^k)$ can be regarded as a sample of the velocity term pointing at $\vec{p}_g^k$, thus we take $\omega = 1$ in the resulting equation in this paper.

Based on the above two modifications, we obtain a novel PSO algorithm, refered to as restricted velocity PSO (RVPSO), which is especially applicable COPs. The update equations of RVPSO are as follows:

$$\vec{v}_i^{k+1} = r_1\big(\vec{p}_g^k - \vec{x}_k^k\big) + r_2\big(\vec{p}_i^k - \vec{x}_i^k\big) + \omega\big(\vec{p}_g^k - \vec{p}_j^k\big) \tag{9}$$

$$\vec{x}_i^{k+1} = \vec{x}_i^k + \vec{v}_i^{k+1} \tag{10}$$

where $r_1, r_2 \in U[0, 1]$, and ω is a scaling parameter. In this paper, we simply set $\omega = 1$.

Furthermore, the velocity term $\vec{v}_i^{k+1}$ in (9) and (10) can be cancelled. As a result, in RVPSO there is only one update equation (i.e., position update equation) of the form:

$$\vec{x}_i^{k+1} = \vec{x}_i^k + r_1\big(\vec{p}_g^k - \vec{x}_i^k\big) + r_2\big(\vec{p}_i^k - \vec{x}_i^k\big) + \omega\big(\vec{p}_g^k - \vec{p}_j^k\big) \tag{11}$$

When applying RVPSO to COPs, we still need to incorporate a constraint-handling method into RVPSO, e.g., our proposed DOM. In this paper, we combine DOM with RVPSO, i.e., DOM+RVPSO, abbreviated to DOPSO (dynamic-objective particle swarm optimization), a novel PSO algorithms aiming at solving COPs. It should be noted that, because DOM is rooted in the search mechanism of PSO, DOM is essentially inherent and indivisible part of DOPSO.

In addition, we use a simple way to clamp down the particles within the search space, i.e., if $x_{id}^k > u_d$ or $x_{id}^k < l_d$, then

$$x_{id}^k = \big(p_{id}^k + p_{gd}^k + p_{jd}^k + p_{j'd}^k\big)/4, \quad d = 1, \ldots, N \tag{12}$$

Step 0: Initialize $\vec{x}_i^0$, $l_i \leq x_{id}^0 \leq u_i$, $i = 1, \cdots, N$, $d = 1, \cdots, n$

Step 1: Update $\vec{x}_i^{k+1}$ according to (11), and clamp down the particles by use of (12) (if needed)

Step 2: Calculate $\vec{p}_i$ and $\vec{p}_g$ according to the procedures described in Figure 1

Step 3: If stop criteria are not satisfied, then go to Step 1

Fig. 2 Pseudo-Code of DOM+RVPSO

where d denotes the d-th dimension of the search space, j and j' are two random integers uniformly distributed in the range $[1, N]$.

In summary, the pseudo-ode of DOPSO is described in Fig. 2.

4 Experiments and discussions

To evaluate the performance of the proposed DOM and RVPSO as well as DOPSO, we conducted two experiments by means of 13 well-known benchmark functions described in (Runarsson and Yao, 2000). In our experiments, the size of the swarm is 50, runs = 30, $\epsilon = 0.001$, and $\delta = 0$.

First we tested the performance of DOM by comparing it with "keeping feasible solutions (KFS)", a constraint-handling method proposed by Hu and Eberhart (2002). The comparison results are described in Table 1, which shows that DOM outperforms KFS when they are both incorporated into CPSO. KFS typically requires a long time process of initialization, and this is prohibited in practical implementation for some problems. For example, KFS did not find any feasible solution for g05. For problems g02, g03, g11 and g13, KFS only tested the simplified version of them, so we did not list the corresponding results for these problems. Unlike KFS, DOM do not have any special requirements on the initialization the particles. Wherever the particles are positioned, the particles will always dynamically seeking for the feasible region. All the solutions obtained by DOM (+CPSO) are feasible and more accurate.

Next, we implemented DOM+RVPSO (RVPSO incorporating DOM) and compared it against DOM+CPSO (CPSO incorporating DOM). The corresponding results are summarized in Table 2. DOM+RVPSO outperforms or exhibits the same as DOM+CPSO on all 13 test problems except g02 in the best and median results. As for the mean results, DOM+RVPSO exhibits almost the same good performance, with g01, g02 and g05 being the exceptions, whereas in terms of worst results, the exceptions consists of g02 and g05. The comparison results in Table 2 show that DOM can be incorporated into different PSO algorithms and can solves COPs effectively.

Table 3presents the comparison results of DOPSO (i.e., DOM+RVPSO) with respect to PESO (Zavala et al., 2005), which is still the state-of-the-art PSO-based algorithm for solving COPs. The experimental settings in PESO are the same as that in DOPSO. It can be seen from Table 3, DOPSO performs the same as or better than PESO in best results for the benchmark functions except for problem g02. It is clear that DOM+RFPSO performs better than PESO in mean and worst results for g07 and g11, but the situation is converse for g01, g02, g05 and g13. In a word, COM+RVPSO is highly competitive and comparable with PESO. However, PESO consumed 350,000 FES in its experiment, our DOM used only 50,000 FES.

In this paper, we also propose to use histogram to describe the test results more sufficiently and effectively. For example, as we have seen in Table 2, DOM+RVPSO is better than

Table 1 Comparison between DOM+CPSO and KFS+CPSO (Hu and Eberhart, 2002)("/" indicates that the corresponding problems were not tested; "–" indicates that the problem was tested but no feasible solutions were available)

		DOM+CPSO	KFS+CPSO	DOM+CPSO	KFS+CPSO	DOM+CPSO	KFS+CPSO
Fun	Optimal	Best		Mean		Worst	
g01	−15	−15	−15	−14.4187	−15	−12.4531	−15
g02	−0.803619	−0.664028	/	−0.413257	/	−0.259980	/
g03	−1	−1.0050	/	−1.0025	/	−0.9334	/
g04	−30665.539	−30665.539	−30665.5	−30 665.539	−30665.5	−30665.539	−30665.5
g05	5126.4981	5126.4842	—	5241.0549	—	5708.2250	—
g06	−6961.81388	−6961.81388	−6961.7	−6961.81388	−6960.7	−6961.81388	−6956.8
g07	24.306	24.306	24.4420	24.317	26.7188	24.385	31.1843
g08	−0.095825	−0.095825	−0.09583	−0.095825	−0.09583	−0.095825	−0.09583
g09	680.630	680.630	680.657	680.630	680.876	680.630	681.675
g10	7049.3307	7049.2480	7131.01	7049.2701	7594.65	7049.5969	8823.56
g11	0.75	0.749	/	0.749	/	0.749	/
g12	−1	−1	−1	−1	−1	−1	−1
g13	0.0539498	0.0538666	/	0.6811235	/	2.0428924	/

Table 2 Comparison between DOM+RVPSO and DOM+CPSO

Fun	Optimal	DOM+	Best	Median	Mean	Worst	Std.
g01	−15	RVPSO	−15	−15	−14.4187	−12.4531	8.5e−1
		CPSO	−15	−15	−14.6094	−11.8281	.9.1e+1
g02	−0.803619	RVPSO	−0.664028	−0.380820	−0.413257	−0.259980	1.2e−1
		CPSO	−0.803578	−0.710560	−0.700890	−0.483212	7.6e−2
g03	−1	RVPSO	−1.0050	−1.0051	−1.0025	−0.9334	1.3e−2
		CPSO	−1.0042	−0.9979	−0.9753	−0.7771	4.7e−2
g04	−30665.539	RVPSO	−30665.539	−30665.539	−30665.539	−30665.539	1.2e−11
		CPSO	−30665.539	−30665.539	−30665.539	−30665.539	3.3e−9
g05	5126.4981	RVPSO	5126.4842	5127.0038	5241.0549	5708.2250	1.8e+2
		CPSO	5126.9691	5193.0519	5233.9116	5625.5668	1.2e+2
g06	−6961.81388	RVPSO	−6961.81388	−6961.81388	−6961.81388	−6961.81388	4.6e−12
		CPSO	−6961.81384	−6961.81324	−6961.81269	−6961.8094	1.3e−3
g07	24.306	RVPSO	24.306	24.307	24.317	24.385	2.4e−2
		CPSO	24.431	25.904	25.988	28.350	1.1e+0
g08	−0.095825	RVPSO	−0.095825	−0.095825	−0.095825	−0.095825	1.4e−17
		CPSO	−0.095825	−0.095825	−0.095825	−0.095825	1.5e−17
g09	680.630	RVPSO	680.630	680.630	680.630	680.630	5.4e−13
		CPSO	680.633	680.659	680.667	680.758	3.0e−2
g10	7049.3307	RVPSO	7049.2480	7049.2483	7049.2701	7049.5969	7.9e−2
		CPSO	7070.2635	7296.8247	7356.0522	7874.0740	1.9e+2
g11	0.75	RVPSO	0.749	0.749	0.749	0.749	2.4e−012
		CPSO	0.749	0.749	0.749	0.749	1.5e−5
g12	−1	RVPSO	−1	−1	−1	−1	0
		CPSO	−1	−1	−1	−1	0
g13	0.0539498	RVPSO	0.0538666	0.6459288	0.6811235	2.0428924	4.0e−1
		CPSO	0.2767082	0.8734151	1.0212616	3.8933448	7.9e−1

Table 3 Comparison between RVPSO (i.e., DOPSO) and PESO (Zavala et al., 2005)

		DOPSO	PESO	DOPSO	PESO	DOPSO	PESO	DOPSO	PESO
Fun	Optimal	Best		Mean		Worst		Std	
g01	−15	−15	−15	−14.4187	−15	−12.4531	−15	8.5e−1	0
g02	−0.803619	−0.664028	−0.792608	−0.413257	−0.721749	−0.259980	−0.614135	1.2e−1	5.4e−2
g03	−1	−1.0050	−1.0050	−1.0025	−1.005006	−0.9334	−1.004989	1.3e−2	4.8e−6
g04	−30665.539	−30665.539	−30665.539	−30665.539	−30665.539	−30665.539	−30665.539	1.2e−11	0
g05	5126.4981	5126.4842	5126.4842	5241.0549	5129.1783	5708.2250	5148.8594	1.8e+2	5.1 6e+0
g06	−6961.81388	−6961.81388	−6961.81388	−6961.81388	−6961.81388	−6961.81388	−6961.81388	4.6e−12	0
g07	24.306	24.306	24.307	24.317	24.371	24.385	24.594	2.4e−2	7.0e−2
g08	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	1.4e−17	0
g09	680.630	680.630	680.630	680.630	680.630	680.630	680.630	5.4e−13	2.6e−7
g10	7049.3307	7049.2480	7049.4595	7049.2701	7099.1014	7049.5969	7251.3962	7.9e−2	5.9e+1
g11	0.75	0.749	0.749	0.749	0.749	0.749	0.749	2.4e−012	0
g12	−1	−1	−1	−1	−1	−1	−1	0	0
g13	0.0539498	0.0538666	0.081498	0.6811235	0.626881	2.0428924	0.997586	4.0e−1	2.2e−1

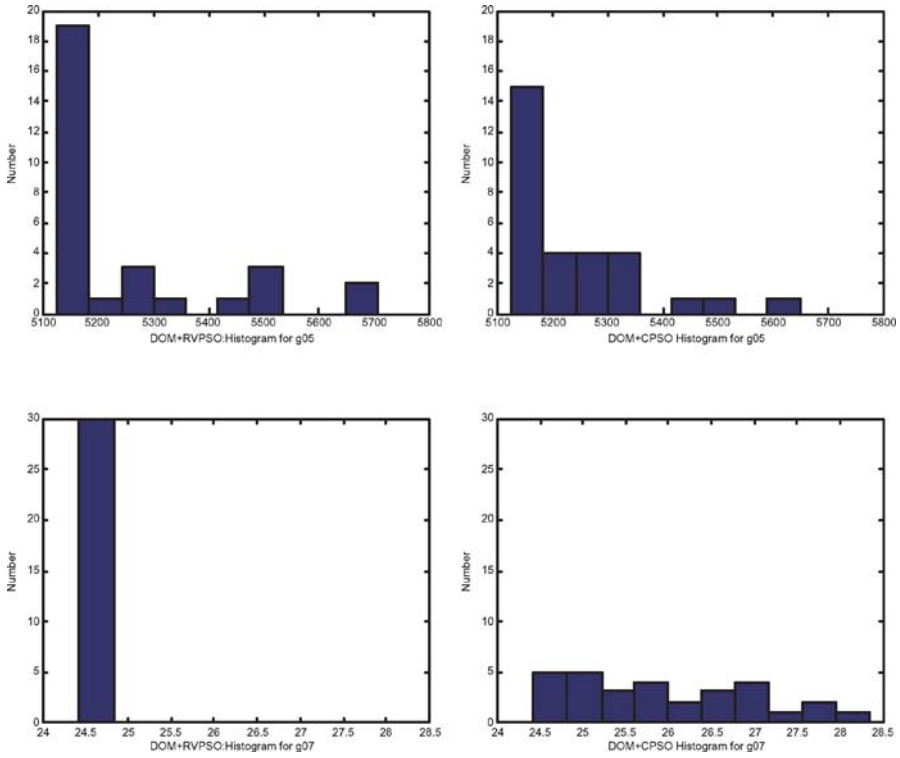

Fig. 3 Histograms of test results for g05 and g07

DOM+CPSO in best result for g05, but worse than DOM+CPSO in worst result. This to some degree causes "bad" mean results and "bad" standard deviation of DOM+RVPSO. However, for g05, Fig. 3 shows that the distribution of test results of DOM+RVPSO is better than that of DOM+CPSO. g07 exemplifies a different situation, in which DOM+RVPSO is better than DOM+CPSO in all comparison criteria, and the comparison results for g07 depicted in Fig. 3 are consistent with those in Table 2. Therefore, histogram is a meaningful criterion for evaluating the performance of EAs, and it can be used as an supplement to other commonly used criteria.

5 Conclusions and future work

In this paper, we have investigated the methods on how to modify or improve the PSO algorithm to make it more applicable to COPs. The contribution of this paper lies in three aspects. We firstly presented a new constraint-handling method—dynamic-objective method (DOM), which is a general technique that can be incorporated into various kinds of PSO algorithms for solving COPs. Secondly, a novel PSO algorithm, called restricted velocity particle swarm optimization (RVPSO) is proposed, which incorporates the impact or information of feasible region into its search mechanism. In addition, we suggest to use the histogram of test results as a criterion for evaluating the performance of an EA, our experiments show that this criterion can provide relative comprehensive insight into the performance of the algorithms.

Our experimental results have shown that DOM is very efficient and effective in handling constraints, and our DOPSO (i.e., DOM+RVPSO) is highly competitive in solving COPs.

We believe that the development obtained in the work is only a start, a lot of work needs to be done along the line of thinking adopted in this work. There is a large space for RVPSO to be further improved. The performance of DOM incorporated into other PSO algorithms needs to be investigated, and the comparison of DOM with other constraint-handling methods is also to be done in the future work.

Acknowledgments We wish to dedicate this paper to the memory of Dr. Yong He, a full professor at Zhejiang University, who passed away on the 5th of August, 2005. We thank Dr. Guohui Lin, Dr. Zhiyi Tan, and the two anonymous referees for their valuable comments. We also thank Prof. Enyu Yao for her encouragement and helpful discussions on this work.

References

Clerc M, Kennedy J (2002) The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space. IEEE Trans Evolut Comput 6(1):58–73

Coath G, Halgamuge SK (2003) A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems. In: Proceedings of the 2003 congress on evolutionary computation. IEEE, pp 2419–2425

Coello CAC (2000) Treating constraints as objectives for single objective evolutionary computations. Eng Opt 32:275–308

Coello CAC (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Meth Appl Mech Eng 191(11–12):1245–1287

Hu X, Eberhart RC (2002) Solving constrained nonlinear optimization problems with particle swarm optimization. In: Proceedings of 6th world multiconference on systemics, cybernetics and informatics (SCI 2002), Orlando, USA

Hu X, Eberhart RC, Shi YH (2003) Engineering optimization with particle swarm. In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, pp 53–57

Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proc. IEEE Int. Conf. on Neural Networks, Perth, Australia, pp 1942–1948

Michalewicz Z, Schoenauer M (1996) Evolutionary algorithms for constrained parameter optimization problems. Evolut Comput 4(1):1–32

Parsopoulos KE, Vrahatis MN (2004) On the computation of all global minimizers through particle swarm optimizzation. IEEE Trans Evolut Comput 8(3):211–224

Runarsson TP, Yao X (2000) Stochastic ranking for constrained evolutionary optimization. IEEE Trans Evolution Comput 4(3):284–294

Shi Y, Krohling RA (2002) Co-evolutionary particle swarm optimization to solve min-max problems. In: Proceedings of 2002 IEEE congress on evolutionary computation. Honolulu, HI, pp 1682–1687

Pulido GT, Coello Coello Ca (2004) A constraint-handling mechanism for particle swarm optimization. In: Proceedings of the 2004 congress on evolutionary computation. IEEE, pp 1396–1403

Zavala M, Hernández Aguirre AE, A and ER Villa Diharce (2005) Constrained optimization via particle evolutionary swarm optimization algorithm (PESO). GECCO'05, Washington, DC, USA, pp 209–216