International Journal of Swarm Intelligence and Evolutionary Computation

Ma and Forouraghi, Int J Swarm Intel Evol Comput 2014, 3:2
http://dx.doi.org/10.4172/2090-4908.1000113

Research Article | Open Access

Multi-objective Particle Swarm Optimization with Gradient Descent Search

Li Ma and Babak Forouraghi*

Department of Computer Science, Saint Joseph's University, USA

Abstract

Particle swarm optimization (PSO) has been proven to be a reliable method to deal with many types of optimization problems. Specifically, when solving multi-objective PSO (MOPSO) optimization problems careful attention must be paid to parameter selection and implementation strategy in order to improve the performance of the optimizer. This paper proposes a novel MOPSO with enhanced local search ability. A new parameter-less sharing approach is introduced to estimate the density of particles' neighborhood in the search space. Initially, the proposed method accurately determines the crowding factor of the solutions; in later stages, it effectively guides the entire swarm to converge closely to the true Pareto front. In addition, the algorithm utilizesthe local search method of gradient descent to better explore the Pareto-optimal region. The algorithm's performance on several test functions and an engineering design problem is reported and compared with other approaches. The obtained results demonstrate that the proposed algorithm is capable of effectively searching along the Pareto-optimal front and identifying the trade-offsolutions.

Keywords: Particle swarm optimization; Crowding factor; Gradient descent; Multi-objective optimization; Pareto front

Introduction

In most real-world engineering design problems multi-objective (MO) optimization plays an important role. Due to the nature of MO problems in which several objectives may conflict with each other, no single solution may be regarded as the optimal solution. Instead, a set of trade-off or Pareto-optimal solutions is provided for designers. The concept of Pareto dominance is utilized to generate the Pareto front which contains the non-dominated solutions. The goal of a MO algorithm, therefore, is to identify as many of the Pareto-optimal solutions as possible in the search space. Further, the discovered solutions must be well-distributed along the Pareto front [1-5].

Particle swarm optimization (PSO) has recently attracted a great deal of attention. PSO has been shown to effectively solve single-objective optimization problems by the virtue of its simplicity, quick convergence rate and the ability to find global optima [6-8]. Various versions of PSO have been proposed to meet different requirements. A fuzzy adaptive method was proposed to dynamically adapt the inertia weight of the PSO which is especially useful for optimization problems in dynamic environments [5]. To solve mechanical design optimization problems involving problem-specific constraints and mixed variables, an efficient constraint handling method called "fly-back mechanism" was proposed to maintain a feasible population in the swarm [9]. To enhance the search ability for finding local optima, a combination of PSO and simultaneous perturbation was proposed to solve multi-modal optimization problems, and this approach was successfully applied to evolve a neural network [10].

Due to its high efficiency, PSO for multi-objective (MOPSO) problems has become an active area of research in the recent years. A MOPSO approach has been proposed that uses the concept of Pareto dominance to determine the flight direction of a particle and maintains the discovered Pareto-optimal solutions in an external repository; in addition, a technique called hypercube formation is used to calculate fitness sharing [2]. A MOPSO was proposed that puts forward an efficient mutation strategy called elitist-mutation to effectively explore the feasible search space and speed up the search towards the Pareto front in conjunction with the crowding distance metric and a variable-sized external repository [3]. To minimize the mean value of the objectives and the standard deviation, a combination of MOPSO and the quasi-Newton method was introduced to find robust solutions against small perturbations of design variables [11].

In this paper a new PSO-based approach is presented. In the proposed algorithm the density of the search space is defined by a new crowding factor which determines social leaders by randomly selecting them from sparsely-populated areas. This new crowding factor is an improvement over previous methods in that not only it provides the estimation of the density in the neighborhood but it also provides a fitness sharing mechanism which degrades fitness values of non-promising solutions. Further, to enhance the local search ability of the algorithm a gradient descent method is applied to a small proportion of the global Pareto-optimal solutions when the size of the repository exceeds a user-defined threshold.

Multi-objective optimization

Multi-objective optimization (MO) is a methodology for finding optimal solutions to multivariate problems with multiple, and often conflicting objectives [12]. A general formulation of MO can be formally stated as:

Maximize/Minimize

$$F(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_n(\vec{x})]^T \tag{1}$$

Subject to:

$$g_i(\vec{x}) \le 0, h_k(\vec{x}) = 0 \tag{2}$$

Where $F(\vec{x})$ represents the objective vector, $g_i(\vec{x})$ represents the

***Corresponding author:** Babak Forouraghi, Computer Science Department, Saint Joseph's University, Pennsylvania, USA, Tel: 610-660-1167; Fax: 610-660-3082; E-mail: bforoura@sju.edu

Received December 23, 2013; **Accepted** May 10, 2014; **Published** May 20, 2014

Citation: Ma L, Forouraghi B (2014) Multi-objective Particle Swarm Optimization with Gradient Descent Search. Int J Swarm Intel Evol Comput 4: 113. doi: 10.4172/2090-4908.1000113

Citation: Ma L, Forouraghi B (2014) Multi-objective Particle Swarm Optimization with Gradient Descent Search. Int J Swarm Intel Evol Comput 4: 113. doi: 10.4172/2090-4908.1000113

design vector, $g_i(\vec{x})$ represents the inequality constraints, and $h_k(\vec{x})$ represents the equality constraints.

MO produces Pareto-optimal solutions along the Pareto front in the objective space. A design vector is Pareto-optimal or non-dominated if there is no other design vector that dominates it, which means no other design vector can optimize one dimension without simultaneously degrading at least one other dimension [13,14]. The goal of MO is to find as many of thenon-dominated and well-spread solutions as possible.

Particle swarm optimization

Particle swarm optimization (PSO) is a heuristic-based random-walk global optimization method, which is inspired by behavior of swarm of creatures [10]. A group of particles is randomly generated throughout the feasible design region, where X_i^k is the position of a particle *i* in the kth generation which is updated as shown in Equation 3:

$$X_i^{k+1} = X_i^k + V_i^{k+1} \tag{3}$$

And the velocity of the ith particle, V_i^{k+1}, is calculated as:

$$V_i^{k+1} = \omega V_i^k + c_1 r_1 (P_i^k - X_i^k) + c_2 r_2 (P_g^k - X_i^t) \tag{4}$$

where P_i^k is the current best position of the ith particle. The term $c_1 r_1 (P_i^k - X_i^k)$ represents the particle's inclination to repeat past behavior that has proven to be successful for that particular particle, where P_g^k is the best global position shared by the whole swarm. The term $c_2 r_2 (P_g^k - X_i^t)$ represents the particle's inclination to imitate or emulate the behavior of other particles that are successful [14]. Further, V_i^k is the particle's previous velocity; $c_1 r_1$ is the cognitive parameter, where r_1 is the cognitive learning rate and r_1 is a random number uniformly-distributed in the interval [0,1]; and $c_2 r_2$ is the social parameter, where r_2 is the social learning rate and r_2 is a random number uniformly-distributed in the interval [0,1]. The cognitive and social factors c_1 and c_2 are typically selected such that $c_1 \times c_1$ and $c_2 \times c_2$ have a mean of 1 so that the particles overshoot the attraction points and P_g^k half the time, thereby allowing wider search fronts [15]. Moreover, the parameter ω is the inertia weight which plays a critical role in balancing the global and local search abilities. The new velocities and positions of the particles are updated during the evolution of solutions in order to guide the swarm towards the global optima.

The proposed MOPSO methodology

In order to solve multi-objective problems, a combination of particle swarm optimization and the Pareto-dominance strategy can be used to find a set of Pareto-optimal solutions. An external repository is used to store the current set of discovered Pareto-optimal solutions.

When a particle violates the specified design constraints, the fly-back mechanism illustrated in Figure 1 is applied to force the particle to revisit its previous feasible position [16]. Also, to promote diversity several randomly-generated (mutated) particles are added to the repository.

The maintenance of the global repository is a crucial issue. The size of the repository is defined as a free system parameter. Particles in the densely-populated areas have the priority to be removed when the repository's size exceeds a predefined value. The density of the search space is defined by a novel crowding factor according to which social leaders are easily determined by randomly selecting candidate solutions in the sparsely-populated areas. Finally, to enhance the local search ability, a gradient-based search method is applied to a small proportion of the global Pareto-optimal solutions when the size of the repository exceeds a user-defined threshold. The novel contributions of the proposed MOPSO are: (1) a new social leader selection mechanism called yet another Crowding Factor (YACF), and (2) an enhanced local search using the steepest descent method. These points are further described below.

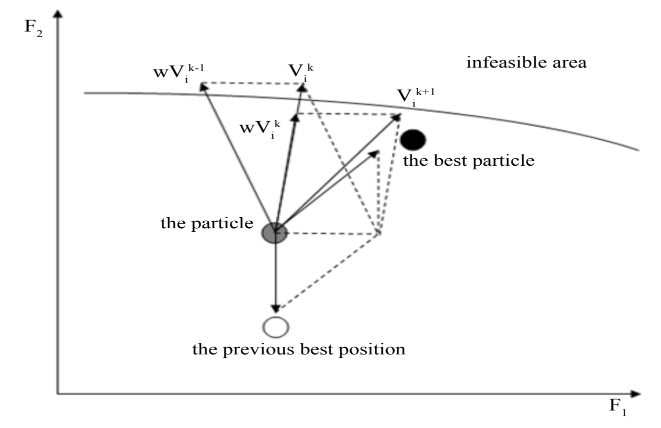

Figure 1: Fly-back mechanism.

A new crowding factor

Several social leader selection strategies based on the density measure of the population have been previously proposed [17,18]. The two most widely-used measures are the nearest-neighbor density estimator and the kernel density estimator.

The nearest neighbor density estimator quantifies how crowded the closest neighbors of a given particle are in the objective space [17]. As illustrated in Figure 2, this measure is estimated by the area of the largest cuboid formed by using the two nearest neighbors of particle *i* as the vertices [19].

The kernel density estimator provides a fitness-sharing mechanism achieved by degrading fitness values which are obtained by dividing the scaled fitness value of an individual by a quantity proportional to the number of individuals in the neighborhood [18]. Figure 3 shows the neighborhood or the niche which is defined in terms of a parameter called σ_{share} that indicates the radius of the neighborhood [13,18,19].

A new social leader selection method, called yet another Crowding Factor (YACF), was devised and implemented in this work. This proposed parameter-less sharing is an improvement over the two above-mentioned methods in that not only it provides the estimation of the density in the neighborhood, but it also provides a fitness sharing mechanism that degrades the fitness value of an individual solution with respect to a set of solutions in a similar circumstance. Before calculating the YACF value, a new sharing area in the objective space is calculated for the current generation. Each swarm particle is viewed as a hyper-circle whose center is a particle's objective vector and whose radius is the vector *V*:

$$V = \frac{\max(f(i,j)) - \min(f(i,j))}{N} \tag{5}$$

where *f*(*i*, *j*) is the *j*th dimension of the *i*th objective function, and *N* is the size of the global repository.

The YACF value for each particle is defined as the number of solutions in that particle's sharing area. Thus, the minimum crowding value of a particle is 1 because it only appears in its own sharing area.

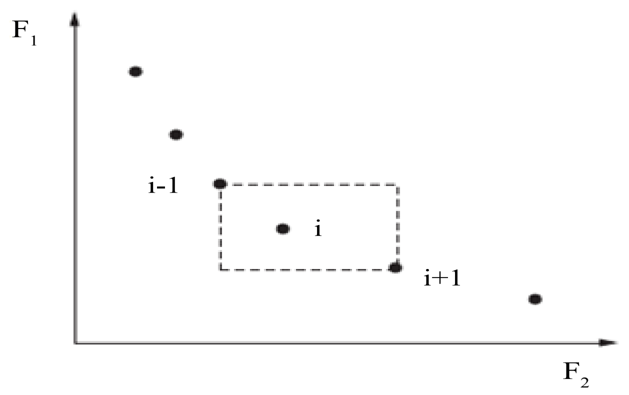

Figure 2: Nearest neighbor density estimator.

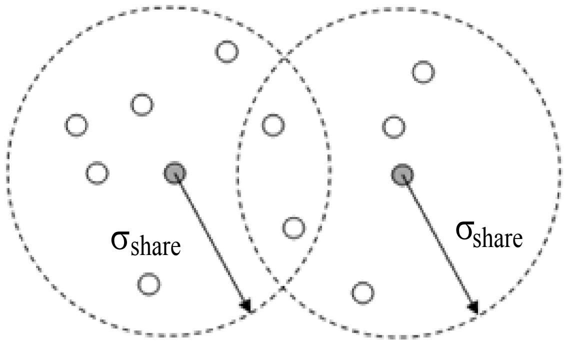

Figure 3: Kernel density estimator.

The Pareto-optimal solutions in the global repository can be divided into groups where members of each group share the same crowding value. Figure 4 demonstrates the basic idea behind YACF.

During the initial stage of social leader selection, 10% of the repository corresponding to the less-populated groups is identified and a leader among that group is then randomly selected.

Enhanced local search

The steepest gradient-descent search method enhances the local search ability of the MOPSO. It must be mentioned that the PSO's linearly-decreasing inertia weight is capable of performing limited and partial local search. This kind of partial local search is non-deterministic and difficult to use in order to measure the performance. In addition, it reduces the optimizer's ability to efficiently explore the solutions space. In order to force MOPSO to focus on the global search front, the steepest gradient-descent search method [20,21] is used to exploit the local area of each Pareto-optimal solution. Searching along the gradient-descent direction is a guarantee of obtaining local Pareto-optimal solutions which cannot be dominated by the original global Pareto-optimal solution because of the search direction along which an objective value is led toward its local optimum. Therefore, the gradient-descent search has a high probability of obtaining the global Pareto-optimal solutions among the local Pareto-optimal solutions.

The enhanced local search developed in this work can be described as follows; first, 5% of the global Pareto-optimal solutions in the less-populated areas are selected; and second, the steepest gradient-descent algorithm is applied to the selected solutions using Equation 6

$$X' = X - (s.\nabla F(X)) \tag{6}$$

where X is the design vector, s is the step size of gradient search for each particle, and $\nabla F(X)$ is the first-order derivative of the objective function.

In addition, instead of the line search a user-defined step size is applied that is doubled during the evolution. This approach greatly speeds up the algorithm and ensures that the same particles conducted the local search method search different areas. The stopping criterion is the maximum number of iterations, and usually 10 iterations is a common trade-off between effectiveness and efficiency.

Finally, Figure 5 depicts the new MOPSO, which utilizes the enhanced crowding factor and local search methods as described

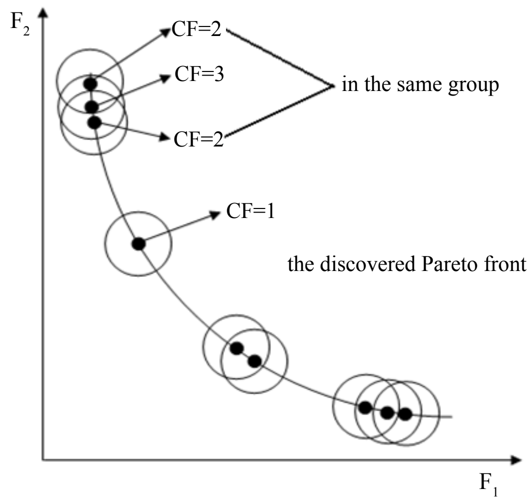

Figure 4: Crowding factor.

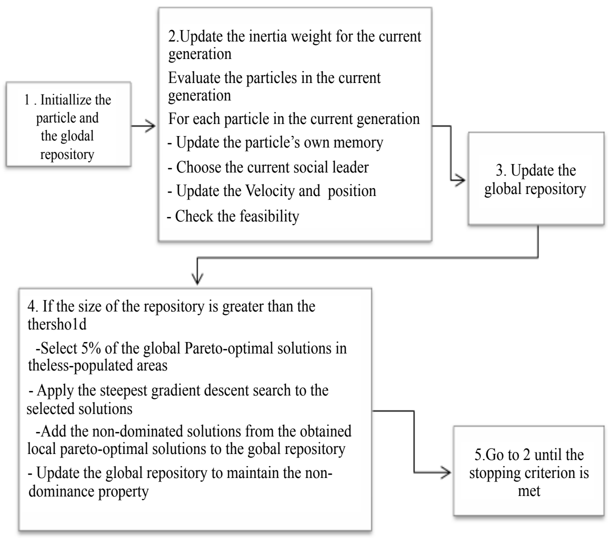

Figure 5: The enhanced MOPSO algorithm.

Citation: Ma L, Forouraghi B (2014) Multi-objective Particle Swarm Optimization with Gradient Descent Search. Int J Swarm Intel Evol Comput 4: 113. doi: 10.4172/2090-4908.1000113

above. A comparison between the basic MOPSO reveals that in Step 2, after particles' positions and velocities are updated, the fly-back mechanism is applied if any infeasible solutions were produced as the result of the update. Further, Step 4 shows that the steepest-decent is applied to the 5% of the repository population only if the size of the repository is greater than a user-define threshold value. All the newly generated non-dominated solutions are then added to and update the global repository.

Experiments and Results

This section presents the proposed multi-objective optimization method and compares it with several other MOPSO methods. All algorithms were thoroughly tested on a set of six benchmark functions. For the sake of fairness, for each problem the inertia weight, the number of particles and the number of generations were set to the same value.

Five different performance metrics were used to evaluate the overall performance of the proposed algorithm since no single performance metric can measure both the approach to the true Pareto front and the spread of the obtained Pareto-optimal solutions. Each performance metric was measured over 30 statistically independent runs for each test case. The true Pareto-front required for the experiments was obtained by the Monte Carlo simulation method applied to the area where the true Pareto-front was located. The size of the samples used in Monte Carlo simulation was approximately 10 million.

Performance metrics

A brief description of each performance metric is provided in this section. For the sake of brevity, the true Pareto-optimal set is denoted as PF_{true} and the discovered Pareto-optimal set as PF_{known} [12].

Generational Distance

This metric reports how far, on average, PF_{known} is from PF_{true}, and it is defined as:

$$GD = \frac{\sqrt[p]{\sum_{i=1}^{n} d_i^p}}{|\mathrm{PF}_{\mathrm{known}}|} \tag{7}$$

where p=2 and d_i is the Euclidean phenotypic distance between each member I of $\mathrm{PF}_{\mathrm{known}}$ and the closest member in $\mathrm{PF}_{\mathrm{true}}$ to that member. Observe that a successful swarm should discover solutions that produce a small value for GD [3,12].

Enhanced spacing

This metric is derived from the spacing metric which describes the spread of the Pareto-optimal solutions on the discovered Pareto front and measures the distance variance of neighboring solutions in PF_{known} [12]. The original version is phenotypic and is based on the real value of the objective functions. This may not reflect the real spread of the discovered Pareto-front when there is a great difference between the objective values. According to the enhanced spacing metric devised in this work, the problem mentioned above is eliminated by using the normalized objective values as shown below:

$$ESP = \sqrt{\frac{1}{|PF_{known}-1|}\sum_{i=1}^{|PF_{known}|}(\bar{d}-d_i)^2} \tag{8}$$

where,

$$d_i = \min_j \left(\sum_{k=1}^{n} \left| n_k^i(x) - n_k^j(x) \right| \right) \tag{9}$$

and

$$n_i^j = \frac{f_i^j - \min_k(f_k^j)}{\max_k(f_k^j) - \min_k(f_k^j)} \tag{10}$$

Similar to the regular spacing metric, an optimizer finding a smaller value of enhanced spacing is better able to identify a well-spread set of non-dominated solutions.

Generational non-dominated vector generation

This metric records how many global Pareto-optimal solutions during the evolution and is defined as [12].

$$GNVG = |PF_{known}| \tag{11}$$

Set coverage

This metric can be termed relative coverage comparison of two sets by mapping the order pair (X', X'') to the interval [1,0], and it is defined as [3,12].

$$SC(X', X'') = \frac{|\{a'' \in X''; \exists a' \in X' : a' \succeq a''\}|}{|X''|} \tag{12}$$

If all points in X' dominate and are equal to all points in X'', SC = 0; otherwise, $LSP = \frac{\quad}{N}$. When $X' = PF_{\mathrm{known}}$ and $X'' = PF_{\mathrm{true}}$, SC should be zero.

Local search performance: This measure reports how efficient the local search is, and it is defined as:

$$LSP = \frac{LPS}{N} \tag{13}$$

Where, LPS is the number of local Pareto-optimal solutions found by the local search during the evolution, and N is the times the local search was applied to the population. The local Pareto-optimal solution is defined as the solution which dominates the solution where the local search starts and cannot be dominated by any other solutions found by the local search at the current generation.

Tested MOPSO algorithms

In order to test the true mettle of the proposed MOPSO, a total of seven algorithms were examined as shown in Table 1. Algorithms PSO_6 and PSO_7 are essentially PSO_5 that use other advanced gradient-based methods, i.e., the BFGS (Broyden–Fletcher–Goldfarb–Shanno hill-climbing) optimization algorithm and simultaneous perturbation, respectively. These two were generated to objectively assess the performance of the steepest gradient-descent method used in PSO_5.The BFGS method is one of the most popular of quasi-Newton methods, and it has proven to be a promising method for solving nonlinear optimization problems. Unlike the Newton's method, quasi-Newtonian methods do not need to compute the Hessian matrix of the function to be minimized but rather approximate it using rank-one updates specified by gradient evaluations [11,21]. The simultaneous perturbation technique, on the other hand, is a stochastic gradient-descent method, which does not need the analytical form of the objective functions. Superior to the finite-difference method,

Algorithm	Description
PSO_1	The original version of MOPSO without any advanced additions.
PSO_2 and PSO_3	Successful variations to PSO_1 that have been applied to various multi-objective problems.
PSO_4	The proposed MOPSO in this work with YACF but without any local search ability.
PSO_5	PSO_4 utilizing steepest gradient-descent for local search.
PSO_6	PSO_4 utilizing the BFGS [11,20] method for local search.
PSO_7	PSO_4 utilizing simultaneous perturbation for local search.

Table 1: List of MOPSO algorithms used.

Citation: Ma L, Forouraghi B (2014) Multi-objective Particle Swarm Optimization with Gradient Descent Search. Int J Swarm Intel Evol Comput 4: 113. doi: 10.4172/2090-4908.1000113

simultaneous perturbation is a computationally efficient technique because it minimizes the number of function evaluations regardless of the number of decision variables [10,20].

PSO_2, which is called EM-MOPSO, uses an external repository and the nearest crowding distance metric [3,17]. Before calculating the crowding distance, a quick sort is applied to arrange the set of solutions in the ascending order of the objective function values, one at a time. The crowding distance value is the average distance of two neighboring solutions along the dimension of a specific objective function. The boundary solutions are always set to infinite values so that they are selected all the time. The final distance of a solution is the sum of the individual distance of each dimension. In addition, an efficient mutation strategy called elitist-mutation is incorporated in it to maintain the diversity of the algorithm. It replaces the infeasible solutions in the generation with the sparsely-populated solutions selected from the repository and mutates a predefined number of the replaced solutions.

In the PSO_3 approach the objective space is divided into a $d \times d \ldots \times d$ hypercube by dividing each objective function into d equal divisions [2]. A shared fitness value, which is defined as any number $x>1$ (usually x=10) divided by the number of Pareto-optimal solutions in the hypercube, is assigned to the hyper-cubes containing more than one Pareto-optimal solution. Then, the roulette-wheel selection is applied to choose the leader hypercube according to their shared fitness values. The social leader is randomly selected from the leader hypercube. Whenever the repository gets full, those particles located in the less-populated areas are given priority to stay in the repository.

Test cases

A total of 6 test cases, which are widely used in literature for testing various performance measures, were utilized.

Test Case 1: This problem was proposed in [22]. The objective is to minimize $F = (f_1, f_2)$ such that:

$$f_1 = x$$
$$f_2 = (1+10y)\times[1-(\frac{x}{1+10y})^{\alpha} - \frac{x}{1+10y}\sin(2q\pi x)] \qquad (14)$$

Where, $0 \le x, y \le 1, \alpha = 2, q = 4$.

The PF_{true} of F is disconnected and also has 4 Pareto curves.

Test Case 2: This problem was proposed in [23]. It requires minimization of $F = (f_1, f_2)$ such that:

$$f_1(\vec{x}) = 1-\exp(-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2)$$
$$f_2(\vec{x}) = 1-\exp(-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{n}})^2) \qquad (15)$$

Where, $-4 \le x_i \le 4, n = 3$.

The PF_{true} of F is a single concave Pareto curve, and the objective space of the PF_{true} is an area of the whole objective space.

Test Case 3: This problem was proposed in [24]. It involves minimization of $F = (f_1, f_2)$ such that:

$$f_1(x) = \begin{cases} -x & x \le 1 \\ -2+x & 1 < x \le 3 \\ 4-x & 3 < x \le 4 \\ -4+x & x > 4 \end{cases} \qquad (16)$$
$$f_2(x) = (x-5)^2$$

Where, $-5 \le x \le 10$.

The PF_{true} of F is disconnected with two curves.

Test Case 4: This problem was proposed by in [22]. It involves minimization of $F = (f_1, f_2)$ such that:

$$f_1(x_1, x_2) = x_1$$
$$f_2(x_1, x_2) = g(x_1, x_2) \times h(x_1, x_2)$$
$$g(x_1, x_2) = 11 + x_2^2 - 10\cos(2\pi x_2) \qquad (17)$$
$$h(x_1, x_2) = \begin{cases} 1-\sqrt{\dfrac{f_1(x_1, x_2)}{g(x_1, x_2)}} & f_1(x_1, x_2) \le g(x_1, x_2) \\ 0 & otherwise \end{cases}$$

Where, $0 \le x_1 \le 1, -30 \le x_2 \le 30$.

The PF_{true} of F is a single convex Pareto curve, and this problem has 60 local Pareto fronts to which the population could be easily attracted [2].

Test Case 5: This problem was cited in [12]. It involves minimization of $F = (f_1, f_2, f_3)$ such that:

$$f(x, y) = \frac{(x-2)^2}{2} + \frac{(y+1)^2}{13} + 3$$
$$f(x, y) = \frac{(x+y-3)^2}{36} + \frac{(-x+y+2)^2}{8} - 17$$
$$f(x, y) = \frac{(x+2y-1)^2}{175} + \frac{(2y-x)^2}{17} - 13 \qquad (18)$$

Where, $-400 \le x, y \le 400$.

The PF_{true} of F is a surface, and it is a connective region of the large 3-dimensional search space [16].

Test Case 6: This side-constrained problem was cited in [12]. It involves minimization of $F = (f_1, f_2)$ such that:

$$f_1(\vec{x}) = -[25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2]$$
$$f_2(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \qquad (19)$$

where,

$$x_1 + x_2 - 2 \ge 0$$
$$6 - x_1 - x_2 \ge 0$$
$$2 + x_1 - x_2 \ge 0$$
$$2 - x_1 + 3x_2 \ge 0$$
$$4 - (x_3-3)^2 - x_4 \ge 0$$
$$(x_5-3)^2 + x_6 - 4 \ge 0$$
$$0 \le x_1, x_2, x_6 \le 10, 1 \le x_3, x_5 \le 5, 0 \le x_4 \le 6$$

This is a difficult problem with six side constraints. The PF_{true} reflects six regions representing the intersection of specific constraints. Maintaining subpopulations at the different intersections is difficult.

Test Parameters

Common parameters used in each test case are listed in Table 2. Monte Carlo simulation was used to generate approximately 10 to 20 million feasible points among which PF_{true} was identified for each test case.

Discussion

The seven MOPSO algorithms were applied to the six test cases

Test Case	Number of articles	Number of Generations	Learning Factors (c_1, c_2)	Inertia Weight	Percent of Random Particles	Size of PF_{true}
1	50	60	(2.0,2.0)	0.9 to 0.3	0.0	10,000
2	30	40	(1.2,1.5)	0.4 to 0.1	0.0	2,000
3	30	40	(2.0,2.0)	0.9 to 0.3	0.0	16,000
4	50	60	(2.0,2.0)	0.4 to 0.2	0.0	10,000
5	70	100	(2.0,2.0)	0.9 to 0.3	0.0	10,000
6	100	120	(2.0,2.0)	0.9 to 0.4	0.1	450

Table 2: Used MOPSO parameters.

mentioned above. The first five test cases are unconstrained numeric multi-objective problems where as Test Case 6 is a difficult side-constrained problem. The results for each test are the average of 30statistically independent runs.

Initial experimentation indicated that Test Cases 1 and 3 have a disconnected Pareto front. In contrast to PSO_1-PSO_3, PSO_4-PSO_7 exhibited a better ability to identify the disconnected curves. It was determined that PSO_5 and PSO_7 are capable of successfully finding Pareto-optimal solutions along the Pareto front. For Test Case 5, which has a large 3-dimensional search space, PSO_5 achieved the best results due to its enhanced global and local search abilities. Although most of the algorithms achieved comparable results, PSO_5 and PSO_7 produced better results. As a whole, experiments indicated that the proposed MOPSO is better able to efficiently and consistently search along the Pareto front and obtain a uniformly-distributed Pareto front. Moreover, a thorough statistical analysis was conducted to verify that the proposed MOPSO is both effective and efficient due to YACF and the constraint-handling strategy which guide a swarm to converge closely to the true Pareto front.

Table 3 through 6 depict the overall performance of the seven algorithms by performing statistical analysis on the observed values of the generational distance, enhanced spacing, set coverage and generational non-dominated vector generation. As shown in Table 3, PSO_4-PSO_7 performed better than other algorithms over 30 statistically independent random experiments. This demonstrates that YACF and the constraint-handling strategy (fly-back mechanism) are capable of guiding a swarm to converge closely to the true Pareto front. Amongst PSO_4-PSO_7, PSO_5 is the leader in optimizing Test Cases 1-4, and it also has the least fluctuation during the experiments. Further, PSO_7 has the best performance on Test Cases 5 and 6.

Table 4 shows that PSO_4-PSO_7 out performed other methods based on the ES metric over 30 random trials. As it turns out, the proposed MOPSO can maintain adequate distribution and diversity along the discovered Pareto front. PSO_7 has comparable performance because of its random perturbation behavior, and it outperformed other local search methods on Test Cases 3 and 5.

Considering the set coverage performance metric, Table 5 shows that PSO_4 was the best performer on Test Cases 1, 2, and 5 while algorithms PSO_1-PSO_3 marginally outperformed PSO_4 on Test Cases 3 and 4. For Test Case 6, no algorithm was successful. In addition, PSO_5-PSO_7 performed almost the same on the entire set of six test cases. This is due to the fact that a small step size of the gradient-based search can find Pareto-optimal solutions in a small range and that these solutions may dominate the solutions on the estimated true Pareto front generated by Monte Carlo simulation. Nevertheless, PSO_4 out performed PSO_1-PSO_3.

Considering the generational non-dominated vector generation metric, Table 6 shows the number of discovered Pareto-optimal solutions during the 30 trials. Note that in PSO_2-PSO_7, the global repository containing the discovered Pareto-optimal solutions had a threshold of 500. According to the results, PSO_5 is superior to other algorithms based on the average number of discovered Pareto-optimal solutions and the exhibited fluctuation during the experiments. Based on the minimum value of GNVG, PSO_1, PSO_3 and PSO_7 failed in at least one trial of Test Case 4 because they were all trapped in local optima.

To measure the local search performance parameter amongst PSO_5-PSO_7, the number of local Pareto-optimal solutions is reported in Table 7. With the exception of the local search of PSO_6 which failed on Test Case 6, all the local search methods were effective with varying degrees although PSO_5 was the best considering the average number of the discovered local Pareto-optimal solutions.

An engineering design example

This section presents the optimization results of applying the proposed MOPSO algorithm to the design of a welded beam as seen in Figure 6.

The beam *A* is to be welded to a rigid support member *B*. The welded beam consists of 1010 steel and is to support a force *F* of 6000 *lb* [25]. The problem has 4 design parameters: the thickness of the beam (b), the width of the beam (t), the length of the weld (l) and the thickness of the weld (h).

The goal of the optimization is to identify optimal dimensions of the beam $\vec{x} = (h, l, b, t)$ such that the fabrication cost (f_1) and the end deflection (f_2) are both minimized. The problem can, therefore, be mathematically stated as follows [3,26]:

Minimize $F = (f_1, f_2)$ such that:

$$f_1(\vec{x}) = 1.10471 \times h^2 l + 0.04811 \times tb(14.0 + l) \qquad (20)$$

$$f_2(\vec{x}) = \frac{2.1952}{t^3 b}$$

Subject to the following inequality constraints involving different stress, buckling and logical limitations:

$$g_1(\vec{x}) = 13,600 - \tau(\vec{x}) \ge 0$$
$$g_2(\vec{x}) = 30000 - \sigma(\vec{x}) \ge 0$$
$$g_3(\vec{x}) = b - h \ge 0$$
$$g_4(\vec{x}) = P_c(\vec{x}) - 6000 \ge 0$$

Where,

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + (\tau'')^2 + \frac{l\tau'\tau''}{\sqrt{0.25[l^2 + (h+t)^2]}}} \qquad (21)$$

$$\tau' = \frac{6000}{\sqrt{2}hl}$$

$$\tau'' = \frac{6000 \times (14 + 0.5l)\sqrt{0.25[l^2 + (h+t)^2]}}{2 \times \{0.707hl[\frac{l^2}{12} + 0.25(h+t)^2]\}}$$

$$\sigma(\vec{x}) = \frac{504000}{t^2 b}$$

$$P_c(\vec{x}) = 64746.022 \times (1 - 0.0282346t)tb^3$$

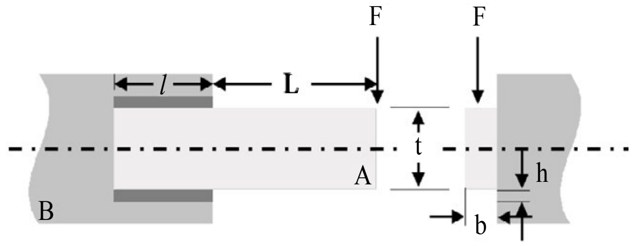

Figure 6: The welded beam example.

The boundaries of $\vec{x}$ is taken as $0.125 \le h,b \le 5.0, \quad 0.1 \le l,t \le 10.0$.

Clearly, the two objectives f_1 and f_2 are conflicting in nature since minimization of the fabrication cost, which is a function of the volume of the beam, requires the smallest beam dimensions while minimization of the deflection requires the largest beam dimensions.

The proposed MOPSO together with other MOPSO algorithms were applied using100 particles over 100 generations, learning factors of 0.5 (c_1) and 1.0 (c_2),and 10% random particles for mutation.

The cost-deflection graph for the design problem, as shown in

Test 1	Statistics	Generational Distance						
		PSO_1	PSO_2	PSO_3	PSO_4	PSO_5	PSO_6	PSO_7
	Avg	1.39E-04	6.38E-05	5.48E-05	2.94E-05	1.67E-05	2.24E-05	2.50E-05
	Std	6.58E-05	3.77E-05	3.07E-05	4.26E-06	1.63E-06	2.81E-06	3.37E-06
2	Avg	2.20E-04	1.96E-04	1.59E-04	1.50E-04	6.64E-05	1.44E-04	6.70E-05
	Std	1.24E-04	4.45E-05	1.61E-05	1.61E-05	1.60E-05	3.79E-05	2.11E-05
3	Avg	2.32E-04	3.41E-04	1.84E-04	1.85E-04	6.44E-05	1.36E-04	6.45E-05
	Std	4.55E-05	1.83E-04	2.43E-05	2.55E-05	1.04E-06	2.38E-05	1.45E-06
4	Avg	6.43E-04	3.25E-04	6.05E-04	2.51E-05	1.25E-05	2.05E-05	2.80E-04
	Std	0.002135	3.69E-04	0.002142	3.33E-06	5.18E-07	3.44E-06	0.001075
5	Avg	9.31E-05	9.68E-05	9.46E-05	1.07E-04	9.58E-05	9.07E-05	9.17E-05
	Std	1.46E-05	1.31E-05	1.46E-05	1.16E-05	1.22E-05	1.27E-05	8.43E-06
6	Avg	1.484679	1.486398	1.546686	1.777849	1.823246	1.729634	1.777121
	Std	0.223294	0.165193	0.217658	0.255305	0.233153	0.26947	0.238666

Table 3: Generational distance (GD).

Test1	Statistics	Enhanced Spacing						
		PSO_1	PSO_2	PSO_3	PSO_4	PSO_5	PSO_6	PSO_7
	Avg	0.003454	0.00218	0.002049	0.001439	0.001147	0.001259	0.001328
	Std	0.001335	4.60E-04	5.18E-04	1.14E-04	6.31E-05	8.97E-05	9.51E-05
2	Avg	0.007135	0.007196	0.005888	0.005554	0.001392	0.003337	0.001667
	Std	0.001981	9.63E-04	5.64E-04	6.64E-04	2.27E-04	6.00E-04	4.15E-04
3	Avg	0.002916	0.003735	0.002549	0.002283	0.001553	0.001671	0.001546
	Std	4.17E-04	6.64E-04	3.82E-04	2.35E-04	3.56E-05	1.26E-04	4.12E-05
4	Avg	0.003257	0.007184	0.002654	0.002414	0.001381	0.002132	0.027963
	Std	0.001782	0.002088	9.36E-04	4.28E-04	1.56E-04	3.25E-04	0.134779
5	Avg	0.008158	0.007847	0.008115	0.005203	0.002054	0.001706	0.00167
	Std	0.002912	0.003164	0.002783	0.001263	6.17E-04	4.34E-04	6.02E-04
6	Avg	0.032146	0.029944	0.028702	0.034487	0.030356	0.031723	0.03357
	Std	0.010274	0.00704	0.009623	0.009983	0.006765	0.00702	0.010338

Table 4: Enhanced spacing (ES).

Test 1	Statistics	Set Coverage						
		PSO_1	PSO_2	PSO_3	PSO_4	PSO_5	PSO_6	PSO_7
	Avg	0.072065	0.075471	0.066215	0.066209	0.066304	0.066046	0.065698
	Std	0.002408	0.003067	0.00127	9.61E-04	8.79E-04	0.001106	0.001164
2	Avg	0.005828	0.00413	0.004685	0.003803	0.061149	0.011949	0.057427
	Std	0.002653	0.002163	0.001773	0.001628	0.017118	0.00507	0.016494
3	Avg	0.010807	0.012233	0.011045	0.014017	0.018941	0.018296	0.019044
	Std	0.001087	0.002452	0.001018	0.001105	8.45E-04	0.001037	8.98E-04
4	Avg	0.031772	0.013197	0.026811	0.024113	0.047911	0.029159	0.028877
	Std	0.009409	0.002485	0.00752	0.003586	0.004352	0.004166	0.008375
5	Avg	0.002345	0.002027	0.002289	0.001446	0.003863	0.004536	0.004667
	Std	5.43E-04	6.18E-04	4.08E-04	4.10E-04	5.33E-04	6.01E-04	5.70E-04

Table 5: Set coverage

	Statistics	Generational Non-dominated Vector Generation						
		PSO_1	PSO_2	PSO_3	PSO_4	PSO_5	PSO_6	PSO_7
Test 1	Avg	551.6	500	500	500	500	500	500
	Std	16.20206	0	0	0	0	0	0
2	Avg	164.9333	144.2333	157.9667	149.7	500	296.4333	498.3333
	Std	11.83479	9.728595	11.34161	9.352896	0	33.01584	8.096639
3	Avg	285.3333	279.4667	315	360.7667	500	484.8333	500
	Std	18.68392	28.31929	32.2604	18.84265	0	18.38493	0
4	Avg	501.8667	272.0333	461.5667	457.7	500	491.3667	465.2667
	Std	139.7831	30.2087	123.905	33.22263	0	27.13114	121.7393
5	Avg	477.4333	420.9	462.9333	282.1333	500	500	500
	Std	52.63059	40.48815	33.97051	28.67023	0	0	0
6	Avg	40.46667	41.5	39.73333	37.5	66.13333	36.1	47.2
	Std	4.849284	4.356987	3.915212	3.658324	44.15256	4.174127	25.79457

Table 6: Generational non-dominated vector generation.

Test 1	Statistics	Local Search Performance		
		PSO_5	PSO_6	PSO_7
	Avg	86.32313	8.60245	39.2479
	Std	3.249478	0.61694	3.384728
2	Avg	181.4407	66.74474	162.4159
	Std	11.47538	5.834797	16.86514
3	Avg	222.4908	41.82584	220.2103
	Std	2.219962	2.653147	3.833463
4	Avg	129.1604	39.31602	80.37065
	Std	6.586379	5.62699	22.5745
5	Avg	214.2054	181.4904	207.3027
	Std	4.302982	6.490768	3.969521
6	Avg	28.83333	3.166667	14.95
	Std	24.27905	5.293602	21.05087

Table 7: Local search performance.

Figure 7, depicts the distribution of the non-dominated solutions found by the proposed MOPSO algorithm (PSO_5). Figure 7 demonstrates that not only PSO_5 can provide a wide distribution of solutions it can also efficiently explore local areas along the Pareto front due to its enhanced local search ability.

Table 8 shows Pareto optimal design solutions obtained by each of the seven algorithms.

It must be noted that the best reported optimized solution for x^* = (h, l, b, t) is (0.2489, 6.1730, 0.2533, 8.1789) with F= (2.4331, 0.0158) [2]. PSO_5, on the other hand, was able to identify the best optimal solution x^* = (0.2439, 6.2356, 0.2443, 8.2976) with F= (2.3838, 0.01572) among all 7 MOPSO methods.

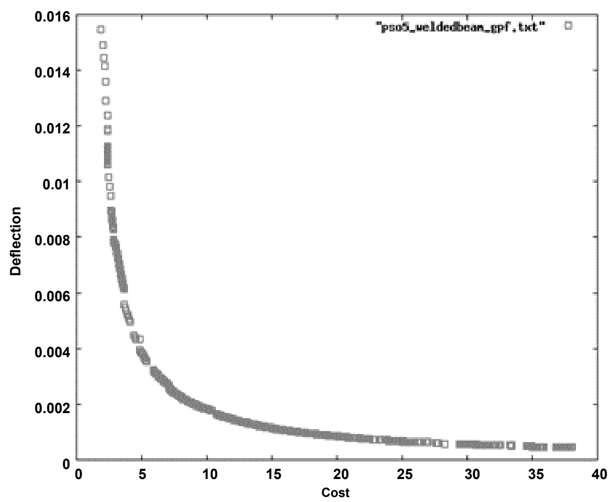

Figure 7: Discovered Pareto fronts for the welded beam problem.

Conclusion

This paper introduced a novel multi-objective particle swarm optimizer with a new crowding factor and enhanced local search ability [27,28]. Performing local search using gradient-based approaches such as the steepest gradient-descent method, the quasi-Newtonian method, and simultaneous perturbation all help MOPSO to focus on global search in order to efficiently solve multi-objective problems. In addition, the newly-developed crowding factor was shown to guide an entire swarm toward the true Pareto front with a good distribution and diversity along the discovered Pareto front.

To highlight the performance of the proposed MOPSO algorithm, six numerical benchmark problems as well as a difficult engineering design problem were attempted and the discovered Pareto front for each case was carefully examined. The experimental results demonstrated that the proposed MOPSO with the steepest gradient-descent method outperformed other MOPSO methods in a variety of canonical problems as well as a real-world design case. Further, it was demonstrated that the enhanced local search was capable of accelerating MOPSO by reducing the population size and shortening the evolution cycle. In other words, when PSO_1-PSO_4 performed as well as PSO_5 or PSO_6, they all required larger number of particles and longer times to converge. Applying a stochastic gradient search such as simultaneous perturbation was also shown to be a trade-off when the analytical form of the multi-objective problem is difficult to obtain.

References

1. Cheng S, Chen MY, Hu G (2013) An Approach for Diversity and Convergence Improvement of Multi-Objective Particle Swarm Optimization. Advances in Intelligent Systems and Computing 212: 495-503.
2. Coello Coello CA (2011) An Introduction to Multi-Objective Particle Swarm Optimizers. Advances in Intelligent and Soft Computing 96: 3-12.
3. Reddy MJ, Kumar DN (2007) An efficient multi-objective optimization algorithm based on Swarm intelligence for engineering design. Engineering Optimization 39: 49-68.
4. Rudolph G, Trautmann H, Sengupta S, Schütze O (2013) Evenly Spaced Pareto Front Approximations for Tricriteria Problems Based on Triangulation. Lecture Notes in Computer Science 7811: 443-458.
5. Shim M,Suh M (2001) Pareto-based continuous evolutionary algorithms for

PSO	h	l	b	t	Cost	Deflection
PSO_1	0.199691	6.518168	0.253676	9.999899	2.791220	0.008654
PSO_2	0.240218	5.785878	0.240467	8.944069	2.416130	0.012759
PSO_3	0.233615	6.278582	0.236438	9.999210	2.685051	0.009287
PSO_4	0.243526	6.533835	0.245912	8.271171	2.437401	0.015776
PSO_5	0.243976	6.235635	0.244342	8.297646	2.383850	0.015726
PSO_6	0.237789	5.486233	0.239379	9.438643	2.460859	0.010906
PSO_7	0.244290	6.228275	0.244843	8.284974	2.384730	0.015766

Table 8: Optimal design solutions.

multi objective optimization. Engineering Computation 19: 22-48.

6. Kennedy J, Eberhart RC, Shi YH (2001) Swarm Intelligence. Morgan Kaufmann, San Mateo, CA.
7. Shi YH, Eberhart R (1999) Empirical study of particle swarm optimization. IEEE Congress on Evolutionary Computation 1945-1950.
8. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 3: 257-271.
9. Ho SL, Yang S, Ni G, Lo EW, Wong HC (2005) A particle swarm optimization-based method for multiobjective design optimizations. IEEE Transactions on Magnetics 41: 1756-1759.
10. Maeda Y, Matsushita N, Miyoshi S, Hikawa H (2009) On Simultaneous Perturbation Particle Swarm Optimization. IEEE Congress on Evolutionary Computation 3271-3276.
11. Ono S, Nakayama S (2009) Multi-objective particle swarm optimization for robust optimization and its hybridization with gradient search. IEEE Congress on Evolutionary Computation 1629-1636.
12. CoelloCoello CA, Lamont GB, Van Veldhuizen DA (2007) Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, New York.
13. Forouraghi B (2009) Optimal tolerance allocation using a multiobjective particle swarm optimizer. International Journal of Advanced Manufacturing Technology 44: 710-724.
14. Ochlak E, Forouraghi B (2006) A Particle Swarm Algorithm for Multiobjective Design Optimization. Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 06) 765-772.
15. Li X (2004) Better Spread and Convergence: Particle Swarm Multiobjective Optimization Using the Maximum Fitness Function. Lecture Notes in Computer Science 3102: 117-128.
16. He S, Prempain E, Wu QH (2004) An improved particle swarm optimizer for mechanical design optimization problems. Engineering Optimization 36: 585-605.
17. Deb K, Pratap A, Agarwal S and Meyarivan T (2002) A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6: 182-197.
18. Goldberg DE, Richardson J (1987) Genetic algorithms with sharing for multimodal function optimization. Proceedings of the Second International Conference on Genetic Algorithms 41–49.
19. Reyes-Sierra M, Coello Coello CA (2006) A Survey of the State-of-the-Art Multi-Objective Particle Swarm Optimizers. International Journal of Computational Intelligence Research 2: 287–308.
20. Gosavi A (2003) Simulation-based optimization: parametric optimization techniques and Reinforcement learning. Springer, New York.
21. Haftka RT, Gürdal Z (1992) Elements of Structural Optimization. Springer, New York.
22. Deb K (2012) Optimization for Engineering Design: Algorithms and Examples. Prentice Hall, New Delhi, India.
23. Fonseca CM, Fleming PJ (1993) Genetic algorithms for multiobjective optimization formulation, discussion and generalization. Proceedings of the fifth International Conference on Genetic Algorithms 416-423.
24. Schaffer JD (1984) Multiple objective optimization with vector evaluated genetic algorithms. Ph.D. Thesis, Vanderbilt University.
25. Ravindran A, Ragsdell KM, Reklaitis GV (2006) Engineering Optimization-Methods and Applications. John Wiley & Sons, Hoboken, New Jersey.
26. Deb K (2014) Multi-Objective Optimization. Search Methodologies: 403-449.
27. Deb K, Goldberg DE (1989). An investigation of niche and species formation in genetic function optimization. Proceedings of the Third International Conference on Genetic Algorithms 42–50.
28. Shi YH, Eberhart RC (2001) Fuzzy adaptive particle swarm optimization. IEEE Congress on Evolutionary Computation 101-106.

Citation: Ma L, Forouraghi B (2014) Multi-objective Particle Swarm Optimization with Gradient Descent Search. Int J Swarm Intel Evol Comput 4: 113. doi: 10.4172/2090-4908.1000113