Soft Comput (2013) 17:939–952
DOI 10.1007/s00500-012-0969-3

FOCUS

A multi-objective artificial physics optimization algorithm based on ranks of individuals

Yan Wang · Jian-chao Zeng

Published online: 22 February 2013

Abstract This paper proposes a multi-objective artificial physics optimization algorithm based on individuals' ranks. Using a Pareto sorting based technique and incorporating the concept of neighborhood crowding degree, evolutionary individuals in the search space are evaluated at first. Then each individual is assigned a unique serial number in terms of its performance, which affects the mass of the individual. Thereby, the population evolves towards the direction of the Pareto-optimal front. Synchronously, the presented approach has good diversity, such that the population is spread evenly on the Pareto front. Results of simulation on a number of difficult test problems show that the proposed algorithm, with less evolutionary generations, is able to find a better spread of solutions and better convergence near the true Pareto-optimal front compared to classical multi-objective evolutionary algorithms (NSGA, SPEA, MOPSO) and to simple multi-objective artificial physics optimization algorithm.

Keywords Multi-objective optimization · Rank · Artificial physics optimization · Neighborhood · Crowding degree

Communicated by A-A. Tantar.

Y. Wang (✉) · J. Zeng
Complex System and Computational Intelligence Laboratory, Taiyuan University of Science and Technology, Taiyuan 030024, People's Republic of China
e-mail: wy75910@gmail.com; wy75910@163.com

1 Introduction

In many real-world search and optimization tasks, it is very common to face problems having two or more objectives that are normally conflict with each other and yet need to be optimized simultaneously. Such problems are called "multi-objective" problems. Due to the multi-criteria nature of multi-objective optimization problems, "optimality" of a solution has to be redefined, giving rise to the concept of Pareto optimality where "trade-off" solutions representing the best possible compromises among the objectives are sought, rather than seeking optimization of a single objective. Traditional approaches for generating such solutions normally aggregate all objectives and then optimize the single composite objective with mathematical programming techniques. However, they have drawbacks of low effectiveness and are sensitive to the order of weights or objectives. Population-based optimal algorithms have the ability to handle a set of solutions in a simultaneous manner and can deal with problems of different types. These characteristics are very suitable for multi-objective optimization problems. Since Schaffer implemented the first multi-objective evolutionary algorithm (vector evaluated genetic algorithm, VEGA) in the mid-eighties of the last century, more and more population-based optimal paradigms have been introduced into multi-objective optimization area. Among these, the multi-objective evolutionary algorithms (MOEAs) and the multi-objective particle swarm optimization (MOPSO) algorithms have become increasingly popular over the last decade.

Recently, there has been an increased interest in the study of MOEAs. Well known MOEAs include Multi-Objective Genetic Algorithm (MOGA) (Fonseca and Fleming 1993), Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas and Deb 1994) and its extension

(NSGA-II) (Deb et al. 2000), Niched-Pareto Genetic Algorithm (NPGA) (Horn et al. 1994), Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler and Thiele 1999) and its improved algorithm (SPEA2) (Zitzler and Thiele 2002), Pareto Archive Evolution Strategy (PAES) (Knowles and Corne 1999) and Pareto Envelope-based Selection Algorithm (PESA) (Corne et al. 2000), etc. The performance of most MOEAs has been assessed using benchmark problems; several of them have shown good performance. Characteristics of typical MOEAs, showing successful applications, can be summed up as follows:

1. Pareto-dominated relationship is used to evaluate fitness values of individuals;
2. Elitist mechanism is incorporated through the use of an archive containing non-dominated solutions previously found; and
3. Archive set is pruned using certain strategies when the number of non-dominated solutions is greater than its size.

At the same time, there are some drawbacks of MOEAs:

1. They have high computational costs though they maintain good diversity of population; and
2. Some approaches strongly depend on corresponding parameters of algorithms, which can usually be adjusted with problem knowledge that may not be available.

In order to solve the problems mentioned above, researchers have proposed many improved MOEAs (Fernández et al. 2010; Antonio López et al. 2010; Molina et al. 2009; Fabre et al. 2009, 2010; Martínez and Carlos 2010). On the other hand, they are finding other meta-heuristics that can be incorporated into multi-objective optimization. Particle Swarm Optimization (PSO) seems particularly suitable for multi-objective optimization mainly because of the high speed of convergence for single-objective optimization (Zeng et al. 2004). The typical MOPSO algorithms include (Coello et al. 2002) and its upgrade (Carlos 2004; Hu et al. 2002) based on dynamic neighborhood; Fieldsend and Singh (2002) based on elitist mechanism and the perturbation factor; Sierra and Coello (2005) based on crowding distance and ε-dominated (Abido et al. 2007). These MOPSO algorithms mainly adopt successful experiences of MOEAs indicated above to deal with multi-objective problems. However, PSO algorithms have drawbacks such as premature convergence, and the tendency to easily get trapped in local best, which are inevitable in solving multi-objective problems. In some improved MOPSO algorithms (Durillo et al. 2009; Nebro et al. 2009; Toscano-Pulido et al. 2007; Santana-Quintero et al. 2006; Nedjah 2010), other algorithms or mutation strategies are adopted to address this problem, which normally adds complexity and computational cost to algorithms.

Artificial physics optimization (APO) is a novel stochastic population-based optimization algorithm for global optimization problems (Xie and Zeng 2009, 2010; Xie et al. 2009, 2010). Because of its high performance in terms of high speed of convergence and good diversity, we have introduced it in multi-objective optimization area (Wang and Zeng 2010; Wang et al. 2011). As expected, MOAPO algorithm has a better performance compared with some classical MOEAs and MOPSO algorithm with respect of several benchmark functions, especially in terms of addressing population diversity. When APO algorithm is applied in multi-objective optimization, mass dealing is a key technique. Mass is an important parameter of the APO algorithm. Responding to virtual forces, an individual in APO moves toward other particles with larger "masses" (better fitness values) and away from lower mass particles (worse fitness values). In order to differentiate from the algorithm proposed in this paper, we call our original algorithms (Wang and Zeng 2010; Wang et al. 2010, 2011) the simple MOAPO algorithm (SMOAPO). We adopt aggregate functions to transform multiple objectives into a single problem so that we can calculate the mass of each individual easily; the mass obtained by this method can represent the performance of the individual to some extent. However, it does not embody the characteristics of MOPs sufficiently. Thus, we present a rank-based MOAPO (RMOAPO) algorithm to solve this problem. In this algorithm, we deal with the mass function by assigning different ranks to individuals by evaluating the Pareto dominant relationships between individuals and their crowding degree.

The remainder of this paper is organized as follows: Section 2 introduces some basic concepts to make this paper self-contained. In Sect. 3 we provide a brief description of the APO algorithm. Section 4 presents our approach of RMOAPO. Section 5 describes the simulation test and its analysis. Finally, conclusions are derived, and recommendations made for further research.

2 Basic concept in multi-objective optimization problems

Without loss of generality, a minimized constraint multi-objective problem can be defined as follows:

$$\begin{aligned} \text{Min} \quad & f(X) := [f_1(X), f_2(X), \ldots, f_k(X)] \\ s.t. \quad & g_i(X) \le 0, \quad i = 1, 2, \ldots, m \\ & h_j(X) = 0, \quad j = 1, 2, \ldots, p \end{aligned}$$

where $X \in R^n$ is the vector of the decision variables, $f_i(X)$, $i = 1, 2, \ldots, k$ is the objective function, $g_i(X) \le 0$,

$i = 1, 2, \ldots, m$ and $h_j(X) = 0, j = 1, 2, \ldots, p$ are inequality and equality constraints, respectively.

In the case of multiple objectives, there may not exist one solution which is the best (global minimum or maximum) with respect to all objectives. Consequently, we normally look for "trade-offs", rather than a single solution when dealing with multi-objective optimization problems. The notion of "optimality" is, therefore, different. The most commonly adopted notion of optimality in multi-objective optimization problems is Pareto optimality. In order to understand Pareto optimality more easily, another important concept of "dominance relation" should be introduced.

Definition 1 (*Dominance relation*) For two arbitrary solutions $p, q \in R^n$, p is said to dominate q (denoted as $p \succ q$) if it is better than or equal to q on all objectives (i.e., $\forall i \in \{1, \ldots, k\}: p_i \leq q_i$) and at least better than q for one objective (i.e., $\exists j \in \{1, \ldots, k\}: p_j < q_j$), where k is the number of sub-objectives. Here p is non-dominated while q is dominated and "$\succ$" denotes a dominance relationship.

Definition 2 (*Pareto optimal set*) The Pareto-optimal set (denoted as P^*) is the set of all possible Pareto-optimal solutions (i.e., $P^* = \{x^* \in X | \neg\exists x \in X \text{ and } f(x) \succ f(x^*)\}$).

The set of optimal solutions in the decision space X is in general denoted as the Pareto set, and we will denote its image in the objective space as the Pareto front (denoted as PF^*).

It is obvious that the target of solving MOPs is to find a set of solutions, which are as close as possible to the Pareto optimal set.

3 Artificial physics optimization algorithm

Artificial physics was originally proposed by Spear et al. (2005) to solve distributed control in robot systems. In the basic AP framework, the robots are treated as physical individuals possessing a position, mass, velocity and momentum. Their motion is controlled by the second Newton's force law $F = ma$. Virtual forces created drive the individuals to move, just as real masses move in response to an externally applied force. The continuous moving of an individual in search space is described as displacement ΔX in some little discrete time-slice Δt. That is, $\Delta X = V \cdot \Delta t$. Here, the velocity variable is decided by $\Delta V = (F/m) \cdot \Delta t$, where F is the combination force exerted by other individuals. Hence, the velocity of an individual at moment t is $V(t) = V(t-1) + (F/m) \cdot \Delta t$ and its position at moment t is $X(t) = X(t-1) + V(t) \cdot \Delta t$. Parameter $F_{\max}$ is used to restrict the maximum force exerted on the individual, which restricts the maximum acceleration. Similarly, individual velocity is restricted by $V_{\max}$.

Then Xie and Zeng studied the original artificial physics and tried to simulate the emergence of swarm intelligence from the viewpoint of artificial physics. They successfully introduced artificial physics into optimization and proposed the original artificial physics optimization (APO) algorithms (Xie and Zeng 2009, 2010; Xie et al. 2009, 2010). In these algorithms, each entity is treated as a physical individual with attributes of mass, position and velocity. The relationship between an individual's mass and its fitness (to be optimized) is constructed. The better the objective function value, the bigger is the mass, and the higher is the magnitude of attraction. The individuals move towards the better fitness region, which can be mapped to individuals moving towards others with bigger masses. Because the virtual forces drive each individual motion, the bigger mass determines the higher magnitude of attraction. In addition, the individual attracts ones with worse fitness while repels those with better fitness. Especially, the individual with the best fitness attracts all others, whereas it is never repelled or attracted by others. The attractive-repulsive rule can be treated as the search strategy in the optimization algorithm used to lead the population to search the better fitness region of the problem.

In APO algorithm, the mass function of individual i, the force exerted on individual i via individual j and the total virtual force exerted on individual i via all other individuals in kth dimension are calculated by Eqs. (1)–(3).

$$m_i = e^{\frac{f(x_{\text{best}}) - f(x_i)}{f(x_{\text{worst}}) - f(x_{\text{best}})}}, \quad \forall i \tag{1}$$

$$F_{ij,k} = \begin{cases} Gm_i m_j (x_{j,k} - x_{i,k}), (f(x_i) > f(x_j)) \\ -Gm_i m_j (x_{j,k} - x_{i,k}), (f(x_i) \leq f(x_j)) \end{cases} \tag{2}$$
$$\forall i \neq j \text{ and } i \neq \text{best}$$

$$F_{i,k} = \sum_{\substack{j=1 \\ i \neq j}}^{n} F_{ij,k} \quad \forall i \neq \text{best} \tag{3}$$

where $f(x_{\text{best}})$ denotes the function value of the best individual, $f(x_{\text{worst}})$ denotes the function value of the worst individual and $f(x_i)$ denotes the function value of individual i. Here G is the "gravitational constant". The distance from individual i to individual j in kth dimension is denoted by $x_{j,k} - x_{i,k}$ and $F_{ii,k} = 0$.

After calculating the total force, velocity and position of individual i at generation $t + 1$ are updated by Eqs. (4) and (5), respectively.

$$v_{i,k}(t+1) = wv_{i,k}(t) + \lambda F_{i,k}/m_i \tag{4}$$

$$x_{i,k}(t+1) = x_{i,k}(t) + v_{i,k}(t+1) \tag{5}$$

where $v_{i,k}(t)$ and $x_{i,k}(t)$ are velocity and position, respectively, of individual i in kth dimension at tth generation. λ is a random variable generated within (0, 1) with normal distribution. w is an inertia weight within (0, 1). Movement of each individual is restricted within the domains of $Xi \in [X\min, X\max]$ and $Vi \in [V\min, V\max]$.

From Eqs. (4) and (5), we can find that the evolutionary equation of the APO algorithm is very similar with that of PSO algorithm. There are some analogous characteristics common between these two algorithms. Firstly, both are population-based algorithms. Secondly, both work by iterative updating of velocity and position, and their actions are discrete. Finally, individuals in both algorithms have integrated space position information; in other words, the individual has global orientation. The difference between them is as follows: At first, the individual of APO algorithm has the attribute of mass, while the particle of PSO is without mass. Then there is no personal best value in APO algorithm, whereas in PSO algorithm, particle action is only relevant to particle personal best value and global best value.

In PSO algorithm, the motion of each particle is guided by its own historical best position and the best particle in the population. So the whole population has the ability of moving to a better search region. However, when a particle falls in an acceptable local best solution, it will likely attract other particles to fall into this local best value rather than explore the remaining unexplored solution space. This leads to a limited search space. As a result, the algorithm yields bad diversity and easily falls into local best value.

From the attractive-repulsive rule mentioned above, we know that an individual in APO algorithm is attracted not only by the global best individual, but also by other individuals in the population. If we consider this attractive-repulsive force as interaction between individuals or cognition and social learning of the individual, information obtained by the individual in APO algorithm is more comprehensive. Thus, the APO algorithm can avoid falling into local best value. Results of previous research reported in literature (Xie and Zeng 2009) to (Xie et al. 2010) have showed that compared with PSO algorithm and other traditional evolutionary algorithms, APO algorithm has better stability, diversity and robustness.

APO algorithm has been applied in single-objective optimization problems. Because of its good performance, we had originally introduced it in multi-objective optimization problems in our previous research works (Wang and Zeng 2010; Wang et al. 2010, 2011). Adopting aggregate functions, we transform multiple objectives into a single objective so that we can calculate the mass of each individual easily. The above-cited studies have shown that SMOAPO has an effective performance. Especially, it has better diversity compared with PSO algorithm and traditional evolutionary algorithms. However, mass function in those algorithms show the performance of each individual in the population to some extent, i.e., in other words, the methods of mass calculation adopted in those algorithms cannot embody the characteristics of MOPs sufficiently. Therefore, we present another APO algorithm for MOPs called RMOAPO, in which individuals' ranks based on Pareto-optimal concept are used to sort all individuals in the population. In addition, crowding degree is adopted to differentiate individuals with the same rank. As a result, each individual in population has a different rank according to the Pareto dominance relationship and information of crowding degree. Thus mass function value of each individual can be calculated in terms of its rank in population. This strategy incorporates characteristics of MOPs sufficiently and also keeps the diversity of population.

4 RMOAPO algorithm

Assuming population $P(t)$ generated in tth generation comprises N individuals, $n_i(t)$ denotes the number of individuals in $P(t)$ dominating individual i. Then the rank of individual i during the tth generation is denoted as $r_i(t) = 1 + n_i(t)$. Obviously, there will exist many individuals with the same rank during the tth generation. Then we adopt the crowding degree in a neighborhood to solve this problem. Here the neighborhood is denoted as a region within a given radius ε of the corresponding individual. According to the concept of individuals' ranks and the crowding degree of the individual within its neighborhood, we assign ranks to the individuals from 1 to N (here N is population size).

The idea of RMOAPO algorithm is summarized as follows:

Individuals in the population are sorted in ascending terms of their respective ranks based on Pareto-optimal concept mentioned above. When there are several individuals with the same rank, we rank these individuals according to their crowding degree within their respective neighborhoods. Thus, a relationship between crowding degree and individual's rank is required. To meet this, we construct the rule: the lower is the crowding degree, the smaller is the rank, while the higher the crowding degree, the greater the rank. Sometimes, there exist several individuals with the same rank and the same crowding degree. To solve this problem, we can adjust the radius and assign ranks to these individuals according to the crowding degree within the adjusted radius. It is seldom the case but sometimes there may be individuals with the same rank in terms of Pareto-optimal concept, and the same crowding degree even after the radius has been adjusted. Then random selection method is used to solve this problem. For example, if there are three non-dominated solutions A, B and C generated in the tth generation, according to the concept of ranking based on Pareto optimality, we know they are with the same rank. Now we want to assign natural numbers 1, 2 and 3 to them. So we have to know how many individuals are included in each individual's neighborhood. If the numbers of individuals within these neighborhoods are unequal, we can assign the three natural numbers to them in terms of the rule mentioned above. If there are 5 individuals in the neighborhood of individual A, 7 individuals in the neighborhood of individual B and 3 individuals in the neighborhood of individual C, then we can rank these three individuals as follows: $rank_A(t) = 2$, $rank_B(t) = 3$ and $rank_C(t) = 1$. However, if the number of individuals in two neighborhoods, say A and B, is the same, and it is 3 in C, then we have to adjust the radius of the neighborhood. Now we reduce the radius of the neighborhood to half at first and then check the number of individuals in A and B. Now we assume that this results in A having 4 and B having 3 individuals after the adjustment of radius. After this there exist $rank_A(t) = 3$, $rank_B(t) = 2$ and $rank_C(t) = 1$. Undoubtedly, sometimes the number of individuals in neighborhoods of A and B may be equal even after adjustment of radius. In this case, natural numbers 2 and 3 will be assigned to $rank_A(t)$ and $rank_B(t)$ randomly. Ranks of dominated individuals are assigned according to the same rule. If there are n_1 non-dominated solutions generated in the tth generation, the rest will be ranked from $n_1 + 1$. We move non-dominated individuals away from the population provisionally and deal with non-dominated individuals of the remainder of the population with the same method. The procedure is iterated until each individual in the original population has a different rank. The mass function in RMOAPO algorithm is modified as follows:

$$m_i(t) = \mathrm{e}^{\frac{-rank_i(t)}{N}} \tag{6}$$

Compared with SMOAPO, the mass function in RMOAPO algorithm can better embody the characteristics of MOPs.

In RMOAPO algorithm, radius r of the neighborhood is an important parameter which affects the algorithm greatly. Usually, it is decided by the decision maker according to the minimum expected distance between individuals in the Pareto-optimal set. However, in the real application, we cannot attach much importance to this information because we are not able to predict the Pareto-optimal set. On the other hand, if we want to decide the rank of an individual according to the number of individuals in its neighborhood, it will be difficult to evaluate the performance of the individuals, because there are too many individuals with no individuals in their neighborhoods when the radius of the neighborhood is set to be too small. Taking all the above into account, we set radius r of the neighborhood as in Eq. (7).

$$r = \frac{\left(\sum_{i=1}^{n}(\mathrm{Max}|\mathrm{E_i}|)^2\right)^{\frac{1}{2}}}{archivesize - 1} \tag{7}$$

where max $|E_i|$ is the maximum of Euclidean distance between individuals in the ith dimension of initial population and *archive size* is the size of the non-dominated solution set (which is the so-called archive).

When individuals with the same rank also have the same crowding degree, we have to adjust the size of radius of neighborhood. In other words, in order to evaluate these individuals' performance we must enlarge or reduce the radius of neighborhood. If the number of individuals in the neighborhood with the same rank is not more than 2, we change the radius to one-and-a-half times the original because it is meaningless to reduce the radius in this case; when there are more than 2 individuals in the neighborhood having the same rank, we reduce the radius to half. Then the crowding degree will be checked again after adjustment of radius.

The procedure of RMOAPO algorithm is summarized as follows:

Step 1: Initialize coordinates $x_{i,k}$ and $\mathrm{v}_{i,k}$ by random sampling within $[x_k^{\min}, x_k^{\max}]$ and $[\mathrm{v}_k^{\min}, \mathrm{v}_k^{\max}]$, respectively. In addition, radius r of neighborhood is calculated according to Eq. (7).

Step 2: The function value of each individual $i(i = 1, \ldots, N)$ with each objective is calculated such that the number of individuals dominating individual i, n_i, can be stored. Synchronously, non-dominated individuals are selected and stored in *archive* set. Rank assigned to non-dominated individuals is 1 while ranks assigned to the rest of individuals are $n_i + 1$.

Step 3: Sort all individuals in population in ascending order, according to ranks based on Pareto optimal mentioned above.

Step 4: For individuals with the same rank, the number of individuals in their neighborhoods r is checked and then they are sorted in ascending order according to the number of individuals in their respective neighborhoods. Individuals with the same number of individuals in their neighborhoods are marked as $flag(i) = 1$.

Step 5: If there exist individuals marked $flag(i) = 1$, radius r of neighborhood i is adjusted to r' according to the method mentioned above. Moreover, if the number of individuals in neighborhoods r' of these individuals is the same, their ranks are assigned randomly.

Step 6: Assign a serial number to each individual in the population in terms of the sorting result mentioned above. In other words, natural numbers 1 to N are assigned to each individual in the population as rank $rank_i(t)$ of the tth generation.

Step 7: Calculate each individual's mass with Eq. (6), as well as the total force exerted on each individual with Eqs. (2) and (3). Besides, velocity and position of each individual are updated with Eqs. (4) and (5).

Step 8: The function value of each individual $i (i = 1, \ldots, N)$ with each objective is calculated such that the number of individuals dominating individual i, $n_i^{\prime} (i = 1, \ldots, N)$, can be stored. Synchronously, non-dominated individuals are selected and non-dominated individuals are assigned rank 1 while ranks assigned to the rest of individuals are $n_i^{\prime} + 1$. Each non-dominated solution is compared with solutions stored in *archive* set and it is stored in *archive* set if the solution is equally good. While this non-dominated solution dominates some individuals in *archive* set, dominated individuals are deleted from *archive* set. Meanwhile, the non-dominated solution is put into *archive* set. When the number of individuals to be stored in *archive* set is more than the size of *archive*, the individual with the least number of individuals dominated by it is moved from *archive* set.

Step 9: Exit when the maximum number of iterations is achieved, otherwise return to Step 3.

5 Analysis to simulation test

For quantitative assessment of the performance of a multi-objective optimization algorithm, two issues are normally taken into consideration. One is minimization of the distance of the Pareto front produced by our algorithm from the true Pareto front, assuming we know its location. The other is maximization of the spread of solutions found, so that we can have a distribution of vectors as smooth and uniform as possible. Based on this notion, we adopted one metric to evaluate each aspect. Generational Distance (GD) addresses the first issue while Spacing (SP) (Carlos 2004) addresses the second issue. In addition, for compare the quality of the solution set of RMOAPO algorithm with those of other well-known algorithms, we use the $C\psi$ metric (Zitzler and Thiele 1999), which compares the convergence rate of two non-dominated sets.

The metric GD returns a value representing the average distance of solutions in the Pareto front obtained by a multi-objective optimization algorithm (PF_{known}) from the true Pareto front (PF_{true}). It is defined as follows:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{8}$$

where n is the number of non-dominated solutions obtained by the proposed algorithm and d_i is the Euclidean distance (in objective space) between each vector in PF_{known} and the nearest member of PF_{true}. It should be clear that a zero result indicates that all elements generated are in the Pareto optimal set, while any other value indicates how far PF_{known} deviates from PF_{true}. As this metric denotes the average distance from PF_{true}, a smaller value means greater proximity.

Another metric SP judges how well solutions in PF_{known} are distributed. It is defined as:

$$\begin{aligned} SP &= \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\bar{d} - d_i\right)^2} \\ d_i &= \min_j \left(\sum_{k=1}^{m} \left| f_k^i(x) - f_k^j(x) \right| \right) \\ & \quad i, j = 1, \ldots, n \text{ and } j \neq i \end{aligned} \tag{9}$$

where n is the number of solutions on PF_{known} and $\bar{d}$ is the mean of all d_i. A zero value for this metric indicates that all members of PF_{known} are equidistant. A smaller value represents a better diversity of PF_{known}.

In order to explain which algorithm has a better solution set, we use the third metric C to compare the convergence rate of two non-dominated sets A and B:

$$C(A,B) = \frac{\left| \left\{ \vec{b} \in B \mid \exists \vec{a} \in A : \vec{a} \succ \vec{b} \text{ or } \vec{a} = \vec{b} \right\} \right|}{|B|} \tag{10}$$

where $\vec{a} \succ \vec{b}$ denotes $\vec{a}$ dominates $\vec{b}$. The value of $C(A, B) = 1$ means that all the members of B are weakly dominated by the members of A. One can also conclude that $C(A, B) = 0$ means that none of the members of B is weakly dominated by the members of A. Usually $C(A, B)$ is not equal to $1 - C(A, B)$, and both $C(A, B)$ an $C(B, A)$ must be considered for comparisons.

We show the performance of RMOAPO algorithm with ZDT and DTLZ test suites and compare them against some well known techniques in multi-objective problems literature. We choose five popular benchmark functions of ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 (Table 1) and three complex benchmark functions of DTLZ1, DTLZ2 and DTLZ4 (Table 5).

6 Tests with ZDT suite

Firstly, we show the performance of RMOAPO algorithm with ZDT test suite. The variable dimensions in decision space of each benchmark functions are 30. Normally these five benchmark functions are used to test MOPs with different characteristics. The first test problem ZDT1 has a convex Pareto front. In the second problem ZDT2, the Pareto front is concave. The third problem ZDT3 is usually used to test the capability of dealing with discontinuous Pareto front. The fourth benchmark function ZDT4 is the most difficult in the ZDT series functions. It has 21^9

Table 1 Test problems of ZDT suite

Problem	Variable bounds	Objective functions
ZDT1	$x = (x_1, x_2, \ldots, x_m)$ $x_i \in [0, 1]$ $i = 1, 2, \ldots, m$ $m = 30$	$f_1(x_1) = x_1$ $f_2(x) = g\left(1 - \sqrt{f_1/g}\right)$ $g(x_2, \ldots, x_m) = 1 + 9\sum_{i=2}^{m} x_i/(m-1)$
ZDT2	$x = (x_1, x_2, \ldots, x_m)$ $x_i \in [0, 1]$ $i = 1, 2, \ldots, m$ $m = 30$	$f_1(x_1) = x_1$ $f_2(x) = g\left(1 - \left(f_1/g\right)^2\right)$ $g(x_2, \ldots, x_m) = 1 + 9\sum_{i=2}^{m} x_i/(m-1)$
ZDT3	$x = (x_1, x_2, \ldots, x_m)$ $x_i \in [0, 1]$ $i = 1, 2, \ldots, m$ $m = 30$	$f_1(x_1) = x_1$ $f_2(x) = g(1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)$ $g(x_2, \ldots, x_m) = 1 + 9\sum_{i=2}^{m} x_i/(m-1)$
ZDT4	$x = (x_1, x_2, \ldots, x_m)$ $x_1 \in [0, 1]$ $x_i \in [-5, 5]$ $i = 2, \ldots, m$ $m = 30$	$f_1(x_1) = x_1$ $f_2(x) = g(1 - \sqrt{f_1/g})$ $g(x) = 1 + 10(m-1) + \sum_{i=2}^{m}(x_i^2 - 10\cos(4\pi x_i))$
ZDT6	$x = (x_1, x_2, \ldots, x_m)$ $x_i \in [0, 1]$ $i = 1, 2, \ldots, m$ $m = 30$	$f_1(x_1) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(x) = g\left(1 - \left(f_1/g\right)^2\right)$ $g(x_2, \ldots, x_m) = 1 + 9\left(\sum_{i=2}^{m} x_i/(m-1)\right)^{0.25}$

different local Pareto fronts. It is usually adopted to test the capability of solving multi-modal MOPs with abundant local Pareto fronts. The last benchmark function ZDT6 is used to test a multi-objective optimization algorithm's capability of handling asymmetric Pareto front. The experimental environment is explained as follows: The size of population is set to 100, which is equal to the archive's size. The maximum generation is set to 100. Gravitational constant G is set to 10. The inertia weight w is decreased from 0.9 to 0.4 with linearity. We run each function 30 times to get the statistic values and show the 15th graphical result for comparing.

Besides designing the experiments on RMOAPO algorithm, we compare it with three famous MOEAs (i.e., NSGA, SPEA, MOPSO) and SMOAPO algorithm. We choose NSGA and SPEA because they are not only the classic MOEAs but also their results can be obtained from an open resource (Zilter and Laumanns 2008). It is a pity that we have not been able to obtain the result of NSGA-II, which is an excellent MOEA. Another compared approach, MOPSO, has been proposed in extant literature (Yang et al. 2008). Fortunately the author provided the results to us. The experiment environment in existing literature (Yang et al. 2008) is set as follows: The size of population is set to 100 and it is the same as the size of archive set. The maximum generation is set to 5,000. The cognitive and social parameters are fixed as $c_1 = c_2 = 2.05$ and inertia weight w is decreased from 0.9 to 0.4 with linearity. The experiment environment of SMOAPO algorithm is the same as that of RMOAPO algorithm.

Figure 1 shows graphical result produced by our RMOAPO in the first test function chosen. It is easy to notice that $\mathrm{PF}_{\mathrm{known}}$ of ZDT1 obtained by RMOAPO algorithm is almost on $\mathrm{PF}_{\mathrm{true}}$. From comparison of results of the five algorithms in Tables 2 and 3, it can be seen that performance of RMOAPO is the best with respect to SP. With respect to GD, it is slightly below the MOPSO algorithm. This is mainly because MOPSO algorithm has a much greater generation than RMOAPO algorithm. We can see from Table 4 that the non-dominated set produced by RMOAPO algorithm has a huge value of metric $C(A, B)$ and a little value of metric $C(B, A)$. So the non-dominated solutions obtained by RMOAPO algorithm dominate most solutions obtained by the other four algorithms.

Figure 2 shows graphical result produced by our approach in the second test function. It is obvious that RMOAPO algorithm has a good performance. From statistical values of function ZDT2, we can draw the conclusion that on this problem RMOAPO algorithm outperforms the other four algorithms not only for the metric of GD but also for SP. With the respect to the metric C, RMOAPO algorithm also has a good performance.

Figure 3 shows graphical result produced by RMOAPO algorithm in the third test function. It can be seen that PF_{known} by RMOAPO algorithm is almost on PF_{true}. The metric values showed in Tables 2 and 3 indicate that our approach has a better GD compared with NSGA, SPEA and SMOAPO algorithms, but it is worse than MOPSO algorithm. With respect to SP, RMOAPO algorithm is almost equivalent to those of SPEA, MOPSO and SMOAPO algorithms, which are much better than NSGA. With the respect to the metric C, RMOAPO algorithm is very close to those of SMOAPO algorithm and MOPSO, which are much better than NSGA.

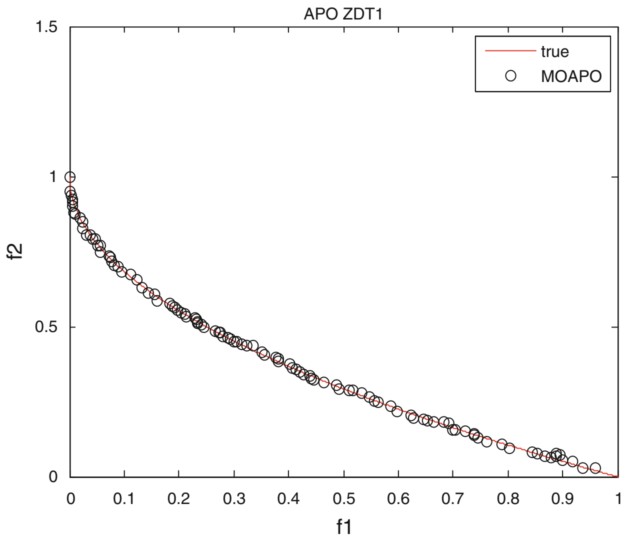

Fig. 1 Non-dominated solutions with RMOAPO on ZDT1

From Fig. 4 we can see that PF_{known} of ZDT4 obtained by our approach has an excellent proximity to PF_{true}. However, it is outperformed by MOPSO algorithm with respect of diversity. Although PF_{known} by SMOAPO algorithm is better than SPEA and NSGA, it is worse than RMOAPO algorithm. With the respect to the metric C, RMOAPO algorithm is much better than NSGA and it is also better than SMOAPO algorithm and SPEA. However, it is worse than MOPSO.

From Fig. 5 we can observe that PF_{known} of ZDT6 obtained is far from PF_{true} because PF_{true} of ZDT6 is uneven while all algorithms in this paper adopt strategies to keep PF_{known} spreading evenly. As the metric of C here has little meaning, we don't list it in Table 4.

Table 2 Statistics values for the metric of GD

Functions	GD	RMOAPO	SMOAPO	NSGA	SPEA	MOPSO
ZDT1	Mean	0.0360	0.0524	0.1715	0.1034	0.0022
	Best	0.0270	0.0365	0.1453	0.0893	0.0016
	Worst	0.0463	0.0628	0.2140	0.1181	0.0048
	Median	0.0433	0.0487	0.2140	0.1064	–
	Std	0.0051	0.0081	0.0155	0.0066	–
ZDT2	Mean	0.0176	0.0459	0.2668	0.1088	0.0509
	Best	0.0083	0.0105	0.2159	0.0889	0.0276
	Worst	0.0353	0.0562	0.3100	0.1340	0.0667
	Median	0.0353	0.0105	0.3019	0.0961	–
	Std	0.0058	0.0104	0.0232	0.0124	–
ZDT3	Mean	0.0429	0.0833	0.1417	0.1537	0.0052
	Best	0.0324	0.0451	0.1044	0.1294	0.0047
	Worst	0.0815	0.1008	0.1620	0.2001	0.0060
	Median	0.0348	0.0451	0.1263	0.1546	–
	Std	0.0094	0.0151	0.0137	0.0163	–
ZDT4	Mean	0.0730	0.0955	2.3950	0.7068	0.014
	Best	0.0180	0.0667	0.8817	0.1856	0
	Worst	0.1655	0.1305	5.1436	1.4667	0.0408
	Median	0.0629	0.1305	2.4916	0.8156	–
	Std	0.0241	0.0350	0.8727	0.3744	–
ZDT6	Mean	0.2131	0.4256	1.0846	0.3939	0.5886
	Best	0.1322	0.2311	0.7630	0.2323	0.2222
	Worst	0.3033	0.6546	1.9086	0.6023	0.8558
	Median	0.2059	0.6546	1.0412	0.5040	–
	Std	0.0341	0.2107	0.2437	0.1028	–

Table 3 Statistics values for the metric of SP

Functions	SP	RMOAPO	SMOAPO	NSGA	SPEA	MOPSO
ZDT1	Mean	0.0060	0.0102	0.0409	0.0218	0.0120
	Best	0.0041	0.0063	0.0300	0.0155	0.0084
	Worst	0.0118	0.0239	0.0603	0.0295	0.0203
	Median	0.0047	0.0069	0.0512	0.0227	–
	Std	0.0016	0.0051	0.0071	0.0031	–
ZDT2	Mean	0.0049	0.0069	0.0725	0.0374	0.0151
	Best	0.0035	0.0038	0.0337	0.0217	0.011
	Worst	0.0111	0.0113	0.1525	0.0573	0.020
	Median	0.0053	0.0083	0.0830	0.0349	–
	Std	0.0015	0.0024	0.0269	0.0085	–
ZDT3	Mean	0.0235	0.0233	0.0425	0.0222	0.0305
	Best	1.005e−4	0.0057	0.0276	0.0166	0.0276
	Worst	0.1611	0.0971	0.0770	0.0295	0.0341
	Median	5.4351e−4	0.0285	0.0379	0.0211	–
	Std	0.0448	0.0266	0.0112	0.0033	–
ZDT4	Mean	0.1817	0.2472	0.2820	0.7068	0.0114
	Best	0.0173	0.0029	9.5013−e9	0.1856	0.0055
	Worst	1.0517	1.3377	0.5634	1.4667	0.0410
	Median	0.2103	0.0239	0.1651	0.8156	–
	Std	0.2949	0.3857	0.1176	0.3744	–
ZDT6	Mean	0.4823	0.7106	0.4156	0.3066	0.4571
	Best	0.3632	0.3013	0.3001	0.0239	0.2848
	Worst	0.8637	0.8426	0.7441	0.6417	0.5994
	Median	0.8637	0.3924	0.4041	0.4898	–
	Std	0.0701	0.0754	0.0935	0.0712	–

Table 4 Numerical results for the metric of *C*, here we assume *A* as the non-dominated set produced by RMOAPO algorithm and *B* as that produced by the other algorithm

Functions	C	SMOAPO	NSGA	SPEA	MOPSO
ZDT1	$C(A, B)$	0.89	0.96	0.93	0.56
	$C(B, A)$	0.17	0.10	0.11	0.59
ZDT2	$C(A, B)$	0.86	0.98	0.93	0.79
	$C(B, A)$	0.21	0.09	0.11	0.29
ZDT3	$C(A, B)$	0.67	0.98	0.76	0.53
	$C(B, A)$	0.59	0.10	0.31	0.64
ZDT4	$C(A, B)$	0.74	0.99	0.71	0.47
	$C(B, A)$	0.31	0.09	0.36	0.68

As we have seen from these experiments, we can conclude that we have built a competitive algorithm using ranks of individuals based on Pareto-dominated concept and the crowding degree between individuals. Moreover, evolutionary generations in our approach are far less than the compared algorithms, except for SMOAPO, which reduces computing and time costs. Hence, our approach is specifically superior for dealing with MOPs whose Pareto fronts have the characteristics of convex, concave, disconnect, multi-modal and numerous local optimal.

7 Experiments with DTLZ test suite

Next we show the performance of RMOAPO algorithm with three complex problems (Table 5) of DTLZ test suite. Normally these three benchmark functions are complex and used to test MOPs with different characteristics. The first test problem DTLZ1 is an M-objective problem with a linear Pareto-optimal front. Its Pareto-optimal solution corresponds to $x_M = 0$ and the objective function values lie on the linear hyper-plane: $\sum_{m=1}^{M} f_m = 0.5$. A value of $k = 5$ is suggested here. In the above problem, the total number of variables is $n = M + k - 1$. The difficulty in this problem is to converge to the hyper-plane. The search space contains $(11^k - 1)$ local Pareto-optimal fronts, each of which can attract the algorithm. In the second problem DTLZ2 the Pareto-optimal solutions corresponds to $x_M = 0.5$ and all objective function values must satisfy the equation of $\sum_{i=1}^{M} (f_i(x))^2 = 1$. Here, a value of $k = |x_M| = 10$ is suggested. The total number of variables

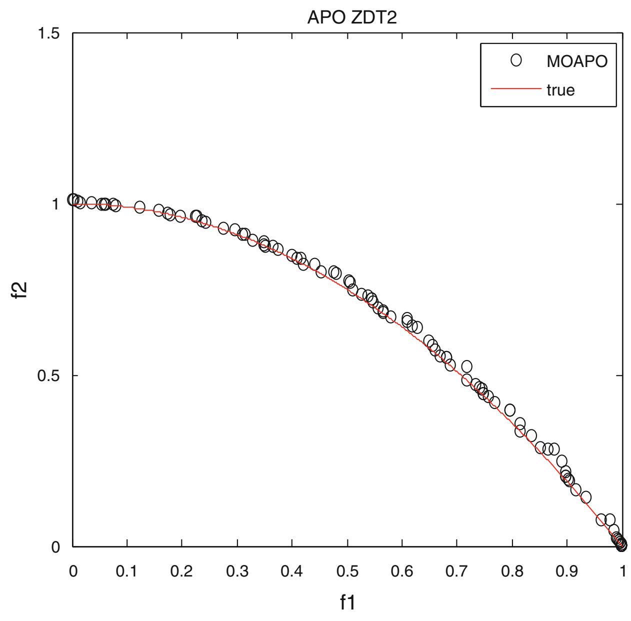

Fig. 2 Non-dominated solutions with RMOAPO on ZDT2

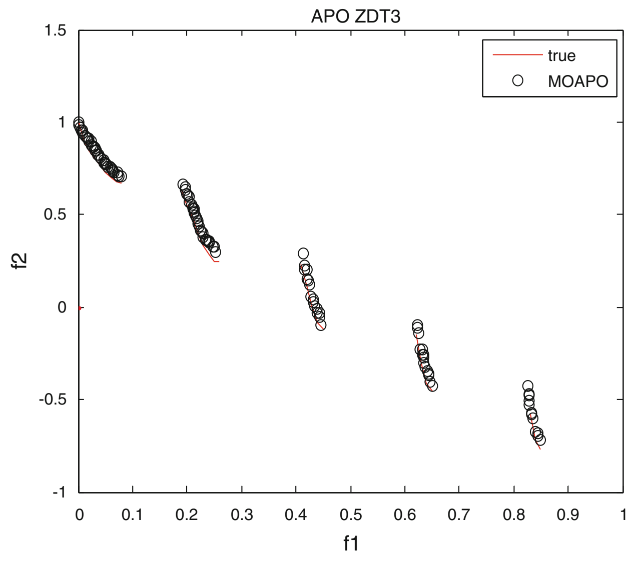

Fig. 3 Non-dominated solutions with RMOAPO on ZDT3

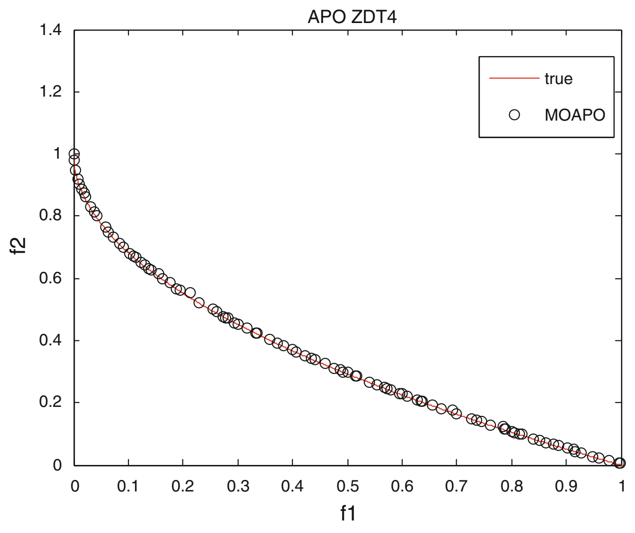

Fig. 4 Non-dominated solutions with RMOAPO on ZDT4

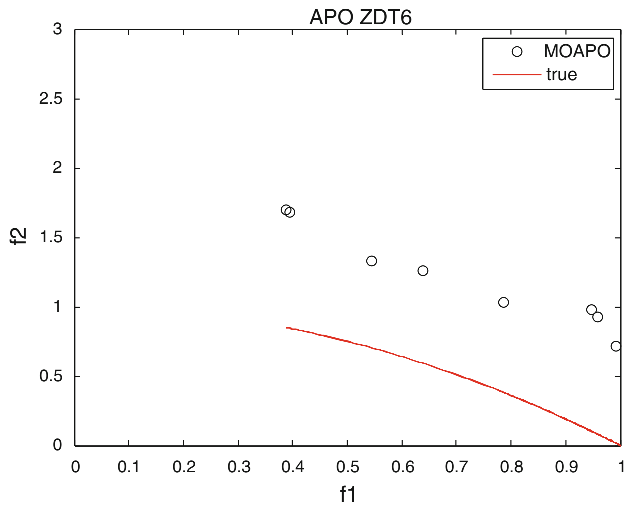

Fig. 5 Non-dominated solutions with RMOAPO on ZDT6

is $n = M + k - 1$. The third problem DTLZ4 is usually used to investigate a multi-objective optimization algorithm's ability to maintain a good distribution of solutions. The parameter $\alpha = 100$ is suggested here. And here $k = |x_M| = 10$ is suggested. There are $n = M + k - 1$ decision variables in the problem.

The experimental environment of RMOAPO algorithm is explained as follows: The size of population is set to 100, which is equal to the archive's size. The maximum generation is set to 100 so that the algorithm has a low computing cost. Gravitational constant G is set to 10. The inertia weight w is decreased from 0.9 to 0.4 with linearity. Each function chosen is run 30 times to get the statistics values and the 15th graphical result of each function is shown.

We compare RMOAPO algorithm with three other multi-objective optimization algorithms (i.e., NSGA-II, MOPSO and SMOAPO). In the experiment of NSGA-II, the environment is set as follows: We use a population of size 100, a crossover probability of 0.8, a mutation probability of $^1/_n$ (where n is the number of variables). We run NSGA-II for 200 generations. As we have not been able to obtain DTLZ's results of MOPSO, we have to code the algorithms according to its pseudocode in Yang et al. (2008) by ourselves. The size of population is set to 100 and it is the same as the size of archive set. The maximum generation is set to 500 so that its solutions are comparable. The cognitive and social parameters are fixed as $c_1 = c_2 = 2.05$ and inertia weight w is decreased from 0.9 to 0.4 with

Table 5 Three test problems of DTLZ test suite

Problem	Variable bounds	Objective functions
DTLZ1	$x = (x_1, x_2, \ldots, x_n)$ $x_i \in [0, 1]$ $i = 1, 2, \ldots, n$ $n = 6$	$f_1(\vec{x}) = \frac{1}{2}x_1(1 + g(\vec{x}))$ $f_2(\vec{x}) = \frac{1}{2}(1 - x_1)(1 + g(\vec{x}))$ $g(\vec{x}) = 100[\lvert\vec{x}\rvert + \sum_{x_i \in \vec{x}} (x_1 - 0.5)^2 - \cos(20\pi(x_i - 0.5))]$
DTLZ2	$x = (x_1, x_2, \ldots, x_n)$ $x_i \in [0, 1]$ $i = 1, 2, \ldots, n$ $n = 11$	$f_1(\vec{x}) = (1 + g(\vec{x}))\cos(x_1\frac{\pi}{2})$ $f_2(\vec{x}) = (1 + g(\vec{x}))\sin(x_1\frac{\pi}{2})$ $g(\vec{x}) = \sum_{x_i \in \vec{x}} (x_1 - 0.5)^2$
DTLZ4	$x = (x_1, x_2, \ldots, x_n)$ $x_i \in [0, 1]$ $i = 1, 2, \ldots, n$ $n = 11$	$f_1(\vec{x}) = (1 + g(\vec{x}))\cos(x_1^{\alpha}\frac{\pi}{2})$ $f_2(\vec{x}) = (1 + g(\vec{x}))\sin(x_1^{\alpha}\frac{\pi}{2})$ $g(\vec{x}) = \sum_{x_i \in \vec{x}} (x_1 - 0.5)^2$ $\alpha = 100$

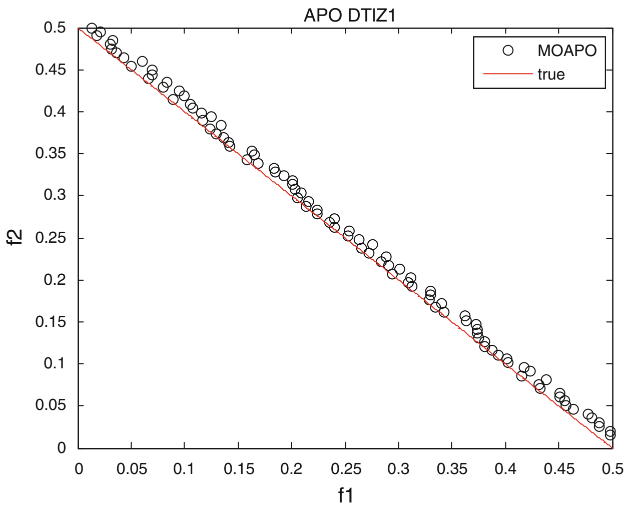

Fig. 6 Non-dominated solutions with RMOAPO on DTLZ1

Table 6 Statistics values for the metric of GD

Functions	GD	RMOAPO	SMOAPO	NSGA-II	MOPSO
DTLZ1	Mean	0.1391	0.1838	0.2489	0.3553
	Best	0.1203	0.1376	0.2097	0.2952
	Worst	0.2447	0.2619	0.3144	0.3649
	Median	0.1412	0.1513	0.3034	0.3573
	Std	0.0065	0.0087	0.0046	0.0055
DTLZ2	Mean	0.0212	0.0433	0.0248	0.0469
	Best	0.0097	0.0112	0.0102	0.0179
	Worst	0.0336	0.0537	0.0348	0.0632
	Median	0.0311	0.0363	0.0295	0.0487
	Std	0.0043	0.0053	0.0037	0.0051
DTLZ4	Mean	0.0194	0.0231	0.0207	0.0386
	Best	0.0083	0.0092	0.0089	0.0139
	Worst	0.0276	0.0357	0.0251	0.0349
	Median	0.0191	0.0304	0.0173	0.0283
	Std	0.0036	0.0039	0.0034	0.0047

linearity. The experimental environment of SMOAPO algorithm is the same as that of RMOAPO algorithm. It is also run 30 times to obtain the statistics values.

Figure 6 depicts graphically the result produced by our RMOAPO in the DTLZ1 problem. It is easy to notice that PF_{known} of DTLZ1 obtained by RMOAPO algorithm is almost on PF_{true}. From comparison of results of the above three algorithms in Tables 6, 7, it can be seen that the performance of RMOAPO is very close to NSGA-II with respect to the metric of SP. However, NSGA-II algorithm has a much greater generation than RMOAPO algorithm. With respect to GD, it is better than the other three algorithms. We can see from Table 8 that the non-dominated set produced by RMOAPO algorithm has a great value of metric $C(A, B)$ and a little value of metric $C(B, A)$. So the non-dominated solutions obtained by RMOAPO algorithm dominate most solutions obtained by the other three algorithms.

Figure 7 depicts graphically the result produced by our approach in the DTLZ2 problem. It is obvious that RMOAPO algorithm has a good performance. From the GD and SP metrics of the function DTLZ2, we can draw the conclusion that on this problem RMOAPO algorithm outperforms the other three algorithms for both the metrics. With the respect to the metric C, RMOAPO algorithm also has a good performance on this problem.

Figure 8 depicts graphically the result produced by RMOAPO algorithm in the DTLZ4 function. It can be seen that PF_{known} by RMOAPO algorithm is almost on PF_{true}. The metric values showed in Tables 6, 7 indicate that our approach has a better *SP* compared with the other three

Table 7 Statistics values for the metric of SP

Functions	SP	RMOAPO	SMOAPO	NSGA-II	MOPSO
DTLZ1	Mean	0.0067	0.0093	0.0064	0.0107
	Best	0.0036	0.0071	0.0031	0.0079
	Worst	0.0094	0.0115	0.0083	0.0171
	Median	0.0061	0.0089	0.0058	0.0095
	Std	0.0023	0.0047	0.0019	0.0034
DTLZ2	Mean	0.0046	0.0073	0.0068	0.0087
	Best	0.0032	0.0046	0.0041	0.0053
	Worst	0.0097	0.0125	0.0113	0.0176
	Median	0.0058	0.0096	0.0095	0.0104
	Std	0.0021	0.0031	0.0026	0.0037
DTLZ4	Mean	0.0029	0.0038	0.0035	0.0053
	Best	0.0017	0.0015	0.0021	0.0032
	Worst	0.0092	0.0091	0.0073	0.0103
	Median	0.0053	0.0085	0.0037	0.0068
	Std	0.0036	0.0039	0.0023	0.0042

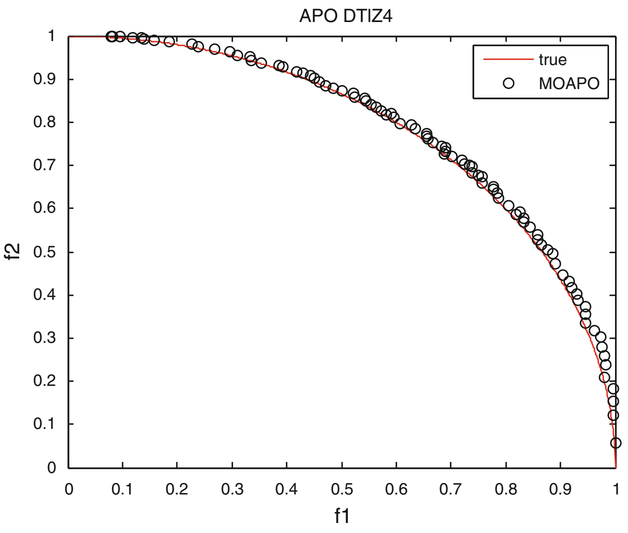

Fig. 8 Non-dominated solutions with RMOAPO on DTLZ4

Table 8 Numerical results for the metric of *C*, here we assume *A* as the non-dominated set produced by RMOAPO algorithm and *B* as that produced by the other algorithm

Functions	C	SMOAPO	NSGA-II	MOPSO
DTLZ1	*C*(*A*, *B*)	0.98	1	1
	C(*B*, *A*)	0.17	0.11	0.09
DTLZ2	*C*(*A*, *B*)	0.78	0.81	0.87
	C(*B*, *A*)	0.31	0.26	0.24
DTLZ4	*C*(*A*, *B*)	0.75	0.79	0.82
	C(*B*, *A*)	0.43	0.37	0.23

Table 9 Comparison of CPU time consuming (s)

Functions	RMOAPO	SMOAPO	NSGA-II	MOPSO
DTLZ1	9.7518	9.4246	17.2397.	18.6546
DTLZ2	8.9512	8.7183	19.8137	20.8769
DTLZ4	8.2471	7.9752	18.7792	20.8213

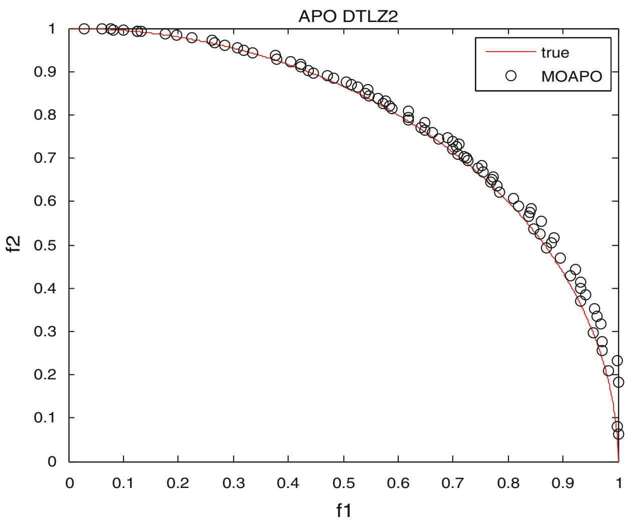

Fig. 7 Non-dominated solutions with RMOAPO on DTLZ2

algorithms. With respect to GD, RMOAPO algorithm is almost equivalent to those of NSGA-II and SMOAPO algorithms, which are better than MOPSO. We can see from Table 8 that RMOAPO algorithm has a better performance with the respect to the metric *C*.

Since the efficiency is an important matter in multi-objective optimization, we use the mean of CPU time consuming to evaluate the efficiency of each algorithm compared. It shows in Table 9.

From Table 9 we can see RMOAPO algorithm has a better efficiency compared with NSGA-II and MOPSO. However the CPU time consumed by RMOAPO algorithm is longer than that of SMOAPO. That is mainly because the masses of individuals in SMOAPO algorithm produced by the method of aggressive function, which is more simple than that in RMOAPO algorithm. However, this method of mass obtaining in SMOAPO algorithm can not embody the characteristics of multi-objective optimization problems adequately.

We can draw a conclusion from these two test suites that our approach has a comparable performance on dealing with many complex multi-objective optimization problems. It can converge to the true Pareto front with fewer generations than NSGA, SPEA, NSGA-II, MOPSO etc. so that it has a lower computing cost. Moreover, RMOAPO algorithm has a good diversity and a comparable efficiency.

8 Conclusions

We have presented a RMOAPO algorithm to deal with MOPs. In this approach individuals' ranks in terms of Pareto-dominated concept are used to evaluate the performance of each individual. In addition, crowding degree within the individual's neighborhood is checked as another index to evaluate the performance of individuals with the same Pareto-dominated rank. Moreover, the radius of the neighborhood is adjusted if there are several individuals with the same crowding degree. Besides, random selection method is used if individuals exist as "equal optimal" after being sorted by means of the method mentioned above. Thus each individual in the population has a unique rank, which indicates the performance of an individual. Then the mass of each individual is calculated using its rank. Afterwards combined virtual force exerted on each individual can be calculated with its mass. As a result, velocity and position of each individual are updated. Simulation results on ZDT and DTLZ series functions have showed RMOAPO algorithm has a competitive performance with less generations compared with some famous MOEAs and SMOAPO algorithm.

As the APO algorithm has strong global search ability, while strategies of ranking based on Pareto-dominated concept and crowding degree within neighborhood embody the characteristics of MOPs, we can obtain satisfactory results when handling the MOPs whose Pareto fronts have the characteristics of convex, concave, disconnect, multimodal and numerous local optimal. However, during the test procedure, we have also observed that our approach is not suitable for MOPs with an uneven Pareto front. In the future, more research is required in this area in order to extend the suitable regions of our approach. Moreover, we plan to test functions with high dimensions and large-scale MOPs. Synchronously, we will analyze the convergence of RMOAPO algorithm theoretically.

Acknowledgments The authors gratefully acknowledge the comments from the anonymous reviewers, which greatly helped them to improve the contents of this paper. The authors acknowledge support from the fund of the scholarship to pursue graduated studies in Taiyuan University of Science and Technology and the project number is 20122030.

References

Abido MA (2007) Two level of nondominated solutions approach to multi-objective particle swarm optimization. In: Thierens D, Beyer HG, Bongard J, Branke J, Clark JA, Cliff D, Congdon CB, Deb K (eds) Proceedings of the genetic and evolutionary computation conference, GECCO 2007. ACM Press, New York, pp 726–733

Coello CAC, Lechuga MS (2002) MOPSO: a proposal for multiple objective particle swarm optimization. Congress on Evolutionary Computation (CEC'2002), Vol 2, pp 1051–1056, IEEE Service Center, Piscataway

Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objective with particle swarm optimization. IEEE Trans Evol Comput 8(3):256–279

Corne DW, Knowles JD, Oates MJ (2000) The Pareto envelope-based selection algorithm for multi-objective optimization. Proceedings of the Parallel Problem Solving from Nature VI Conference, pp 839–848. Springer

Deb K, Pratap A, Agarwal S, Meyarivan T (2000) A fast and elitist multi-objective genetic algorithm: NSGA-II. Parallel problem solving from nature (PPSN VI). pp 849–858, Springer

Durillo JJ, García-Nieto J, Nebro AJ, Coello CAC, Luna F, Alba E (2009) Multi-objective particle swarm optimizers: an experimental comparison. Evolutionary multi-criterion optimization. 5th International Conference, EMO 2009, pp 495–509, Springer. Lecture Notes in Computer Science, vol 5467, Nantes, April 2009

Fernández E, López E, Bernal S, Coello CAC (2010) Evolutionary multiobjective optimization using an outranking-based dominance generalization. Comput Oper Res 37(2):390–395

Fieldsend JE, Singh S (2002) A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence. Proceedings of the UK workshop on computational intelligence. Birmingham, pp 37–44

Fonseca CM, Fleming PJ (1993) Genetic algorithms for multi-objective optimization: formulation, discussion and generalization. Presented at Proceedings of the 5th International Conference on Genetic Algorithms, San Mateo

Garza Fabre M, Toscano Pulido G, Coello CAC (2009) Ranking methods for many-objective problems. MICAI 2009: advances in artificial intelligence. 8th Mexican International Conference on Artificial Intelligence, pp 633–645, Springer, Lecture Notes in Artificial Intelligence, vol 5845, Guanajuato

Garza Fabre M, Toscano Pulido G, Coello CAC (2010) Two novel approaches for many-objective optimization. 2010 IEEE congress on evolutionary computation (CEC'2010), pp 4480–4487, IEEE Press, Barcelona

Horn J, Nafpliotis N, Goldberg DE (1994) A niched pareto genetic algorithm for multi-objective optimization. IEEE World Congress on Computational Computation, Piscataway

Hu X, Eberhart RC (2002) Multiobjective optimization using dynamic neighborhood particle swarm optimization. Proceedings of IEEE congress on evolutionary computation (CEC2002). Honolulu

Knowles JD, Corne DW (1999) The Pareto archived evolution strategy: a new baseline algorithm for pareto multi-objective optimization. Congress on Evolutionary Computation (CEC99) Piscataway

López A, Aguirre H, Tanaka K, Coello CAC (2010) Objective space partitioning using conflict information for many-objective optimization. Parallel problem solving from nature–PPSN XI, 11th International Conference, Part I, pp 657–666, Springer, Lecture Notes in Computer Science, vol 6238, Krakow

Martínez SZ, Coello CAC (2010) An archive strategy based on the convex hull of individual minima for MOEAs. 2010 IEEE Congress on Evolutionary Computation (CEC'2010), pp 912–919, IEEE Press, Barcelona

Molina J, Santana LV, Hernández-Díaz AG, Coello CAC, Caballero R (2009) g-dominance: reference point based dominance for multi-objective metaheuristics. Eur J Oper Res 197(2):685–692

Nebro AJ, Durillo JJ, Garcia-Nieto J, Coello CAC, Luna F, Alba E (2009) SMPSO: a new PSO-based metaheuristic for multi-objective optimization. IEEE symposium on computational intelligence in multicriteria decision-making, pp 66–73, IEEE Press, Nashville, March 30–April 2, 2009

Nedjah N, dos Santos Coelho L, de Macedo de Mourelle L (2010) Multi-objective swarm intelligent systems. Theory Exp, pp 83–104, Springer, Berlin

Santana-Quintero LV, Ramírez-Santiago N, Coello CAC, Molina Luque J, García Hernández-Díaz A (2006) A new proposal for multiobjective optimization using particle swarm optimization and rough sets theory. Parallel Problem Solving from Nature (PPSN IX). 9th International Conference, Springer, pp 483–492, Lecture Notes in Computer Science, vol 4193, Reykjavik, September 2006

Sierra MR, Coello CAC (2005) Improving PSO-based multi-objective optimization using crowding, mutation and ε-dominance. In: Coello CAC, Aguirre AH, Zitzler E (eds) Proceedings of the 3rd international conference on evolutionary multi-criterion optimization EMO 2005. Springer, Berlin, pp 505–519

Spears WM, Spears DF, Kerr W et al (2005) An overview of Physicomimetics'. Lecture Notes in Computer Science-State of the Art Series 3342:84–97

Srinivas N, Deb K (1994) Multi-objective optimization using nondominated sorting in genetic algorithms. Evol Comput 2:221–248

Toscano-Pulido G, Coello CAC, Santana-Quintero LV (2007) EMO-PSO: A multi-objective particle swarm optimizer with emphasis on efficiency. Evolutionary multi-criterion optimization, 4th International Conference, EMO 2007, pp 272–285, Springer. Lecture Notes in Computer Science, vol. 4403, Matshushima

Wang Y, Zeng JC (2010) A multi-objective optimization algorithm based on artificial physics optimization. Control Decis 25(7): 1040–1044

Wang Y, Zeng JC, Tan Y (2010) A multi-objective artificial physics optimization algorithm based on virtual force sorting. International Conference on Swarm, Evolutionary and Memetic Computing (SEMCCO 2010), pp. 615–622, Springer LNCS 6466, Chennai, Dec 16–18, 2010

Wang Y, Zeng JC, Cui ZH, He XJ (2011) A novel constraint multi-objective artificial physics optimization algorithm and its convergence. Int J Innov Comput Appl 3(2):61–70

Xie LP, Zeng JC (2009) A global optimization based on physicomimetics framework. The 2009 World Summit on Genetic and Evolutionary Computation (GEC'09), Shanghai

Xie LP, Zeng JC, Cui ZH (2009) Using artificial physics to solve global optimization problems. The 8th IEEE international conference on cognitive informatics (ICCI 2009), Hong Kong

Xie LP, Zeng JC (2010) The performance analysis of artificial physics optimization algorithm driven by different virtual force. ICIC Express Lett (ICIC-EL) 4(1):239-244

Xie LP, Zeng JC, Cui ZH (2010) On mass effects to artificial physics optimization algorithm for global optimization problems. Int J Innov Comput Appl 2(2):69–76

Yang JJ, Zhou JZ, Fang RC et al (2008) Multi-objective particle swarm optimization based on adaptive grid algorithms. J Syst Simul 20(21):5843–5847

Zeng JC, Jie J, Cui ZH (2004) Particle swarm optimization. Science Press, Beijing

Zilter E, Laumanns M (2008) Test problems and test data for multiobjective optimizers [EB/OL] (30 Jun 2008) http://www.tik.ee.ethz.ch/sop/download/supplementary/testProblemSuite/

Zitzler E, Thiele L (1999) Multi-objective evolutionary algorithms: a comparative case study at strength Pareto approach. IEEE Trans Evol Comput 3:257–271

Zitzler E, Thiele L (2002) SPEA2: Improving the strength Pareto evolutionary algorithm for multi-objective optimization. EUROGEN 2001-evolutionary methods for design, optimization and control with applications to industrial problems, pp 95–100, Athens