Computing and Informatics, Vol. 29, 2010, 701–718

# A HYBRID PARTICLE SWARM EVOLUTIONARY ALGORITHM FOR CONSTRAINED MULTI-OBJECTIVE OPTIMIZATION

Jingxuan Wei

*School of Computer Science and Technology*
*Xidian University, Xi'an 710071, China*
*&*
*Department of Maths and Computing*
*University of Southern Queensland, Australia*
*e-mail:* wjxjingxuan@yahoo.com.cn

Yuping Wang

*School of Computer Science and Technology*
*Xidian University, Xi'an 710071, China*

Hua Wang

*Department of Maths and Computing*
*University of Southern Queensland, Australia*

Manuscript received 17 March 2008; revised 19 August 2008
Communicated by Vladimír Kvasnička

**Abstract.** In this paper, a hybrid particle swarm evolutionary algorithm is proposed for solving constrained multi-objective optimization. Firstly, in order to keep some particles with smaller constraint violations, a threshold value is designed, the updating strategy of particles is revised based on the threshold value; then in order to keep some particles with smaller rank values, an infeasible elitist preservation strategy is proposed in order to make the infeasible elitists act as bridges connecting disconnected feasible regions. Secondly, in order to find a set of diverse and well-distributed Pareto-optimal solutions, a new crowding distance function is designed

for bi-objective optimization problems. It can assign larger crowding distance function values not only for the particles located in the sparse region but also for the particles located near to the boundary of the Pareto front. In this step, the reference points are given, and the particles which are near to the reference points are kept no matter how crowded these points are. Thirdly, a new mutation operator with two phases is proposed. In the first phase, the total force is computed first, then it is used as a mutation direction, searching along this direction, better particles will be found. The comparative study shows the proposed algorithm can generate widely spread and uniformly distributed solutions on the entire Pareto front.

**Keywords:** Conostrained multi-objective optimization, particle swarm optimization, evolutionary algorithm

## 1 INTRODUCTION

In general, many real-world applications involve complex optimization problems with variants competing specifications and constraints. Without loss of generality, we consider a minimization problem with decision space $S$, which is described as follows:

$$\begin{cases} \min_{x\in S} & f(x) = \{f_1(x), f_2(x), \ldots, f_m(x)\} \\ \text{s.t.} & g_j(x) \le 0, j = 1, 2, \ldots, s \\ & h_j(x) = 0, j = 1, 2, \ldots, p, \end{cases} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n)$ is the decision vector, $S$ is an $n$-dimensional rectangle space defined by the parametric constraints $l_k \le x_k \le u_k$, $k = 1, 2, \ldots n$. In constrained multi-objective optimization problems, all equality constraints can be converted to inequality constraints by $|h_j(x)| \le \varepsilon$, where $\varepsilon$ is a small value. This allows us to deal with only the inequality constraints. Constraint violation is defined as $\Phi(x) = \sum_{j=1}^{s+p} \max(0, g_j(x))$. $f_1, f_2, \ldots, f_m$ are $m$ objectives to be minimized. The aim of the constrained multi-objective optimization problems (constrained MOPs) is to find multiple nondominated solutions under constraints. If these nondominated solutions are uniformly distributed and widely spread along the Pareto front, their quality is better.

The use of evolutionary algorithms for multi-objective optimization problems has significantly grown in the last five years [1–4]. As any other research areas, multi-objective evolutionary algorithms (MOEAs) have presented certain trends. One of them is to improve the efficiency of both of the algorithms and of the data structures used to store nondominated vectors. Researchers have produced some clever techniques to maintain diversity [5], new algorithms that use very small populations [6]. The most promising ones are nondominated sorting genetic algorithm II (NSGAII) [7], the strength Pareto evolutionary algorithm 2 (SPEA2) [8], the incrementing MO evolutionary algorithm (IMOEA), etc. Despite all these rapid developments, there seem to be not enough studies concerning handling constraints.

Constraint handling is a crucial part of real-world problem solving and it is time that MOEA researchers focus on solving constrained MOPs [9].

One of the typically algorithms to solve constrained MOPs is NSGAII with constraint dominance principle [7], which allow each feasible solution has a better rank than any infeasible one. Obviously, the main drawback of this principle is that the algorithm probably results in the premature convergence. In order to overcome the problem, some relevant algorithms are proposed [10, 11]. In these problems, the infeasible solutions which are located near to the boundary of the feasible region and have small rank values are kept during the evolution. But the infeasible solutions with larger constraint violations and smaller rank values are not concerned. In fact, these solutions may be valuable for finding the true Pareto front. So we design two new infeasible particle preservation strategies in Section 3.

Particle swarm optimization (PSO) [12, 13] is a relatively recent heuristic inspired by the choreography of a bird flock. PSO seems suitable for the multi-objective optimization mainly because of the high speed of convergence that the algorithm presents for single-objective optimization. However, such convergence speed may be harmful in the context of multi-objective optimization, because a PSO-based algorithm may converge to a false Pareto front. In order to overcome this drawback, some PSO algorithms incorporate a mutation operator which can enriches the exploratory capabilities of algorithms [14, 15].

In recent years, the field of MOPSO has been steadily gaining attention from research community [16–19]. While PSO is rarely considered in constrained MOPs [20].

In this paper, we develop a hybrid particle swam evolutionary algorithm to solve constrained MOPs. The proposed algorithm is able to find a diverse and well-distributed nearly optimal Pareto front for every test fuction. In order to overcome the drawbacks mentioned above, firstly, some infeasible solutions with smaller rank values and constraints are preserved during optimization which can avoid premature convergence. Secondly, in order to make the Pareto-optimal solutions competitive in terms of diversity and distribution, a new crowding distance function is proposed for two-objective optimization problems. In this section, the particles located in the sparse region of the objective space and near to the boundary of the Pareto front are assigned larger distance function values. Thirdly, although the mutation operator in PSO is not new, we design a new mutation operator which includes two steps. The first step: in the original PSO, every particle learns from its own pbest and gbest. However, it can not learn from other particles, in order to make full use of the beneficial information of other particles, we design a mutation operator based on the concept of total force. The force is computed first, which leans toward the particle of small constraint violation or small objective function value. Then it is used as a mutation direction, searching along this direction, better particles will be found. The second step: in order to avoid particles searching along one fixed direction, we select some particles generated by the first step to undergo the second mutation, which can guarantee the convergence of our algorithm. A hybrid particle swam evolutionary algorithm for constrained MOPs is proposed. The numerical

simulations for five constrained MOPs are made. The performances of the proposed algorithm are compared with that of NSGAII and MOPSO. The results indicate that the proposed algorithm has better performances for the test functions, especially for the problems with two or more disconnected feasible regions.

## 2 PRELIMINARIES

### 2.1 Particle Swarm Optimization

PSO has been developed by Kennedy and Eberhart [12, 13]. The original PSO formulae are:

$$V_i(t+1) = \omega V_i(t) + c_1 \mathrm{rand}_1(\mathrm{pbest}_i(t) - x_i(t)) + c_2 \mathrm{rand}_2(\mathrm{gbest}(t) - x_i(t)) \quad (2)$$

$$x_i(t+1) = x_i(t) + V_i(t+1), i = 1, 2, \ldots, \mathrm{popsize}. \quad (3)$$

In essence, the trajectory of each particle is updated according to its own best position pbest , and the best position of the whole swarm denoted as gbest. $V_i = (V_{i1}, V_{i2}, \ldots, V_{in})$ represents the velocity of the $i^{\mathrm{th}}$ particle, and the $i^{\mathrm{th}}$ particle is denoted as a $n$-dimensional vector $x_i = (x_{i1}, x_{i2}, \ldots x_{in})$, namely the position of the $i^{\mathrm{th}}$ particle is $x_i$, and every particle represents a potential solution. $rand_1$ and $rand_2$ are two random values obeying uniform distribution in $[0, 1]$. $c_1$ is the cognition weight and $c_2$ is the social weight. popsize is the size of the swarm.

### 2.2 Constrained MOPs

**Definition 1.** For the multi-objective optimization, a vector $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$ is said to dominate a vector $\nu = (\nu_1, \nu_1, \ldots, \nu_n)$ (denoted as $\mu \prec \nu$) if: $\forall i \in \{1, 2, \ldots, m\}$, $f_i(\mu) < f_i(\nu) \wedge \exists j \in \{1, 2, \ldots, m\}$, $f_j(\mu) < f_j(\nu)$. A solution $x \in S$ is called a Pareto-optimal solution for problem (1), if $\Phi(x) = 0$ and $\sim \exists \overrightarrow{x} \in S$ such that $\Phi(\overrightarrow{x}) = 0$ and $\vec{x} \prec x$. All Pareto-optimal solutions constitute the Pareto-optimal set. The corresponding set in the objective space is called Pareto front.

## 3 THE PRESERVATION STRATEGIES OF THE PARTICLES

From above, in [9, 10, 11] each feasible solution has a better rank than infeasible one, and in MOPSO [20], when two particles are compared, constrained Pareto dominance is directly used to decide which one is the winner. Obviously, these algorithms have one common drawback: the feasible ones are better than the infeasible ones. This will result in premature convergence, especially for the problems with two or more disconnected feasible regions. In order to overcome the problems mentioned above, we propose two strategies for keeping some infeasible solutions which act as bridges to connect the feasible regions.

### 3.1 The New Comparison Strategy in Particle Swarm Optimization

In order to keep some particles with smaller constraint violations, a new comparison strategy is proposed. Firstly, a threshold value is proposed: $\psi = \frac{1}{T}\sum_{i=1}^{\text{popsize}} \Phi(x_i)/$ popsize, $T$ is a parameter, which decreases from 0.4-0.8 with the increasing of the generation number. In every generation, if the constraint violation of a particle is less than the threshold value, the particle is acceptable, else it is unacceptable. The comparison strategy is described as follows:

1. If two particles are feasible solutions, we select the one with the smaller rank values.
2. If two particles are infeasible, we select the one with the smaller constraint violation.
3. If one is feasible and the other is infeasible, if the constraint violation of the infeasible one is smaller than the threshold value and it is not dominated by the feasible one, we select the infeasible one.

### 3.2 Infeasible Elitist Preservation Strategy

In order to keep some particles with smaller rank values, some infeasible solutions beside the feasible ones should be kept to act as a bridge connecting two or more feasible regions during optimization. The procedure of the process of keeping and updating infeasible elitists is given by Algorithm 1. Let $R$ denotes the Pareto-optimal set found so far, IR denotes the infeasible elitist set, and SI denotes the size of the infeasible elitist set. SR denotes the size of $R$.

**Algorithm 1. Step 1:** For every infeasible particle in the swarm, if it is not dominated by any other particle in the set $R$, add it to IR.

**Step 2:** Remove particles in IR which are dominated by any other member of $R$.

**Step 3:** If the size of the IR exceeds the given maximum number SI, then do:

1. select SI particles in IR with smaller rank values when $t < t_{\text{mean}}$.
2. select SI particles in IR with smaller constraint violations when $t > t_{\text{mean}}$.

The rank value of $x_i$ is equal to the number of the solutions that dominate it. $t$ is the current generation number and $t_{\text{mean}} = \frac{t_{\max}}{2}$, $t_{\max}$ is the maximum generation number. At the early stage of evolution($t < t_{\text{mean}}$) we need to enlarge the search domain in order to keep the diversity of the swarm. So the particles of smaller rank values are kept, which will make the swarm converge to the true Pareto front, and at the later stage of evolution ($t > t_{\text{mean}}$), in order to guarantee the convergence of the swarm, the particles of smaller constraint violations are kept.

## 4 A NOVEL CROWDING DISTANCE FUNCTION

In multi-objective optimization, we have to find a set of solutions which is high competitive in terms of convergence, distribution, and diverse. In section 3, the infeasible elitist preservation strategy and the new comparison strategy make the algorithm have better convergence but can not guarantee the diversity and distribution of Pareto-front. In [7, 20], crowded-comparison operator is adopted which guides the search toward a sparse region of Pareto front, but it does not pay more attention to the crowded regions near to the boundary of Pareto front. However, the boundary points (they are also called reference points) can guarantee the diversity of the Pareto-front, some particles located near to the boundary of the Pareto front are of important information. In this part, for the two-objective optimization, we design a novel crowding distance function denoted as crowd($x$), which assigns larger function values not only for the solutions located in a sparse region, but also for the solutions located in a crowded region near to the boundary of the Pareto front.

The procedure is given by Algorithm 2. Suppose that the current Pareto front is composed of $N$ points, denoted as $M = \{u_1, u_2, \ldots, u_N\}$, and the corresponding points in the decision space are denoted as $x_1, x_2, \ldots, x_N$, let $u_b$, $u_c$ be two reference points of the Pareto front (can be seen from Figure 1), and the corresponding points in the decision space be $x_b, x_c$. $x_b = \arg\min\{f_2(x_1), f_2(x_2), \ldots, f_2(x_N)\}$, $x_c = arg\min\{f_1(x_1), f_1(x_2), \ldots, f_1(x_N)\}$.

**Algorithm 2. Step 1:** Calculate the Euclidean distance between every point in the Pareto front and $u_b$, find region $M_1$ which is near to the boundary of the Pareto front:

$$D_{i1} = \text{dist}(u_i, u_b), i = 1, 2, \ldots, N$$

$$\overline{D_1} = \sum_{i=1}^{N} D_{i1} \frac{1}{N}$$

$$M_1 = \{u_j \mid \text{dist}(u_j, u_b) \leq 0.5\overline{D_1}, 1 \leq j \leq N, u_j \neq u_b\}.$$

**Step 2:** Calculate the Euclidean distance between every point in the set $M/M_1$ and $u_c$, find region $M_2$ which is near to the reference point:

$$D_{i2} = \text{dist}(u_i, u_c), u_i \in M/M_1$$

$$\overline{D_2} = \sum_{i=1}^{|M/M_1|} D_{i2} \frac{1}{|M/M_1|}$$

$$M_2 = \{u_j | \text{dist}(u_j, u_c) \leq 0.5\overline{D_2}, u_j \in M/M_1, u_j \neq u_c\}$$

**Step 3:** Calculate the crowding distance of every point in the Pareto front:

$$\mathrm{crowd}(x_i) = \begin{cases} \mathrm{crowd}_1(x_i) \cdot \frac{\mathrm{dist}(u_b,u_c)}{\mathrm{dist}(u_b,u_i)}, & u_i \in M_1 \\ \mathrm{crowd}_1(x_i) \cdot \frac{\mathrm{dist}(u_b,u_c)}{\mathrm{dist}(u_c,u_i)}, & u_i \in M_2 \\ \mathrm{crowd}_1(x_i) \cdot \frac{\mathrm{dist}(u_b,u_c)}{\min\{a_{\min},b_{\min}\}}, & u_i = u_b \text{ or } u_i = u_c \\ \mathrm{crowd}_1(x_i), & \text{else} \end{cases} \tag{4}$$

where $a_{\min} = \min\{\mathrm{dist}(u_b, u_i) | u_i \in M_1\}$, $b_{\min} = \min\{\mathrm{dist}(u_b, u_i) | u_i \in M_2\}$, $\mathrm{crowd}_1 = \frac{D_1+D_2}{2}$, $D_1, D_2$ are two nearest distances between point $u_i$ and the other points in the Pareto front.

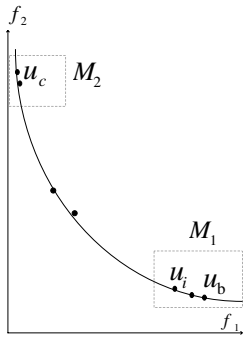



Fig. 1. Crowding distance calculation

In doing so, when the Pareto optimal solutions exceed the given number, we keep these particles with larger crowding distance function values, namely the particles located in the sparse region of the objective or near to the boundary of the Pareto font are preserved. So finally, we will get a widely spread and uniformly distributed Pareto front.

## 5 USE OF A NOVEL MUTATION OPERATOR

In PSO, the search scheme is singleness: every particle is updated only according to its own best position as well as the best particle in the swarm. So it often results in two outcomes:

1. the loss of diversity required to maintain a diverse pareto front;
2. premature convergence if the swarm is trapped into a local optima.

The infeasible elitist preservation strategy can overcome the premature convergence in some extent, but it is not enough. This motivated the development of a mutation operator that tries to explore all particles at the beginning of the search. In this

section, a novel mutation operator with two phases is proposed. The aim of the first phase is to make full use of all particles in the swarm, and define a mutation direction. Searching along this direction, better new particles will be found. In order to make particles search different regions, which can guarantee the convergence of the algorithm, the second phase of mutation is proposed.

More details are described as follows:

1. The first phase of mutation: We can image that there is a force between any two particles in the search space, particle $x$ has a force to $y$ ($y$ is the selected particle according to the mutation probability $p_{m1}$) which is defined as (5). For a swarm including $y$, swarm pop has a total force to y which is the sum of forces of each particle in pop to $y$, it can be defined by $F(y)$ and calculated by $F(y) = \sum_{x_i \in \text{pop}, x_i \neq y} F_i(x_i, y)$. The mutation to $y$ is a random variation along the direction $\frac{F(y)}{\|F(y)\|}$, its offspring is defined as $fy_1 = y + \lambda \cdot \frac{F(y)}{\|F(y)\|}$. $\lambda \in (0, 1)$ is the step size when $y$ searches along the direction $\frac{F(y)}{\|F(y)\|}$.

$$F_i(x_i) = \frac{x_i - y}{\|x_i - y\|} \cdot \frac{C - IF(x_i)}{\text{rank}(x_i)} \tag{5}$$

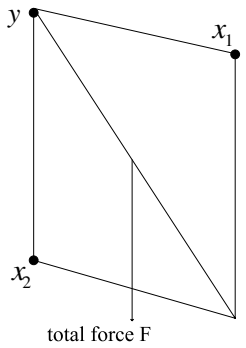



Fig. 2. The total force imposed on $y$

$IF(x_i)$ is the constraint violation function, $IF(x_i) = \sum_{j=1}^{s+p} \max(0, g_j(x_i)), x_i \neq y, i = 1, 2, \ldots,$ popsize, popsize is the size of the swarm. $\text{rank}(x_i)$ is the rank value of $x_i$ in the current swarm. $C$ is a big constant, let $C = 10\,000$. This can be graphically illustrated by Figure 2.

From Figure 2 and Equation (5), we can see that the total force imposed on $y$ leans to the particle with lower constraint violation or smaller rank value. So searching along this direction, better particles may be found.

2. The second phase of mutation: In order to avoid particles searching along one fixed direction, for some offspring of the first mutation, the second phase

of mutation is performed: for $fy_1$ ($fy_1$ is selected according to the second mutation probability $p_{m2}$, we divide the $i^{\text{th}}$ interval $[l_i, u_i]$ into $n$ subintervals $[\varpi_1^i, \varpi_2^i], \ldots, [\varpi_i^{n_i-1}, \varpi_i^{n_i}]$, where $|\varpi_j^i - \varpi_{j-1}^i| < \varepsilon$, $j = 2, \ldots n_i$, randomly generate a number $\delta \in (0, 1)$; if $\delta < p_{m2}$, then randomly select one $\varpi_j^i$ in $\{\varpi_1^i, \varpi_2^i, \ldots, \varpi_{n_i}^i\}$ as the value of the $i^{\text{th}}$ dimension.

## 6 THE PROPOSED ALGORITHM

**Algorithm 3** (Hybrid particle swarm evolutionary algorithm: HPSEA)**.**

**Step 1:** Choose the proper parameters, given swarm size popsize, randomly generate initial swarm pop(0) including every particle's position and velocity, find the Pareto optimal solutions in the current swarm and copy them to $R$. Let $t = 0$.

**Step 2:** Preserve infeasible elitist set IR according to Section 3.

**Step 3:** For every particle, let $\text{pbest}_i = x_i, i = 1, 2, \ldots, \text{popsize}$.

**Step 4:** Select the best position of the swarm as *gbest*, *gbest* is randomly selected from set $R$.

**Step 5:** For every particle in the swarm, do:

1. If $\text{rand}(0, 1) < p_c$, $p_c = 0.2 + e^{-\left(\frac{t}{t_{\max}}+1\right)}$, $\text{pbest}_i$ is selected randomly from the set IR, else $\text{pbest}_i$ is unchangeable.
2. The position and velocity of $x_i$ are updated according to (2), (3). The new swarm is denoted as $\overline{\text{pop}}(t + 1)$.

**Step 6:** For every particle in $\text{pop}(t + 1)$, use mutation operator in Section 5 and get the swarm $\text{pop}(t + 1)$.

**Step 7:** Maintain the particles in $\text{pop}(t+1)$ within the search space: when a decision vector goes beyond its boundaries, then the decision vector takes the value of its corresponding boundary.

**Step 8:** Update the set $R$ with all particles in $\text{pop}(t+1)$, the update means adding all Pareto optimal solutions in $\text{pop}(t+1)$ into $R$, and delete the dominated ones in it.

If the size of $R$ is full, the predetermined number of particles with larger crowding distance function values are preserved.

**Step 9:** Update the infeasible elitist set IR with all particles in $\text{pop}(t+1)$ according to the infeasible elitist preservation strategy in Section 3.2.

**Step 10:** For every particle $x_i$, update its $\text{pbest}_i$ according to the new comparison strategy in 3.1.

**Step 11:** If the maximum number of the cycles has been reached, stop, output the solutions in the set $R$ as the Pareto optimal solutions, else go to step 4.

In step 5, when $\text{rand}(0,1) < p_c$, we choose the pbest from the set IR; in doing so, a particle's potential search space will be enlarged, where $p_c = 0.2 + e^{-\left(\frac{t}{t_{\max}}+1\right)}$ means the value is dynamically changed with the current generation number $t$. It means at the beginning of the search, in order to enlarge the potential search space and keep diversity of the swarm, a larger number of infeasible solutions are selected while at the later stage of evolution, in order to guarantee the convergence of the swarm, a smaller number of infeasible solutions are kept. So the infeasible elitists must be in a proper proportion, in this paper $p_c \in [0.335, 0.561]$.

## 7 THE PROPOSED ALGORITHM

### 7.1 Test Functions

There are five test problems from the literature in our experiments. The $F_1 \sim F_4$ are chosen from [9], $F_5$ is chosen from [7]. The above test problems provide difficulty to an algorithm in the vicinity of the Pareto front; difficulties also come from the infeasible search regions in the entire search space.

$$F1: \begin{cases} \min f_1(x) = x_1 \\ \min f_2(x) = c(x)\exp(-f_1(x)/c(x)) \\ \text{s.t. } g_1(x) = f_2(x) - 0.858\exp(-0.541 f_1(x)) \geq 0 \\ \text{s.t. } g_2(x) = f_2(x) - 0.728\exp(-0.295 f_1(x)) \geq 0 \end{cases}$$

$x = (x_1, x_2, \ldots, x_5)$, $0 \leq x_1 \leq 1$, $-5 \leq x_i \leq 5, i = 2, 3, 4, 5$.

$$F_2 \sim F_4: \begin{cases} \min f_1(x) = x_1 \\ \min f_2(x) = c(x)(1 - \frac{f_1(x)}{c(x)}) \\ \text{s.t. } g_1(x) = \cos(\theta)(f_2(x) - e) - \sin(\theta) f_1(x) \geq \\ a|\sin(b\pi(\sin(\theta)(f_2(x) - e) + \cos(\theta) f_1(x))^c)|^d \end{cases}$$

$x = (x_1, x_2, \ldots, x_5)$, $0 \leq x_1 \leq 1$, $-5 \leq x_i \leq 5, i = 2, 3, 4, 5$, parameter settings are described in Table 1. For $F_1 \sim F_4$, $c(x) = 41 + \sum_{i=2}^{5}(x_i^2 - 10\cos(2\pi x_i))$.

| Problem | $\theta$ | $a$ | $b$ | $c$ | $d$ | $e$ |
|---|---|---|---|---|---|---|
| $F_2$ | $-0.2\pi$ | 0.2 | 10 | 1 | 6 | 1 |
| $F_3$ | $-0.2\pi$ | 0.75 | 10 | 1 | 0.5 | 1 |
| $F_4$ | $-0.05\pi$ | 40 | 5 | 1 | 6 | 0 |

Table 1. Parameter settings in $F_2 \sim F_4$

$$F_5: \begin{cases} \min f_1(x) = x_1 \\ \min f_2(x) = (1 + x_2)/x_1 \\ \text{s.t. } g_1(x) = x_2 + 9x_1 \geq 6 \\ \text{s.t. } g_2(x) = -x_2 + 9x_1 \geq 1 \end{cases}$$

$x_1 \in [0.1, 1.0]$, $x_2 \in [0, 5]$.

### 7.2 Parameter Values

Each simulation run was carried out using the following parameters: swarm size: 100; infeasible elitist size: 20; size of $R$: 100; mutation probability $p_{m1} : 0.6, p_{m2} = 0.3$; maximum number of generations $t_{\max} : 200$, $c_1 = c_2 = 0.8$; $\omega_{\max} = 1.0, \omega_{\min} = 0.4$.

### 7.3 Performance Measures

In the following of this section let $A$, $B$ represent two sets of Pareto optimal solutions in the decision space, and $PA$, $PB$ denote the corresponding Pareto solution sets in the objective space. We propose three metrics to measure the quality, distribution, and diverse of the Pareto front. The first metric known as Q-measure [21] is used to compare the quality of solutions found by the two algorithms. The second known as S-measure [22] is used to evaluate the uniformity of the Pareto optimal solutions. The third known as the FS-measure [23] is used to measure the size of the space covered by the Pareto front found by an algorithm.

**Metric 1** (Q-measure)**.** Let $\Phi$ be the nondominated solution set of $A \cup B$, let $\Psi = \Phi \cap A$, $\Theta = \Phi \cap B$, the metric is defined by $M_1(A, B) = \frac{|\Psi|}{|\Phi|}$, $M_1(B, A) = \frac{|\Theta|}{|\Phi|}$. The solution set $PA$ has better convergence to the true Pareto front than the solution set $PB$, if and only if $M_1(A, B) > M1(B, A)$ or $M_1(A, B) > 0.5$.

**Metric 2** (Metric two (S-measure))**.** $S = [\frac{1}{n_{PF}} \sum_{i=1}^{n_{PF}} (d_i^{'} - \overline{d^{'}})^2]^{1/2}$, $\overline{d^{'}} = \frac{1}{n_{PF}} \sum_{i=1}^{n_{PF}} d_i^{'}$, where $n_{PF}$ is the number of the solutions in the known Pareto front, $\overline{d_i^{'}}$ is the distance (in the objective space) between the number $i$ and its nearest member in the known Pareto front. The smaller the value of S-measure, the more uniformity the Pareto front found by the algorithm will be.

**Metric 3** (Metric three (FS-measure))**.**
$FS = \sqrt{\sum_{i=1}^{m} \min_{(x_0,x_1)\in A\times A}\{(f_i(x_0) - f_i(x_1))^2\}}$, the larger the value of $FS$ is, the better diverse of solutions on the Pareto front will be.

### 7.4 Simulation Results and Comparisons

The proposed algorithm has been run on each test problem for 10 times independently. On each run, outcome of the simulation is the Pareto front found by the proposed algorithm (HPSEA). Figures 3–7. summarize the statistical performances of the different algorithms HPSEA, multi-objective PSO with constraints [20] (MPSO), and NSGAII with constraint Pareto dominance [7] (CNSGAII). In this section, "1" denotes HPSEA, "2" denotes CNSGAII, "3" denotes MPSO.

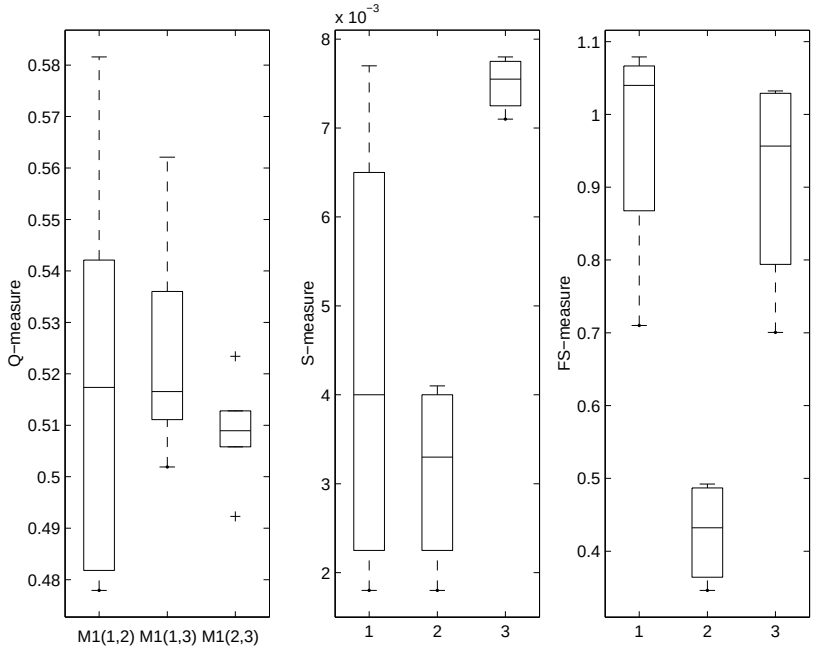



Fig. 3. Statistical performance on $F_1$

A. $F_1$

Since a part of the constrained boundary constitutes the Pareto optimal region, a spread in Pareto optimal solutions require decision variables to satisfy the inequality constraints. Thus, it may be difficult to find a number of solutions on a nonlinear constraint boundary. It can be observed from Figure 3 that most of the algorithms are able to find at least part of the true Pareto front. Figure 3 shows that HPSEA is able to evolve a diverse set of solutions due to the high value of FS-measure. We can see that the Pareto front found by CNSGAII is more uniform than those found by HPSEA and MPSO due to the low value of S-measure; but, it can be seen that in most runs, among the union of the solutions found by HPSEA and CNSGAII, the nondominated solutions found by HPSEA exceed 0.5, so the solutions found by HPSEA have better convergence to the true Pareto front than CNSGAII. Furthermore, the MPSO is unable to find a diverse and well distributed optimal Pareto front due to the high value of S-measure and a low value of FS-measure. Generally, the HPSEA shows competitve performance in terms of convergence and diversity.

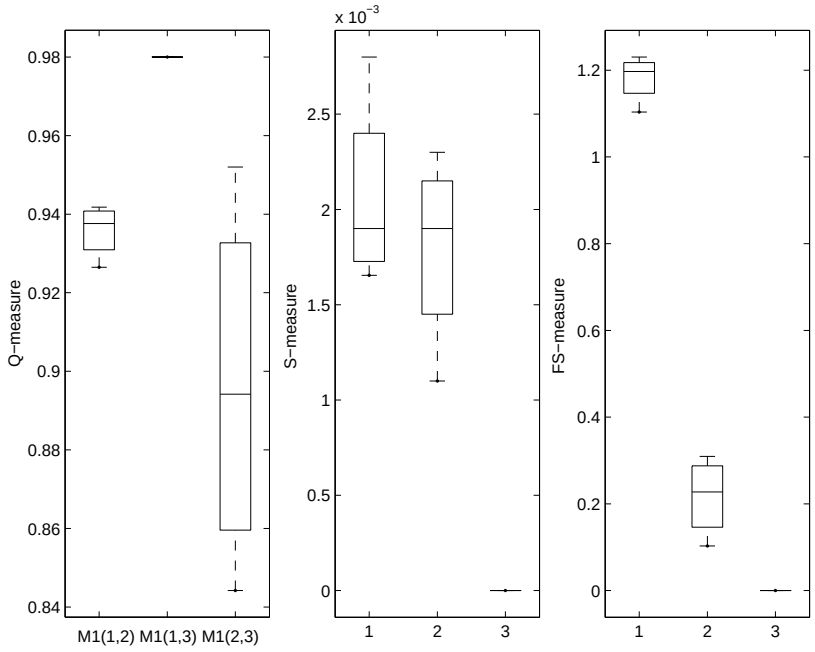



Fig. 4. Statistical performance on $F_2$

B. $F_2$

The nature of the constraint boundary of $F_2$ makes the Pareto optimal region discontinuous, having a number of disconnected continuous regions. The task of an optimization algorithm would be to find as many such disconnected regions as possible. It can be seen from Figure 4 that HPSEA and CNSGAII are able to find at least part of the true Pareto front, but MPSO can not find a few part of the true Pareto front. It can be observed that HPSEA is able to find a diverse and nearly Pareto optimal front due to the high value of Q-measure and FS-measure. We can also see that the Pareto front found by CNSGAII is more uniform than HPSEA due to the low value of S-measure, but CNSGAII can only find a small part of the true Pareto front. This is because HPSEA adopts two phases infeasible elitist preservation strategies which can maintain diversity of the swarm during optimization, and finally get the true Pareto front.

C. $F_3$

The problems have different form of complexity by increasing the value of parameter $a$, which has an effect of making the transition from continuous to discontinuous far away from the Pareto optimal region, since an algorithm has to travel through a long narrow feasible tunnel in search of the lone Pareto-optimal solutions at the end of the tunnel. Because the problem is more difficult compared to the previous problems, none of the algorithms can find the true

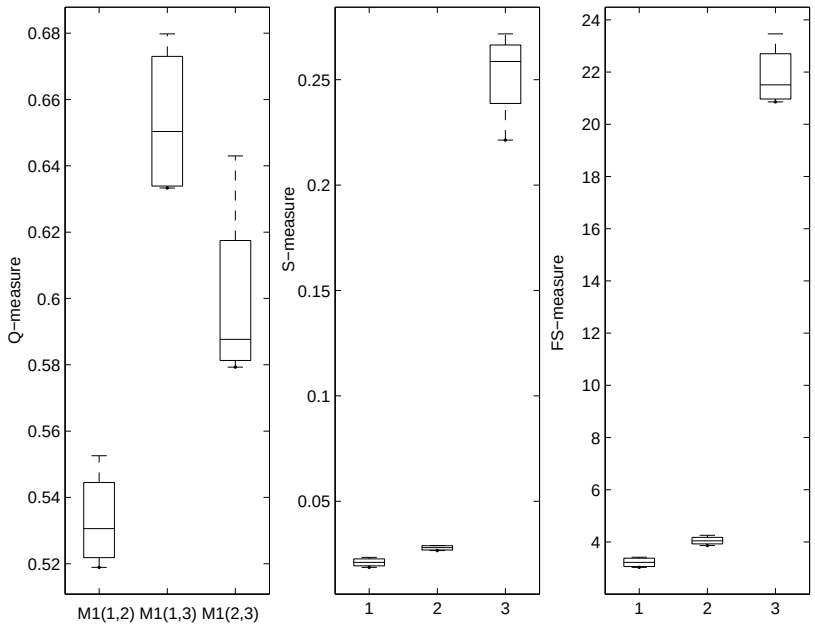



Fig. 5. Statistical performance on $F_3$

Pareto-front. Although MPSO maintains a better spread of solutions in terms of the high value of FS-measure in Figure 5, HPSEA is able to come closer to the true Pareto-front due to the high value of Q-measure. Algorithms which tend to converge anywhere in the Pareo front first and then work on finding a spread of solutions will end up finding solutions. HPSEA shows a similar behavior in this problem.

D. $F_4$

In this problem, the infeasibility in the objective search space can also come along the Pareto optimal region. In order to find such disconnected regions, an algorithm has to maintain adequate diversity. It can be observed from Figure 6 that all of the three algorithms can find a large part of the Pareto front. We can also see that the Pareto front found by HPSEA is more uniform than those found by the other two algorithms in terms of the low S-measure. From Figure 6, we know that the in most runs, solutions found by HPSEA have better convergence to the Pareto front than those by CNSGAII and MPSO due to the large value of Q-measure. From Figure 6, we see that MPSO maintains a better diverse Pareto front in terms of the high value of FS-measure. Thus, HPSEA shows competitive performance in terms of convergence and uniformity, MPSO shows competitive performance in terms of diversity.

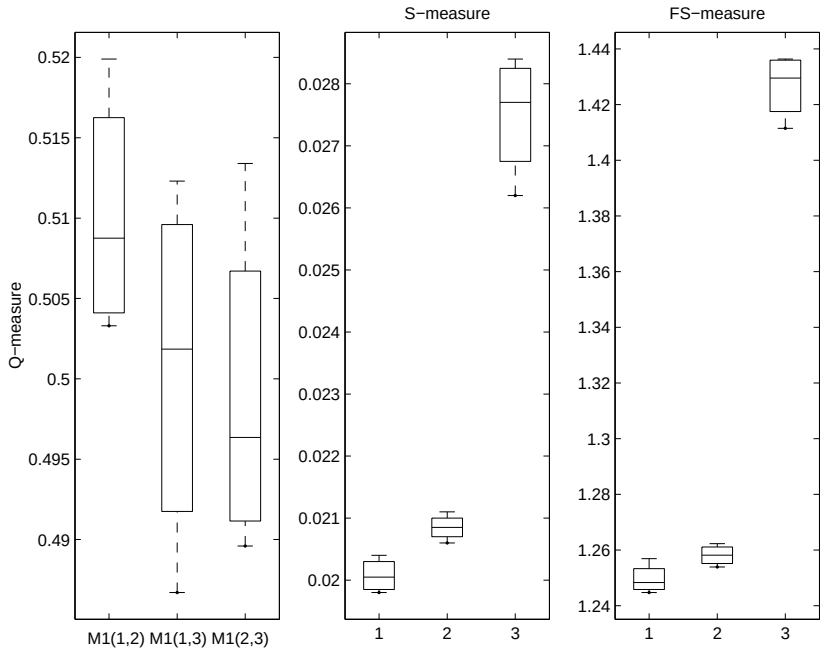



Fig. 6. Statistical performance on $F_4$

E. $F_5$
For the problem $F_5$, HPSEA have better convergence to the true Pareto front due to the high value of Q-measure. The Pareto front found by NSGAII is more uniform due to the low value of S-measure and MPSO finds a wider spread Pareto front due to the higher value of FS-measure.

Finally, we can conclude that for $F_1, F_2$, HPSEA performs best in terms of convergence and diversity, and for $F_3, F_4$, HPSEA performs best in terms of convergence and uniformity. For $F_5$, IPSEA performs best in terms of convergence, MPSO performs best in terms of diversity and CNSGAII performs best in terms of uniformity.

## 8 CONCLUSIONS

In this paper, a hybrid particle swarm evolutionary algorithm for constrained MOPs is proposed. It can effectively handle constrained multi-objective optimization problems. It can generate wide spread and uniformly distributed solutions along the entire Pareto front no matter what the shape of the Pareto front is. Moreover, the Pareto front generated by the proposed algorithm is more close to the true Pareto

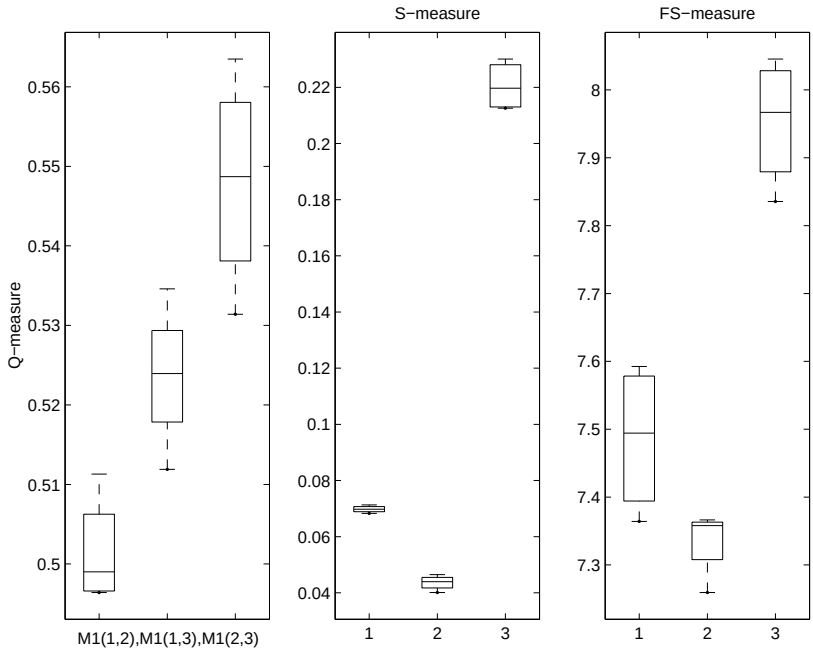



Fig. 7. Statistical performance on $F_5$

front. Compared to the other algorithms, the proposed algorithm is superior to these algorithms.

## Acknowledgement

This research is supported by National Natural Science Foundation of China (No. 60-873099).

## REFERENCES

[1] Hajela, P.—Lin, C. Y.: Genetic Search Strategies in Multicriterion Optimal Design. Structural Optimization, Vol. 11, 1992, No. 5, pp. 99–107.

[2] Fonseca, C. M.—Fleming, P. J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: S. Forrest, et al. (Eds.): Proc. 5th Inter. Conf. Genetic Algorithms, San Mateo, California, Morgan Kaufman, 1993, pp. 416–423.

[3] Horn, J. N.—Goldberg, D. E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In: N. Nafpliotis, et al. (Eds.): Proc. 1st IEEE Conf. Evolutionary Computation, IEEE World Congress on Computational Computation, Piscataway, NJ, IEEE, 1994 pp. 82–87.

[4] Srinivas, N.—Deb, K.: Multobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation, Vol. 2, 1994, No. 3, pp. 221–248.

[5] Knowles, J. D.—Corne, D. W.: Approximating the Nondominated Front Using The Pareto Archived Evolutionary Strategy. Evolutionary Computation, Vol. 8, 2000, pp. 149–172.

[6] Coello, C. A.—Pulido, G. T.: Multiobjective Optimization Using A Microgenetic Algorithm. In: L. Spector, et al. (Eds.): Proc. Genetic and Evolutionary Computation Conf. (CEC '2002). Honolulu, HI, 2002, pp. 1051–1056.

[7] Deb, K.—Pratap, A.—Agarwal, S.—Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGAII. IEEE Transactions on Evolutionary Computation, Vol. 6, 2002, No. 2, pp. 182–196.

[8] Zitzler, E.—Laumanns, M.—Thiele, L.: Improving the Strength Pareto Evolutionary Algorithm [R]. Zurich. 2001.

[9] Deb, K.—Pratap, A.—Meyarivan, T.: Spea 2: Constrained Test Problems for Multiobjective Evolutionary Optimization [R]. KanGAl Report No. 200002.

[10] Cai, Z.—Wang, Y.: A Multiobjective Optimization-based Evolutionary Algorithm for Constrained Optimization. IEEE Transactions on Evolutionary Computation, Vol. 10, 2006, No. 6, pp. 658–663.

[11] Zhou, Y.—He, J.: A Runtime Analysis of Evolutionary Algorithms for Constrained Optimization Problems. IEEE Transactions on Evolutionary Computation, Vol. 11, 2007, No. 5, pp. 608–619.

[12] Kennedy, J.—Eberhart, R.: Particle Swarm Optimization. In Proceedings of the IEEE International Conference on Neural Networks. IEEE Service Center, Piscataway, NJ, IV, 1995, pp. 1941–1948.

[13] Kennedy, J.—Eberhart, R.—Shi, Y.: Swarm Intelligence: SanFrancisco: Morgan Kaufmann Publishers. 2001.

[14] Xie, X.—Zhang, F.: A Dissipative Particle Swarm Optimization. Proc. of the IEEE Int. Conf. on Evolutionary Computation. IEEE Service Center, Piscataway, 2002, pp. 1666–1670.

[15] Eberhart, R. C.—Kennedy, J.: A New Optimizer Using Particle Swarm Theory. The 6$^{\text{th}}$ Int. Symp. on Micro Machine and Human Science. Nagoya, 1995, pp. 39–43.

[16] Coello, C.—Lechunga, M.: MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In Proc. of the 2002 Congress on Evolutionary Computation. IEEE Service Center, Piscataway, 2002, pp. 1051–1056.

[17] Li, X.: A Nondominated Sorting Particle Swarm Optimizer for Multiobjective Optimization: Formulation, Discussion and Generalization. In: E. Cantu-Paz, et al. (Eds.): Proc. Genetic and Evolutionary Computation, GECCO, Springer-Verlag, Berlin, 2003, pp. 37–48.

[18] Parsopoulos, K.—Vrahatis, M.: Particle Swarm Optimization Method in Multiobjective Problems. In Proc. 2002 ACM Symp. Applied Computing. Madrid, Spain, 2002, pp. 603–607.

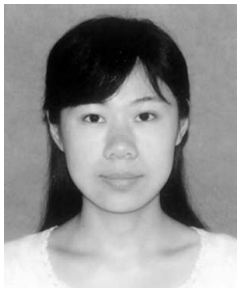
[19] Fieldsend, J.—Sing, S.: A Multi-Objective Algorithm Based Upon Particle Swarm Optimization, An Efficient Data Structure and Turbulence. In Proc. 2002 U.K. Workshop on Computational Intelligence. UK, 2002, pp. 147–152.

[20] Coello, C. A.—Toscano Pulido, G.: Handling Multiple Objectives with Particle Swarm Optimization. IEEE Trans on Evolutionary Computation, Vol. 8, 2004, No. 3, pp. 205–230.

[21] Wang, Y.—Dang, Ch.: Improving Multiobjective Evolutionary Algorithm by Adaptive Fitness and Space Division, ICNC'05, Springer-Verlag, Berlin. Changsha, China, 2005, pp. 392–398.

[22] Shaw, K. J. A.—Notcliffe, L.: Assessing the Performance of Multiobjective Genetic Algorithms for Optimization of Batch Process Scheduling Problem. In Proc. Conf. Evol. Comput. UK, 1999, pp. 37–45.

[23] Bosman Thierens, P. D.: The Balance Between Proximity and Diversity in Multiobjective Evolutionary Algorithms. IEEE Trans on Evolutionary Computation, Vol. 7, 2003, No. 2, pp. 174–188.



**Jingxuan Wei** received Ph. D. degree in applied mathematics from Xidian University, Xi'an, China. She is a lecturer at Xidian University. Her research interests include evolutionary computation, particle swarm optimization, multi-objective optimization.

**Yuping Wang** is a Full Professor at Xidian University, Xi'an, China. He has published more than 40 journal papers. He received the Ph. D. degree in Computational mathematics from Xi'an Jiaotong University, Xi'an, China, in 1993. his current research interests include evolutionary computation and optimization theory, methods and applications, etc.

**Hua Wang** is Associate Professor at Department of Maths and Computing, University of Southern Queensland.