

# Duality evolution: an efficient approach to constraint handling in multi-objective particle swarm optimization

Amin Ebrahim Sorkhabi<sup>1</sup> · Mehran Deljavan Amiri<sup>2</sup> · Ali Reza Khanteymoori<sup>2</sup>

Published online: 14 November 2016  
© Springer-Verlag Berlin Heidelberg 2016

**Abstract** This paper proposes an efficient approach for constraint handling in multi-objective particle swarm optimization. The particles population is divided into two non-overlapping populations, named infeasible population and feasible population. The evolution process in each population is done independent of the other one. The infeasible particles are evolved in the constraint space toward feasibility. During evolution process, if an infeasible particle becomes a feasible one, it migrates to feasible population. In a parallel process, the particles in feasible population are evolved in the objective space toward Pareto optimality. At each generation of multi-objective particle swarm optimization, a leader should be assigned to each particle to move toward it. In the proposed method, a different leader selection algorithm is proposed for each population. For feasible population, the leader is selected using a priority-based method in three levels and for infeasible population, a leader replacement method integrated by an elitism-based method is proposed. The proposed approach is tested on several constrained multi-objective optimization benchmark problems, and its results are compared with two popular state-of-the-art constraint handling multi-objective algorithms. The experimental results indicate that the proposed algorithm is highly competitive in solving the benchmark problems.

**Keywords** Evolutionary algorithms · Multi-objective optimization · Particle swarm optimization · Constraint handling · MOPSO

## 1 Introduction

One of the most important applications of evolutionary algorithms (EAs) in engineering is solving optimization problems. In single objective optimization, the goal is to find optimal solution(s) for one objective function. In an advanced form, the optimizer should find optimal solutions for multiple objective functions simultaneously (Liu et al. 2014). The objectives can be conflicting together in some complicated cases. This type of optimization problems is called multi-objective optimization problems (MOOPs). The optimization problems could be difficult to solve if there exist some constraint functions and boundary limitations for variables. These types of optimization problems are called constrained multi-objective optimization problems (CMOPs) (Tessema and Yen 2006; Coello et al. 2010). A CMOP can be formulated as:

$$\begin{aligned} f_i(x) &= f_i(x_1, x_2, \dots, x_n) \quad i = 1, \dots, k \\ g_j(x) &= g_j(x_1, x_2, \dots, x_n) < 0 \quad j = 1, \dots, q \\ h_j(x) &= h_j(x_1, x_2, \dots, x_n) = 0 \quad j = q + 1, \dots, m \\ x_j^{\min} &\leq x_j \leq x_j^{\max} \quad j = 1, \dots, n \end{aligned} \quad (1)$$

where  $x$  is the decision vector of  $n$  variables.  $x_j^{\max}$  is its upper bound, and its lower bound is  $x_j^{\min}$ .

The search space is an  $n$ -dimensional hyperbox in  $\mathfrak{R}^n$ , and the optimizer's task is to find the optimal solutions for  $k$  objective functions  $f_i(x)$ , simultaneously that have been defined on the search space  $S \subseteq \mathfrak{R}^n$ . Generally, to solve the

Communicated by A. Di Nola.

✉ Mehran Deljavan Amiri  
mehran\_deljavan@znu.ac.ir

<sup>1</sup> Computer Engineering Department, Islamic Azad University – Qazvin Branch, Qazvin, Iran

<sup>2</sup> Computer Engineering Department, University of Zanjan, Zanjan, Iran

CMOPs, the optimizer faces with two types of constraints, called inequality and equality constraints. There are a total of  $m$  constraints,  $q$  inequality and  $m - q$  equality, which are required to be satisfied by the optimal solutions. In the above formulation,  $g_j(x)$  is the  $j$ th-inequality constraint, and  $h_j(x)$  is the  $j$ th-equality constraint. The search space is bounded in its each dimension with lower ( $x_j^{\min}$ ) and upper ( $x_j^{\max}$ ) limitations. Due to these constraints, the search space is divided into two regions, named feasible region  $F \subseteq S$  and infeasible region  $I \subseteq S (I \cup F = S)$ . The solutions located in feasible region satisfy all constraints, whereas the solutions in infeasible region violate some constraints. In the recent decades, evolutionary algorithms have been used to solve MOOPs (Lu and Yen 2003; Yen and Haiming 2003). A complete literature review of evolutionary MOO could be found in Coello (2006), Coello et al. (2010), Mukhopadhyay et al. (2014a), Mukhopadhyay et al. (2014b). The multi-objective evolutionary algorithms' (MOEAs) goal is to find optimal solutions for objective functions simultaneously. There are more researches in the field of constrained multi-objective optimization described in Deb (2001).

This paper extends the multi-objective particle swarm optimization (MOPSO) algorithm proposed by Coello et al. (2004) in order to focus on handling constraints. The proposed algorithm basically modifies and enhances the mentioned MOPSO algorithm with the purpose of handling constraints by proposing a novel method. This method follows the general approach of dividing the total search space into two non-overlapping, constraint and objective spaces, and avoids manipulating the objective vector of the infeasible solutions. The population is also divided into two non-overlapping sub populations, called infeasible population and feasible population. The infeasible particles are placed in the infeasible population, and the feasible particles are placed in the feasible population. The infeasible population is evolved in the constraint space, and the feasible population is evolved in the objective space. In the constraint space, the infeasible particles move toward feasibility, while in the objective space the feasible particles move toward the Pareto optimality. During evolution in the constraint space, each infeasible particle that becomes a feasible one migrates to the feasible population. Furthermore, two new approaches based on the combination of the gravity distance and the crowding distance are proposed to select the leaders for the feasible and infeasible particles.

The rest of this paper is organized as follows. Section 2 provides a brief overview of the various evolutionary approaches developed for CMOPs. Section 3 discusses about the main motivation of the proposed algorithm. In Sect. 4, PSO is briefly introduced. In Sect. 5, the general concepts of the proposed algorithm for constraint handling in MOPSO (CHMOPSO) is presented. Section 4 discusses some useful related preliminaries about PSO basics. Sections 6 and 7

describe evolution process in feasible population and infeasible population in details. In Sect. 8 the proposed method is applied to some CMOP benchmark problems, and the results and analysis are shown. Finally, some conclusions are drawn in Sect. 9.

## 2 Related works

In this section, a review of developed EAs to solve CMOPs is presented. In some approaches, individuals (or particles) that violate any constraints (infeasible members) are ignored (Back et al. 1991). Regardless the fact that this method is easy to implement, it might be difficult to find feasible individuals. Rejecting infeasible solutions in the vicinity of feasible individuals impairs the search capability which affects the algorithm's ability to find feasible individuals. In fact, the major drawback of these methods is that no information is extracted from the ignored members. The most popular way to handle constraints in evolutionary methods is using penalty functions. In the constraint handling literature, there are dozens of evolutionary methods that uses penalty functions to handle constraints. There are three types of penalty functions entitled as *static penalty functions* (Yeniay 2005), *dynamic penalty functions* (Joines and Houck 1994) and *adaptive penalty functions* (Bean and Hadj-Alouane 1992; Farmani and Wright 2003; Hu and Yen 2015). The static penalty functions are the weighted sum of the constraint violations. In dynamic penalty functions, the current generation number is considered in calculating the penalty value. Finally, the adaptive penalty functions gather information from search process to use in calculating the penalty value for infeasible solutions.

In another categorization, the proposed constraint handling methods in the literature may be categorized as: non-elitism-based methods and elitism-based methods. Fonseca and Fleming proposed a rank-based fitness assignment method for multi-objective genetic algorithms (Fonseca and Fleming 1993). A non-dominated Sorting Genetic Algorithm (NSGA) was proposed by Srinivas and Deb in Srinivas and Deb (1994) as another non-elitism-based approach. This method presents various techniques to distribute solutions in non-dominated fronts. Elitism-based algorithms are proposed to reach better convergence properties. An improved version of NSGA, called NSGA-II, was later proposed by Deb et al. (2000). The Pareto Archived Evolution Strategy (PAES) (Knowles and Corne 1999) by Knowles et al. archives a history of all non-dominated solutions found in all entire generation to elite solutions. It also uses a recursive algorithm to distribute Pareto front diversity. Zitzler et al. propose the Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler and Thiele 1999) that uses a repository set to archive non-dominated solutions. Also, they use a clustering

approach to maintain diversity. [Yen and He \(2014\)](#) propose an ensemble method to compare MOEAs by combining a number of performance metrics using double elimination tournament selection. The double elimination design allows characteristically poor performance of a quality algorithm to still be able to win it all. This method can provide a more comprehensive comparison among various MOEAs than what could be obtained from a single performance metric alone.

Some approaches such as [Joines and Houck \(1994\)](#), [Deb \(2000\)](#) prefer feasible solutions to infeasible solutions. In these methods, feasible individuals always receive more attention than infeasible individuals. So, in population fitness ranking, feasible individuals come first followed by infeasible individuals with lower constraint violation and individuals with higher constraint violation come at the end of the ranked list. [Takahama and Sakai \(2005\)](#), [Takahama and Sakai \(2006\)](#), proposed algorithms in which constraint violation and objective function are separately optimized. The proposed method in [Carreno Jara \(2014\)](#) allows to value the relative importance of those solutions with outstanding performance in very few objectives and poor performance in all others, regarding those solutions with an equilibrium (balance) among all the objectives. In this method, the optimality criteria avoid to interrelate the relative values of the different objectives, respecting the integrity of each one in a rational way.

Balancing constraints and objective function is a fundamental issue in constrained evolutionary optimization. But this issue had not been well studied in the methods based on preference of feasible solutions over infeasible ones. To cover this gap, [Wang et al. \(2015\)](#) proposed a differential evolution (DE)-based method to incorporate objective function information into the feasibility rule for constrained evolutionary optimization. In this method, after generating an offspring for each parent in the population using DE, the well-known feasibility rule is used to compare the offspring and its parent. The feasibility rule prefers constraints to objective function. Also the information of objective function is utilized to generate offspring in DE. This process allows this method to achieve balance between constraints and objective function.

Some methods also try to solve constrained optimization problems using multi-swarm approaches. These methods evolve multiple populations in a parallel manner to converge to the optimal solutions which handle constraints. [Wang and Cai \(2009\)](#) proposed a hybrid multi-swarm PSO (HMPSO) to solve constrained optimization problems. At each generation, the swarm is first split into several sub-swarms and the personal best of each particle is updated by DE. HMPSO uses the feasibility-based rule to compare particles in the swarm. [Liu et al.](#) proposed a hybrid algorithm named PSO-DE in [Liu et al. \(2010\)](#). This algorithm integrates particle swarm optimization (PSO) with differential evolution (DE)

to solve constrained and engineering optimization problems. To handle constraints, the proposed method minimizes the original objective function as well as degree of constraint violation. In the evolution process of this method, two kinds of populations with the same size are used. In the initial step of the algorithm, a population is created randomly. At each generation, the population is sorted according to the degree of constraint violation in a descending order. Only the first half of pop are evolved by using Krohling and dos Santos Coelho's PSO ([Krohling and Santos 2006](#)). After evolving each particle, if its value violates the boundary constraint, violating variable value is reflected back from the violated boundary. After the PSO evolution, this method employs DE to update personal best of particles.

Recently, multi-objective optimization methods are used to solve constrained optimization problems. The main idea of the multi-objective optimization-based methods is converting constrained optimization problems into unconstrained multi-objective optimization problems and solving the converted problems using multi-objective optimization techniques ([Wang and Cai 2009](#)). [Venkatraman and Yen \(2005\)](#) proposed a two-phase constraint handling algorithm based on multi-objective optimization. In the first phase, the constraint optimization problem is done as a constraint satisfaction problem. The objective function is completely ignored in this phase. In the second phase, constraint satisfaction and objective optimization are both treated as an optimization problem with two objectives. The proposed method in [Angantyr et al. \(2003\)](#) introduces a combination of multi-objective optimization technique and penalty function as an approach for constraint handling. This method is very similar to penalty-based approaches. However, it borrows the ranking scheme from multi-objective optimization methods.

A constrained multi-objective method based on constrained dominance of solutions is proposed in [Deb et al. \(2000\)](#) by Deb. According to this method, an individual  $i$  is said to constrained-dominate an individual  $j$  if (1)  $i$  is feasible, while  $j$  is infeasible; (2) both  $i$  and  $j$  are infeasible and  $i$  has less constraint violation; or (3) both  $i$  and  $j$  are feasible and  $i$  dominates  $j$ . Feasible individuals constrained-dominate all infeasible individuals. The constraint violation level is used to compare two infeasible solutions. The drawback of this method is that it reduces the search ability of the algorithm because of the less flexibility of the constrained dominance operator. [Jimenez et al.](#) suggest an Evolutionary Algorithm of non-dominated Sorting with Radial Slots in [Min et al. \(2006\)](#), which uses the min-max formulation for constraint handling. In this method, feasible and infeasible individuals are evolved separately. Feasible individuals evolve toward Pareto optimality, while infeasible individuals evolve toward feasibility.

[Ray et al.](#) suggest using three different non-dominated rankings of the population based on the objective function

values, different constraints and combination of all objective functions, respectively, in Zitzler (1999). Using these rankings, the algorithm acts according to the predefined rules. Chafekar et al. (2003), propose two new approaches to solve constrained MOOPs. Their first method runs several GAs parallel with each GA optimizes one objective. Each GA exchanges information about its objective with other GAs. The second method uses a common population for all objectives and runs each objective sequentially on the population.

Cai and Wang (2006) proposed a MOO-based EA for constrained optimization, abbreviated as CW method. The optimization process in CW method includes two major components, (1) the population evolution model inspired by the proposal in Deb (2005), and (2) is the infeasible solution archiving and replacement mechanism which steers the population toward the feasible region. This approach only exploits Pareto dominance to compare the individuals. The main shortcoming of CW approach is that a trial-end-error process has to be used to choose suitable parameters. To overcome the above shortcoming, Wang and Cai (2012a) proposed an improved version of CW, called CMODE, which combines MOO with DE to deal with constrained optimization problems. The comparison of individuals in CMODE is based on MOO and DE serves as search engine. Compared with CW, CMODE has two main differences: (1) instead of using the simplex crossover, it uses DE as the search engine, and (2) CMODE has proposed a novel infeasible solution replacement mechanism based on MOO, which guides the population toward promising solutions and the feasible region.

Wang and Cai (2012b) proposed a dynamic hybrid framework, called DyHF, to solve constrained optimization problems. This framework composed of two main steps: global search model and local search model. In both search models, differential evolution utilized as the search engine and the Pareto dominance used in multi-objective optimization is used to compare the individuals. The above two steps are executed dynamically according to the feasibility proportion of the current population. Compared with other HCOEAs, DyHF has the following features: (1) the global and local search models are dynamically applied; (2) DE serves as the search engine; and (3) the parameter settings are kept the same for different problems. In an other research, Wang and Cai (2011) proposed a  $(\mu + \lambda)$ -DE and an improved adaptive trade-off model to solve constrained optimization problems. In the  $(\mu + \lambda)$ -DE,  $\mu$  parents are used to generate  $\lambda$  offsprings. Then these  $\mu$  parents and  $\lambda$  offsprings are combined to select  $\mu$  candidate individuals for the next generation. Each parent produces three offsprings using three different mutation strategies and a binomial crossover of DE. Moreover, the paper improves the current-to-best/1 strategy to further enhance the global exploration ability by exploiting the feasibility proportion of the last population. Also,

the paper proposed the improved adaptive trade-off model which includes three main situations: the infeasible situation, the semi-feasible situation, and the feasible situation. In each situation, a constraint handling method is designed based on the characteristics of the current population.

Woldesenbet et al. (2009), proposed a constraint handling technique for multi-objective EAs based on an adaptive penalty function and a distance measure. This algorithm will be referred to as Woldesenbet's algorithm in Sect. 8. The two functions vary dependent upon the objective function value and the sum of constraint violations of an individual. In this method, the modified objective functions are used in the non-dominance sorting to facilitate the search of optimal solutions not only in the feasible space but also in the infeasible regions. The search in the infeasible space is designed to exploit those individuals with better objective values and lower constraint violations.

### 3 Motivation

In this section, the main motivation of the proposed algorithm is described. The main novelty of this work is dividing population into feasible and infeasible populations and duality evolution in a parallel manner in populations. As mentioned earlier in Sect. 2, the major difference between constraint handling methods is how to deal with infeasible particles in the evolution process. Most of the proposed methods to handle constraints focus on using penalty functions dealing infeasible solutions. The main drawback of using penalty functions is that the near feasibility infeasible particles that will be converted to feasible ones after a minor movement will have very lower chance to survive and move toward feasibility by applying penalty values.

To overcome this drawback, the proposed method divides the search space into two non-overlapping spaces as constraint space and objective space. The infeasible particles are evolved in the constraint space toward feasibility and the feasible particles are evolved in objective space toward Pareto optimality. By infeasible particles evolution in constraint space, we do not need to manipulate the fitness vector of the infeasible particles using a penalty function. So, the method will be capable to use infeasible particles information properly in the evolution process.

### 4 PSO basics

Particle swarm optimization (PSO) is a stochastic global optimization method which has been initially proposed by Kennedy and Eberhart (1995). It is a population-based method and the evolution process in it is similar to social behavior of bird flocking. PSO utilizes a swarm (as a number of solutions), named particles, to move through a hyper

dimensional search space toward the most promising area for optimal solution(s). Given an N-dimensional problem, PSO associates each particle with a position vector  $\mathbf{x} = (x_1, x_2, \dots, x_N)$  and a velocity vector  $\mathbf{v} = (v_1, v_2, \dots, v_N)$ , which are iteratively adjusted in every dimension  $j \in \{1, \dots, N\}$ . Each particle moves toward a new position that is being calculated using its previous position, the positions of its own previous best performance **pbest** and the best previous performance of the whole swarm **gbest**.

$$v_j^{(t+1)} = \omega v_j^{(t)} + c_1 r_1 (\text{pbest}_j^{(t)} - x_j^{(t)}) + c_2 r_2 (\text{gbest}_j^{(t)} - x_j^{(t)}) \tag{2}$$

$$x_j^{(t+1)} = x_j^{(t)} + v_j^{(t+1)} \tag{3}$$

where  $c_1$  and  $c_2$  are two acceleration constants reflecting the weighting of cognitive and social learning, respectively,  $r_1$  and  $r_2$  are two distinct random numbers in  $[0, 1]$ , and  $\omega \in [0, 1]$  is the inertia factor.

When a particle discovers a position that is better than any it has seen previously, it stores the new position in the corresponding **pbest** as its personal best found history, and when a particle discovers a position that is better than any position found by all of the particles in the swarm in all previous generations, it stores the new position in the **gbest** as global best found history.

### 5 The proposed method

The proposed method is depicted in Fig. 1. The first population is initialized randomly at the start phase of algorithm. The initial population is divided into two non-overlapping populations, called *feasible population* and *infeasible population*, according to satisfying the constraints or not. At the next step, using the objective vector, the non-dominated particles in feasible population are selected as feasible repository set, which is denoted as  $R_F$ . Also, the elite particles in infeasible population are selected into infeasible repository set. This selection is done using the constraint vector for each infeasible particle. This set is denoted as  $R_I$  at remaining of the paper, too.

The particles in feasible population evolve in objective space toward Pareto optimality, and particles in infeasible population evolve in constraint space toward feasibility. Details of feasible population evolution and infeasible population evolution are described in detail in Sects. 6 and 7, respectively.

### 6 Feasible particles evolution in objective space

Figure 2 shows the evolution process for feasible population. At each generation and for each feasible particle  $S_F$  in feasi-

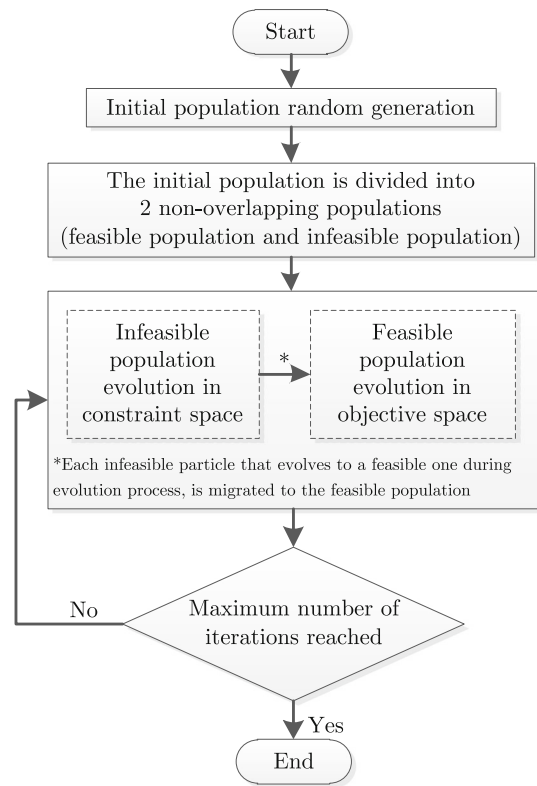


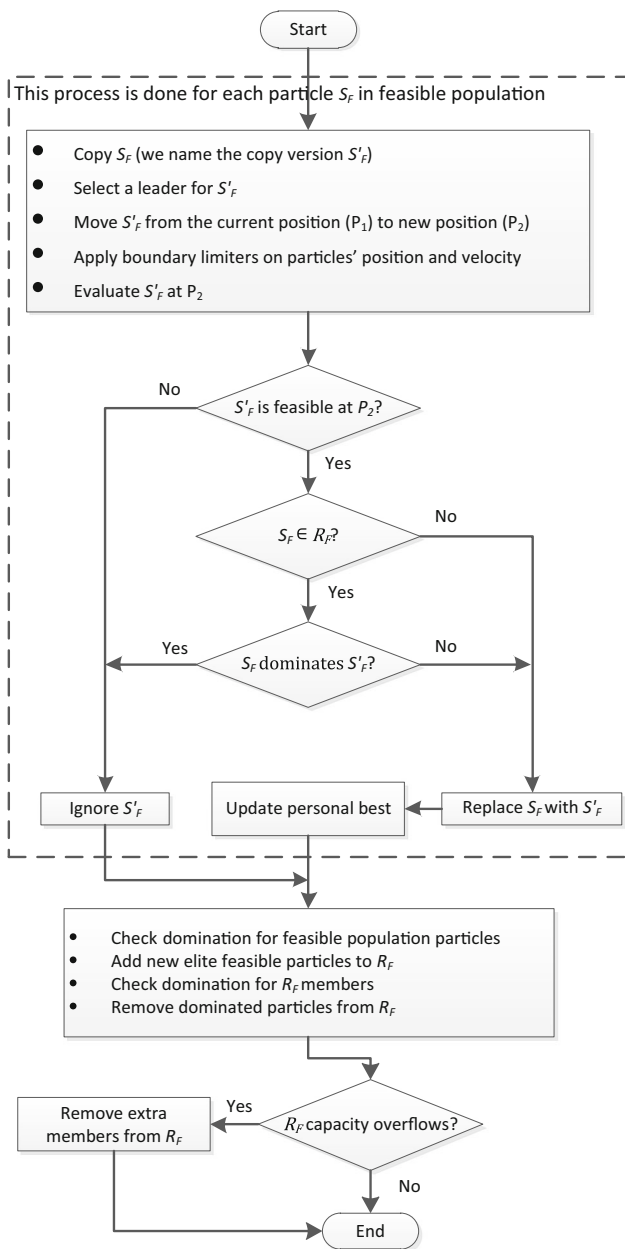
Fig. 1 Overview of the proposed method

ble population, first a copy of  $S_F$  is made (the copy particle is denoted as  $S'_F$ ). Then, to move  $S'_F$ , a leader is assigned to it from  $R_F$  and  $S'_F$  moves toward the leader. Leader selection is an important issue in the proposed algorithm, and it will be described in detail in Sect. 6.1.

After movement, if the copy particle ( $S'_F$ ) becomes infeasible, it is ignored. In fact, in this case the initial particle do not move. But if  $S'_F$  already remains a feasible particle, then two possible cases may occur. If  $S_F$  (the particle before movement) is a non-dominated solution, it is replaced by  $S'_F$  (the moved particle) if  $S_F$  does not dominates  $S'_F$ . This strategy is used in proposed algorithm to supply elitism. But if  $S_F$  is not a non-dominated solution, it is replaced by  $S'_F$ . Then best personal experience of the particle is updated.

After movement of all particles in feasible population, the non-dominated members of the new population are selected and are added to the feasible repository set ( $R_F$ ). After adding new members to  $R_F$ , its members should be checked again for domination and the dominated members should be eliminated.

Usually by increasing the generations numbers, the number of  $R_F$  members also increases. Then if this number exceeds the maximum capacity of  $R_F$ , some of its members must be eliminated. The  $R_F$  members that are in dense areas of the optimal Pareto front are suitable candidates for elim-

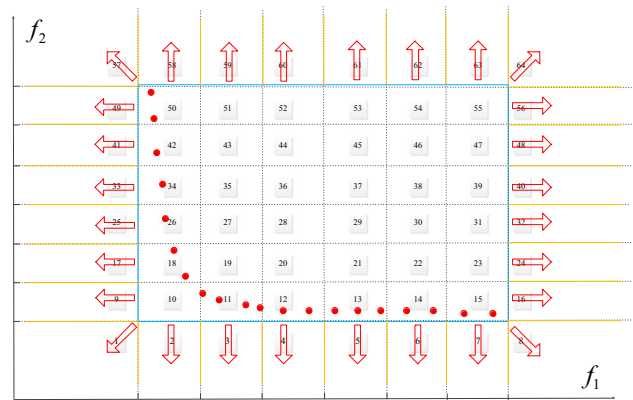


**Fig. 2** Feasible particles evolution in objective space

ination. The process of elimination of overflowed members from  $R_F$  is described in details in Sect. 6.2.

**6.1 Leader selection for feasible particles**

The proposed MOPSO in Coello et al. (2004) uses the adaptive grid method to leader selection. The motivation behind using grid method in MOPSO is position-based selection versus particle-based selection. In the grid method, the leader is selected randomly from the elite members in the grid cell with lowest crowding density. But this method is affected by a major drawback. Suppose a multi-objective optimization problem with  $M$  objective functions which each dimension



**Fig. 3** Grid method drawback. For  $d = 3$  and  $M = 2$ , the objective space is divided into 64 hyper-cubes

in the objective space is divided into  $N = 2^d$  sections. In this case, the objective space is divided into  $2^{d \times M}$   $M$ -dimensional hyper-cubes. In other words, the complexity of grid method is of exponential order and the linear modifications of  $d$  or  $M$  cause exponential modifications on the number of the grid hyper-cubes. For example, in a problem with 6 objective functions, that each objective space dimension is divided into 8 sections ( $d = 3$ ), the grid will have  $2^{3 \times 6} = 262144$  cells.

Figure 3 shows a simple example of using the grid method for leader selection. As it can be seen in Fig. 3, for  $d = 3$  and  $M = 2$ , the objective space is divided into 64 ( $=2^{2 \times 3}$ ) 2-dimensional cells.

The exponential growth of hyper-cubes number results small grid cells. Therefore, many of the cells will be empty of elite particles and most of the non-empty cells will contain one or two elite particles. It means that, in the high-dimensional problems, the selection of cells with lower density will usually have a similar result as the random selection from elite members and the desired effect of the grid method will be aborted. On the other hand, regarding that the size of hyper-cubes is controlled by parameter  $d$ , if we select a small value for  $d$  (i.e.,  $d = 2$ ), the hyper-cubes size will grow and this effect increases the probability of achieving a Pareto front with lower density and coverage. In summary, the grid method is suitable for the problems with small number of objectives and  $d$  (Schutze et al. 2011).

By tacking into account the drawback of grid method, this paper proposes different methods for leader selection. To move a feasible particle  $S_F$ , a leader should be assigned to it from the repository set  $R_F$ . For this purpose, the proposed method offers a priority-based selection algorithm in three levels. First, the crowding distance criterion that has been used in NSGA-II algorithm is used here (Deb et al. 2000). Crowding distance is introduced in NSGA-II to select repository members in the areas with lower crowds. Also this method proposes gravity distance. The gravity distance

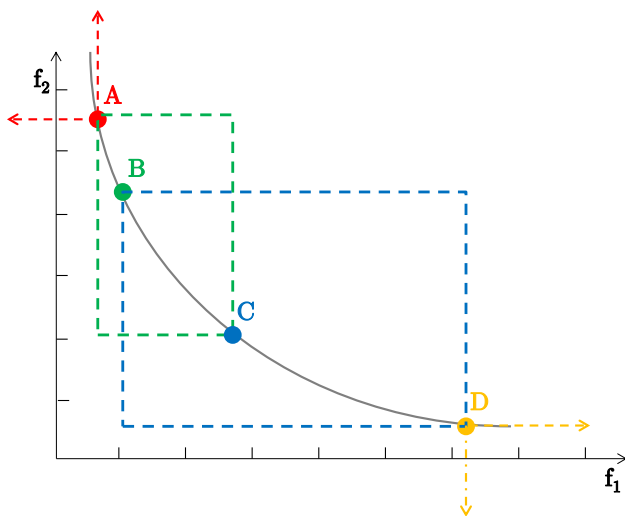


Fig. 4 Crowding distance for particles A, B, C and D

is a criterion to find the closest  $R_F$  member for the feasible particle  $S_F$ .

### 6.1.1 Crowding distance

In a non-dominated set of particles, to achieve an appropriate estimation of crowded density in the neighborhood of a specified particle  $x$ , the total distance between  $x$  and its two neighbor particles (one neighbor at each side) is calculated for all  $M$  objective functions. Figure 4 depicts crowding distance for particles A, B, C and D as rectangles. For a particle  $x$ , the crowding distance  $d_m$  (for  $m = 1, \dots, M$ ) measures hyper-cubic environment around  $x$  that has been created by its nearest neighbors on each side based on the  $m$ th objective. The crowding distance for extreme particles (i.e., A and D in Fig. 4) is defined as *infinite*. The crowding distance is completely formulated in Deb et al. (2000).

### 6.1.2 Gravity distance

The aim of introducing gravity distance is to offer a real value as a threshold, so if distance between particle  $x$  and an elite particle  $x^*$  (one of  $R_F$  members) is lower than the threshold value,  $x^*$  is assumed as a suitable candidate to be selected as the leader for  $x$ . In fact if  $x$  is inside a hypothetical orbit centered by  $x^*$ , it is gravitated by  $x^*$ . The gravity distance is the radius of this hypothetical orbit (see Fig. 5). The solution  $x$  is inside of a orbit centered by the optimal Pareto solution  $x^*$  and radius of gravity distance.

The first step to calculate the gravity distance is to obtain minimum and maximum values of each objective function as the known search space boundaries. It is done using Eqs. 4 and 5.

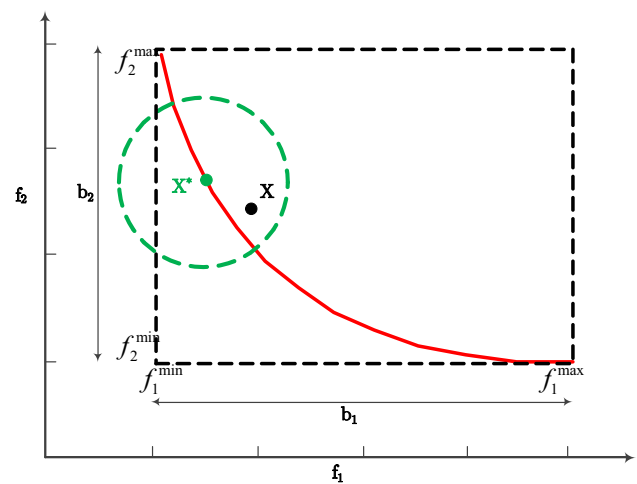


Fig. 5 Gravity distance of particle  $x$  from the elite particle  $x^*$

$$f_i^{\min} = \min_x f_i(x) \tag{4}$$

$$f_i^{\max} = \max_x f_i(x) \tag{5}$$

Having the lower and upper bounds of each objective function, the range of the known search space is computed using Eq. 6.

$$b_i = f_i^{\max} - f_i^{\min} \tag{6}$$

To calculate the gravity distance, the vector  $u$  is defined based on parameter  $a$  as follows:

Now, we define  $u$  to calculate gravity distance as Eq. 7.

$$u = a.(b_1, \dots, b_M) \quad 0.1 \leq a \leq 0.2 \tag{7}$$

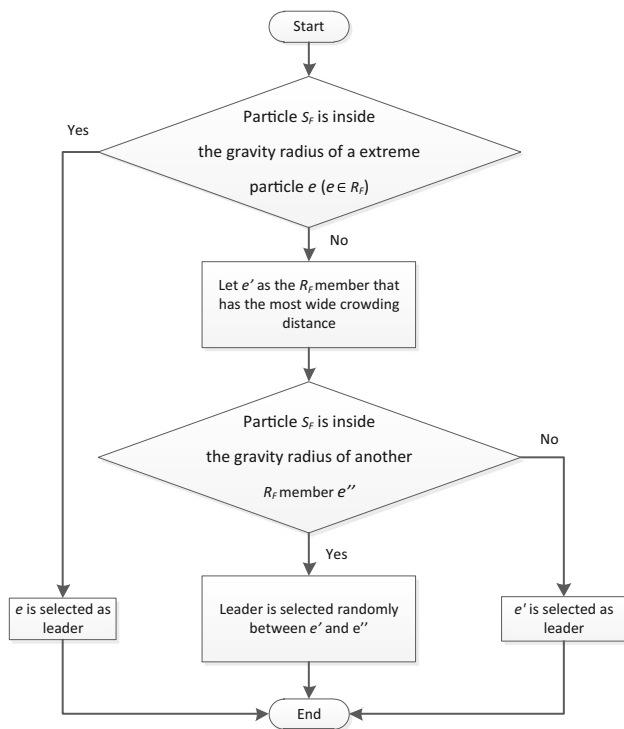
The gravity distance is defined as follows in Eq. 8.

$$r = \sqrt{\sum_{j=1}^M u_j^2} \tag{8}$$

If  $f(x) = (f_1(x), \dots, f_M(x))$  is assumed as the objective vector of particle  $x$  and  $f(x^*) = (f_1(x^*), \dots, f_M(x^*))$  be the objective vector of non-dominated solution  $x^*$ , using Eqs. 7 and 8, the normalized distance between  $x$  and  $x^*$  is calculated by Eq. 9.

$$d = \sqrt{\sum_{j=1}^M ((f_j(x^*) - f_j(x))/u_j)^2} \tag{9}$$

If  $d \leq \sqrt{M}$ , the proposed algorithm selects  $x^*$  as another candidate to be selected as leader for  $x$  because  $x$  is inside the orbit centered by an elite particle.



**Fig. 6** Leader selection for feasible particles

### 6.1.3 Select leader for feasible particles

In MOPSO algorithm, a particle for moving in search space has to select a leader (Coello et al. 2004). For this purpose, the proposed algorithm, presents a priority-based algorithm in three levels. Considering the importance of well distribution and full coverage of optimal Pareto front, extreme non-dominated particles (first and last particles in the non-dominated repository set) are placed at the first level of importance. Locating extreme optimal solutions in first priority level of the leader selection algorithm leads the population to find the optimal solutions on optimal Pareto front extremes. Also having uniform distribution on repository set is very important. So among other particles in  $R_F$ , the optimal solutions with maximum crowding distance are placed at the second priority level. At the next step, the proposed method prefers to select the closest  $R_F$  member as leader for each particle to avoid inefficient movements for a particle at its life time and emphasize to the local search. This selection results better local search for each particle around its position. So, the closest  $R_F$  member to the particle is placed at the third priority level in the proposed leader selection algorithm.

Figure 6 shows the proposed algorithm for leader selection for feasible particles. As Fig. 6 depicts, to select a leader for a feasible particle  $S_F$ , the proposed method first looks the  $R_F$  extreme particles. If  $S_F$  is inside the gravity distance of an elite extreme particle  $e$ , the  $e$  is selected as  $S_F$ 's leader. If not, at the second try, the algorithm finds the  $R_F$  member

$e'$  that has the most wide crowding distance. In this case, if  $S_F$  is inside the gravity distance of any  $R_F$  non-extreme member  $e''$ , then the leader is selected randomly between  $e'$  and  $e''$ . But if  $S_F$  is not inside the gravity distance of any  $R_F$  non-extreme member,  $e'$  is selected as leader.

In Fig. 7, the proposed method for leader selection for feasible particles is explained. In Fig. 7a, the feasible particle  $x$  is located inside the gravity radius of the extreme optimal solution  $x_1^*$ . The gravity radius for  $x_1^*$  is illustrated as dotted circle centered by  $x_1^*$  in Fig. 7a. By taking into account the leader selection algorithm (see Fig. 6), the extreme optimal particle  $x_1^*$  is selected as the leader for  $x$  in Fig. 7a. In Fig. 7b, the particle  $x$  is not inside the gravity radius of any extreme  $R_F$  member ( $x_1^*$  and  $x_5^*$ ). Among non-extreme  $R_F$  members ( $x_3^*$ ,  $x_4^*$  and  $x_4^*$ ), the particle  $x_4^*$  which has the most wide crowding distance is selected as default leader. Because particle  $x$  is out of the gravity distance of its closest non-dominated particle ( $x_3^*$ ), the default non-dominated particle  $x_4^*$  is selected as leader for  $x$ . Locating areas with lower density of optimal Pareto front is for density preservation and results in more diversity population. In Fig. 7c there are two candidate particles ( $x_3^*$  and  $x_4^*$ ) to be selected as leader for particle  $x$ . The first candidate particle is  $x_4^*$ , which has the greatest crowding distance among all non-dominated particles (except  $x_1^*$  and  $x_5^*$ ).

Taking into account the fact that  $x$  is inside the gravity distance of  $x_3^*$ , as the closest non-dominated particle, then  $x_3^*$  is selected as the second candidate to be selected as leader for  $x$ . The leader of  $x$  is selected randomly between  $x_3^*$  and  $x_4^*$ . The suitable threshold value to randomly selecting one of the two candidate solutions is one of the input parameters of the proposed method.

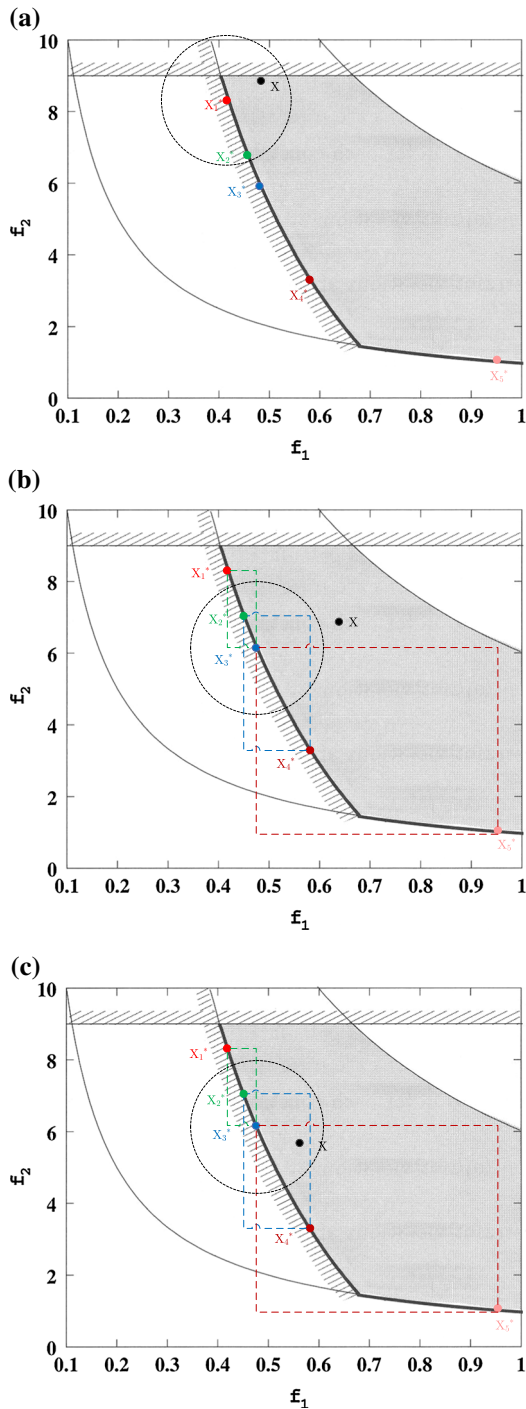
## 6.2 Eliminating extra particles from repository set

By increasing the number of generations, the number of  $R_F$  members also increases. On the other hand,  $R_F$  capacity is limited. So if the number of  $R_F$  members exceed from its maximum capacity, some of its members should be eliminated. The particles in crowded areas are suitable candidates to be eliminated. Therefore the particles with lowest crowding distance are eliminated from  $R_F$ .

## 7 Infeasible particles evolution in constraint space

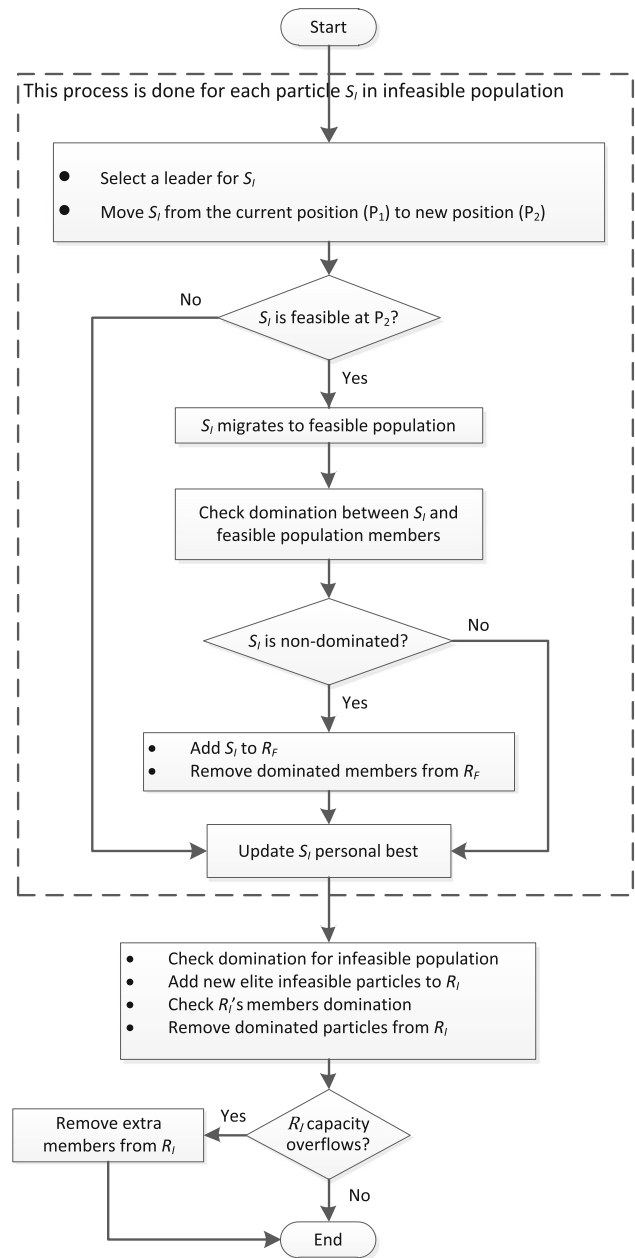
At each generation, parallel to feasible population particles movement, infeasible population particles move toward the new positions. Figure 8 shows the evolution process for particles in infeasible population. Similar to the feasible particles movement, a leader must be assigned to each infeasible particle before it moves. The leader selection algorithm for infeasible particles is explained in Sect. 7.1. After movement, if the





**Fig. 7** Leader selection for feasible particle  $x$  in objective space. **a**  $x$  is inside of the gravity distance of non-dominated particle  $x_1^*$  ( $x_1^*$  is selected as the leader for  $x$ ) **b**  $x$  is outside of the gravity distance of the nearest non-dominated particle ( $x_4^*$  is selected as leader for  $x$ ) **c**  $x$  is inside of the gravity distance of the nearest non-dominated particle (the leader for  $x$  is selected randomly between  $x_3^*$  and  $x_4^*$ )

particle still remains infeasible, its best personal experience is updated. But if it becomes a feasible particle, it migrates to the feasible population. After migration, domination between the migrated particle and all particles in feasible population



**Fig. 8** Infeasible particles evolution in constraint space

should be evaluated. If the migrated particle is located on the optimal Pareto front at its migration time,  $R_F$  members should be checked again for dominance and its dominated members should be eliminated. Finally, the new position of the migrated particle is registered as its best personal experience.

When all infeasible particles moved, the non-dominated members of new infeasible population are added to the infeasible repository set  $R_I$  and then  $R_I$  members are checked for dominance and dominated members are eliminated. Also, if  $R_I$  members number exceeds its maximum capacity, its extra members are eliminated. Eliminating extra particles from the

infeasible repository set ( $R_I$ ) is done in a same manner as described for feasible repository set ( $R_F$ ) in Sect. 6.2.

### 7.1 Leader selection for infeasible particles

Similar to feasible particles movement in objective space, to move an infeasible particle in constraint space, a leader should be assigned to the infeasible particle to move toward it. As mentioned earlier in Sect. 6.1, the non-dominated particles in feasible population are determined using the objective vector. In a similar manner, the non-dominated particles in infeasible population are determined using the constraint violation vector. For this purpose, a constraint violation vector  $C(x)$  is calculated for each particle  $x$  in infeasible population at each generation using Eq. 10.

$$C_i(x) = \text{Violation}(x, i), \quad i = 1, \dots, q, q + 1, \dots, m \quad (10)$$

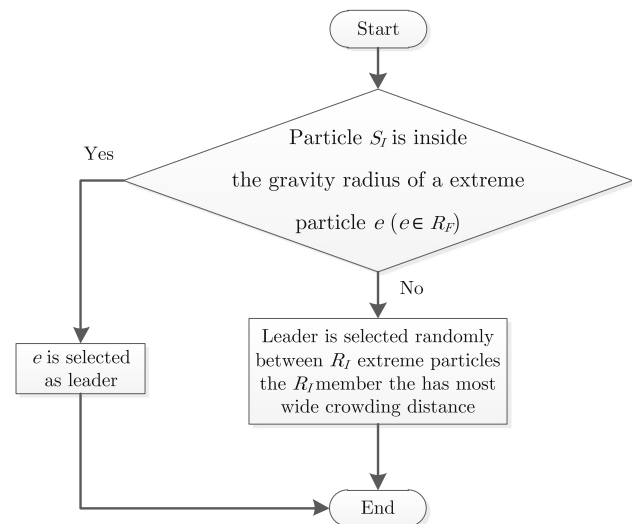
For each particle  $x$ , the constraint violation vector  $C(x)$  has  $m$  elements related to  $m$  constraints and  $\text{Violation}(x, i)$  shows the violation from  $i$ th constraint by  $x$ . Therefore,  $C_i(x)$  is the violation value from  $i$ th constraint by particle  $x$ . If  $C_i(x) > 0$ , it implies that the particle  $x$  does not satisfy the  $i$ th constraint ( $x$  violates  $i$ th constraint). However, the 0 value for  $C_i(x)$  shows that  $x$  satisfies the  $i$ th constraint. As a result, for a feasible particle, all of the elements in  $C$  should be 0 and for an infeasible particle, at least one element in  $C$  must have value greater than 0.

After calculating the constraint violation vector for each infeasible particle, as same as dominance comparison for feasible particles using objective vector, the dominance between particles in infeasible population is evaluated and the non-dominated particles are selected as infeasible population repository set  $R_I$ . The  $R_I$  members are candidates to be selected as leader for infeasible particles to move.

The goal of evolution in infeasible population is converting the infeasible particles to feasible ones. Now, there is 2 questions. Does the movement of infeasible particles toward elite infeasible particles lead these particles toward feasible space? And is there a way for leading the infeasible particles to feasible areas that are close to optimal Pareto front?

To answer these questions, the proposed method suggests leader replacement solution. Each infeasible particle has one constraint violation vector and one objective vector. Because of using the constraint violation vector to evaluate the particles in infeasible population, objective vector of infeasible particles has not been paid attention yet.

Figure 9 shows the proposed method to select leader for infeasible particles. Imagine an infeasible particle that has objective values close to optimal values for all objectives, but violates one of the constraints. This particle may be converted to a feasible optimal particle after a small movement. This



**Fig. 9** Leader selection for infeasible particles

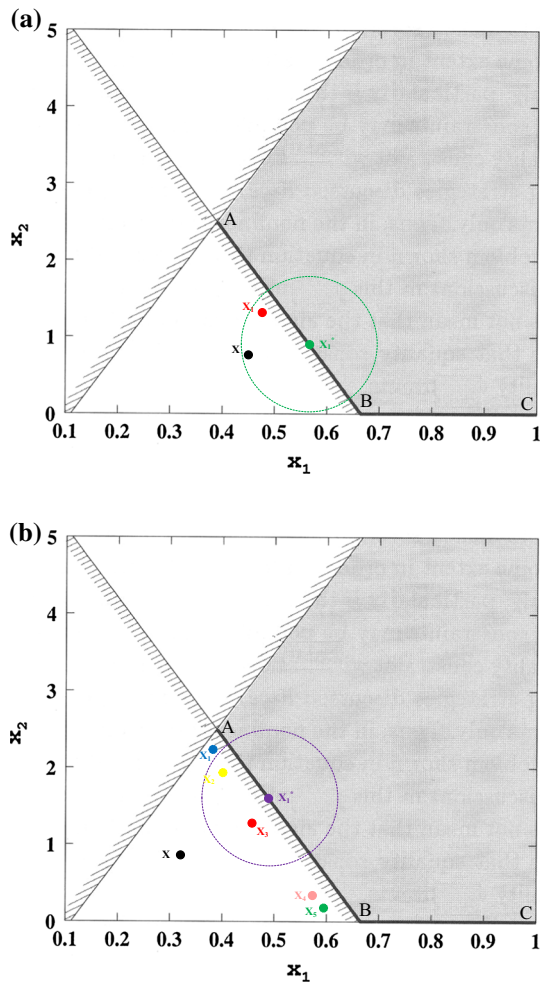
infeasible particle may be located inside the gravity radius of a particle in feasible population repository set ( $R_F$ ). So if the distance between the infeasible particle and a particle in  $R_F$  be less than the gravity radius of the  $R_F$  member, that  $R_F$  member is selected as the leader for the infeasible particle instead of selecting an elite infeasible particle. But if the infeasible particle is not located at the gravity radius of any of the  $R_F$  members, its leader is selected among  $R_I$  members. In this process, the algorithm is interested to select extreme particles or particles with higher crowding distance, respectively, in infeasible repository set.

Figure 10 shows an example of the proposed algorithm to select leader for an infeasible particle. The shaded areas show feasible space, and rest of figures belong to infeasible space. The points over lines AB and BC show optimal Pareto fronts. In Fig. 10a,  $x$  is an infeasible particle,  $x_1$  is the closest  $R_I$  member to  $x$ , and  $x_1^*$  belongs to  $R_F$ . As Fig. 10a shows,  $x$  is inside of the gravity distance of  $x_1^*$ . Therefore, instead of selecting  $x_1$ ,  $x_1^*$  is being selected as leader. This strategy is called *leader replacement* in this paper. In Fig. 10b,  $x$  is an infeasible particle,  $x_i$ s (for  $i = 1, \dots, 5$ ) are  $R_I$  members, and  $x_1^*$  is the closest  $R_F$  member to  $x$ . In this figure, the particle  $x$  is outside from the gravity distance of the closest  $R_F$  member. So,  $x_1$  and  $x_5$  (as extreme particles) and  $x_3$  (which has the most wide crowding distance in  $R_I$  members) are candidates to be selected as leader for  $x$ . Selection between these 3 candidates is done randomly.

Leader replacement acts as a bridge here. It creates a relation between infeasible particles in constraints space and feasible particles in objective space.

## 8 Simulation and experimental results

In this section, the proposed method is applied to several constrained multi-objective benchmark problems, and the



**Fig. 10** Leader selection for infeasible particle  $x$  in constraint space. **a**  $x$  is inside of the gravity distance of a optimal Pareto solution (non-dominated feasible solution  $x_1^*$ ) **b**  $x$  is outside of the gravity distance of all of the Pareto optimal solutions

validity of the proposed method is demonstrated through computer simulation. The performance of the proposed algorithm was compared to two popular state-of-the-art algorithms, named Woldesenbet’s algorithm (Woldesenbet et al. 2009) and NSGA-II (Deb et al. 2000). Both quantitative and qualitative comparisons were made to validate the proposed algorithm. This section presents the results for running three algorithms on test benchmark and comparing the results.

### 8.1 Experimental setup

Fourteen benchmark problems have been selected to evaluate the performance of the algorithms. These problems are all minimization problems and are denoted as BNH (Binh and Korn 1997), SRN (Srinivas and Deb 1994; Chankong and Haines 1983), TNK (Tanaka et al. 1995), CTP1 (Deb 2001), CTP2 (Deb 2001), CTP3 (Deb 2001), CTP4 (Deb 2001), CTP5 (Deb 2001), CTP6 (Deb 2001), CTP7 (Deb 2001), CTP8 (Deb 2001), OSY (Deb 2001), CONSTR (Deb 2001) and Welded Beam Problem (Zitzler 1999).

Some of the test problems have continuous Pareto fronts (BNH, SRN, CTP1, CTP6, OSY, CONSTR and Welded Beam), while the remaining problems have disjoint Pareto fronts (TNK, CTP2-CTP5, CTP7 and CTP8). The problem characteristics for these test problems are summarized in Table 1.

For each CMOEA (the proposed algorithm, Woldesenbet’s algorithm and NSGA-II), each benchmark problem was ran 50 times and the results were analyzed using statistical performance metric measures. At all of the experiments, the population size was limited to 200 (for the proposed algorithm, the total capacity of feasible and infeasible populations was limited to 200 particles) and maximum number of gen-

**Table 1** Summary for the basic characteristics of the benchmark test problems used in this paper

Function name	Objective functions	Decision dimensions	Feasibility ratio ( $\rho$ )	Constraints			
				Inequality	Equality	Linear	Nonlinear
BNH	2	2	93.61	2	0	0	2
SRN	2	2	16.18	2	0	1	1
TNK	2	2	5.09	2	0	0	2
CTP1	2	2	99.58	2	0	0	2
CTP2	2	2	78.65	1	0	0	1
CTP3	2	2	76.85	1	0	0	1
CTP4	2	2	58.17	1	0	0	1
CTP5	2	2	77.54	1	0	0	1
CTP6	2	2	0.40	1	0	0	1
CTP7	2	2	36.68	1	0	0	1
CTP8	2	2	17.83	2	0	0	2
OSY	2	6	3.25	6	0	4	2
CONSTR	2	2	52.52	2	0	2	0
Welded beam	2	4	18.67	5	0	1	4

erations was set to 100. As it can be seen in Table 1, the feasibility ratio is different in various problems. The feasibility ratio is determined experimentally by calculating the percentage of feasible solutions among 1000000 randomly generated solutions (Venkatraman and Yen 2005).

For qualitative comparison, the plots of final non-dominated fronts that were obtained from the same initial population are presented (Fig. 11). The quantitative comparison was performed using hyper-volume indicator ( $I_H$ ) and additive epsilon indicator ( $I_{\varepsilon+}$ ). These two Pareto compliant performance metrics are able to measure the performance of algorithms with respect to their diversity preservation and dominance relations. A detailed discussion about these measures can be found in Zitzler et al. (2003) and Knowles et al. (2006). The quantitative comparisons are illustrated by statistical box plots, and the Mann–Whitney rank-sum test was implemented to evaluate whether the difference in performance between two independent samples is significant or not (Knowles et al. 2006).

## 8.2 Comparative study

The performance metric for hyper-volume indicator ( $I_H$  value) was computed for each CMOEA over 50 independent runs. Fig. 12 presents the box plots of  $I_H$  indicator found in all CMOEAs, in which 1 is denoted by the proposed algorithm, 2 as the Woldesenbet's algorithm and 3 as the NSGA-II. Higher  $I_H$  value indicates the ability of the algorithm to dominate a larger region in the objective space. Figure 12 shows that the proposed algorithm has better  $I_H$  values for the test functions BNH, SRN, TNK, CTP1, CTP2, CTP4, CTP5, CTP6, CTP7, CTP8 and CONSTR. The proposed algorithm and Woldesenbet's algorithm showed comparable  $I_H$  values for test functions OSY and Welded Beam. Also, all three algorithms showed comparable  $I_H$  values for test function CTP3.

Woldesenbet's algorithm showed better  $I_H$  value than NSGA-II for test functions CTP4, CTP5, CTP7, OSY and Welded Beam. On the other hand, NSGA-II showed better  $I_H$  value than Woldesenbet's algorithm for test functions BNH, SRN, CTP1, CTP6 and CONSTR. And finally, the  $I_H$  values for Woldesenbet's algorithm and NSGA-II are very close for test functions TNK, CTP2, CTP3 and CTP8.

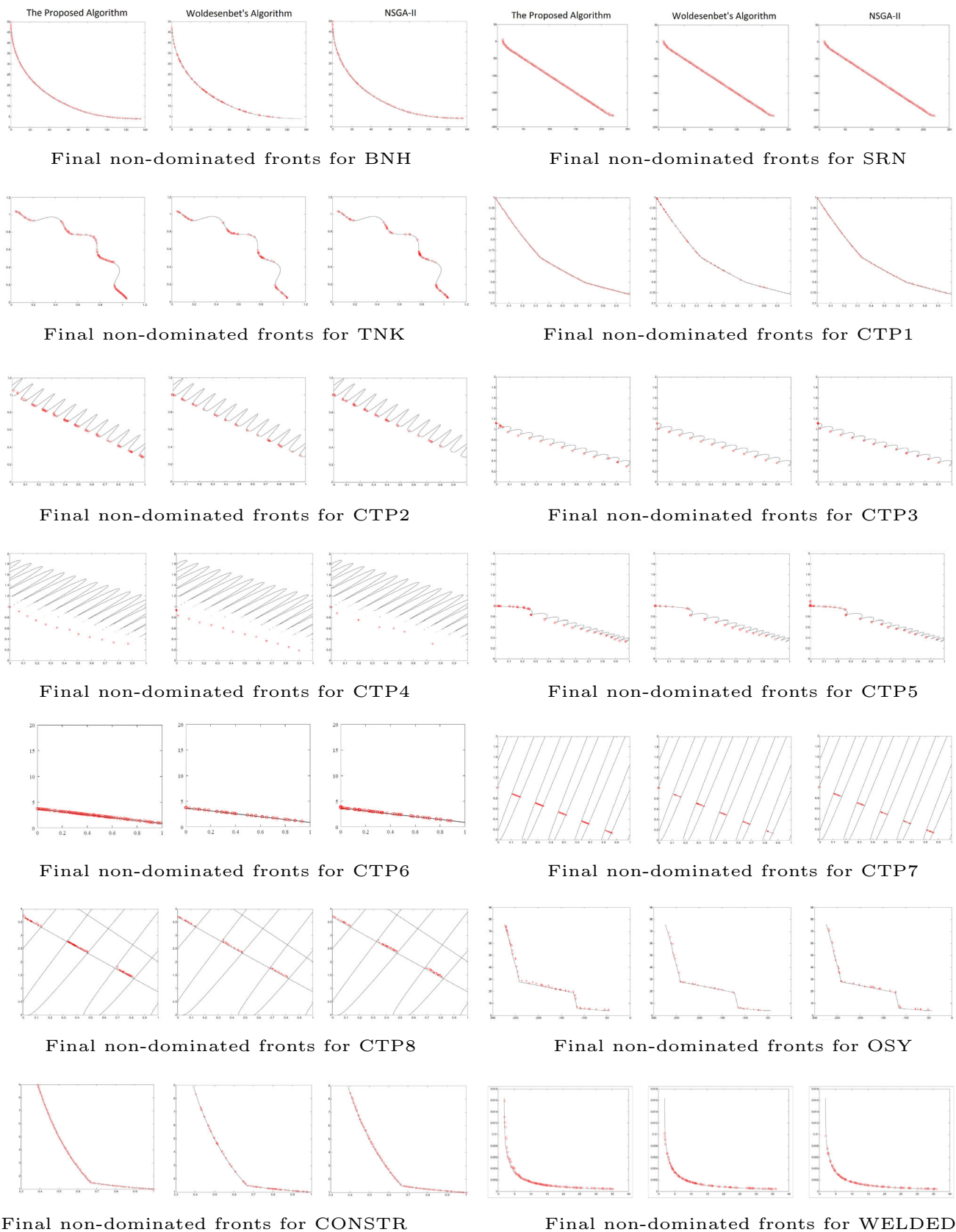
In some of the problems shown in Fig. 12, it is hard to determine whether the proposed algorithm is significantly better than the other CMOEAs since they attain close values. Hence, the Mann–Whitney rank-sum test was used to examine the distribution of the values. The tested results are presented in Table 2, and they indicate that the proposed algorithm's performance has a significant advantage compared to the distribution in Woldesenbet's algorithm in most test functions except CTP3, OSY and Welded Beam. Also the results indicate that the proposed algorithm's performance is better

compared to the distribution in NSGA-II in most test functions except CTP3.

Figure 13 illustrates the results of additive epsilon indicator using statistical box plots. This indicator, recommended in Zitzler et al. (2003), is capable of detecting whether a non-dominated set is better than another. There are two box plots for each test problem, i.e.,  $I_{\varepsilon+}(A, X_{1,2})$  and  $I_{\varepsilon+}(X_{1,2}, A)$ , in which algorithm A refers to the proposed algorithm, while algorithms 1 and 2 represent Woldesenbet's algorithm and NSGA-II, respectively. The results in Fig. 13 show that the proposed algorithm and two other CMOEAs are incomparable for BNH and SRN test functions because  $I_{\varepsilon+}(A, X_{1,2}) > 0$  and  $I_{\varepsilon+}(X_{1,2}, A) > 0$ . For the other test functions, the results are comparable because both  $I_{\varepsilon+}(A, X_{1,2})$  and  $I_{\varepsilon+}(X_{1,2}, A)$  has values close to 0 which implies that the performance of all 3 algorithms attain close values and it is hard to determine which algorithm has better performance than the other CMOEAs. Hence, the Mann–Whitney rank-sum test was used to examine the distribution of the values again. Table 3 shows the Mann–Whitney rank-sum test results for  $I_{\varepsilon+}$ . The results in Table 3 implies that  $I_{\varepsilon+}$  the proposed algorithm is able to find the well-extended, well-spread, and near-optimal solutions for test functions TNK, CTP2, CTP3, CTP4, CTP5, CTP6, CTP7, CTP8, OSY and Welded Beam compared to both Woldesenbet's algorithm and NSGA-II. Also Table 3 shows that the optimal Pareto fronts for proposed algorithms has better distribution than Pareto fronts for Woldesenbet's algorithm for test function CTP1. In other test functions, it cannot be claimed that the proposed algorithm has better  $I_{\varepsilon+}$  distribution than the other CMOEAs; but it does not confirm that the results for other CMOEAs is better than the proposed algorithm's results.

## 9 Conclusion

In this paper, a new constraint handling algorithm for multi-objective problems was introduced. The proposed method extends a previously proposed MOPSO algorithm to handle constraints. The particles population was divided into two non-overlapping populations, infeasible population and feasible population. At each generation, in a parallel process, each population was evolved independent from the other one. The infeasible population particles were evolved in the constraint space toward feasibility and then migrate to the feasible population. The particles in feasible population were also evolved in the objective space toward Pareto optimality. To select leader for feasible particles, a priority-based method in three levels was proposed and for infeasible particle, an elitism-based method was integrated with a leader replacement approach. The proposed algorithm was tested on 14 constrained multi-objective optimization benchmark problems, and its results were compared with two state-



**Fig. 11** Final non-dominated fronts plots for all test functions by the proposed algorithm (*on the left*), Woldesenbet's (*in the middle*), and NSGA-II (*on the right*)



**Fig. 12** Box plot of hyper-volume indicator ( $I_H$  values) for all test functions by algorithms 1-3 represented (in order): the Proposed algorithm, Woldesenbet's, and NSGA-II

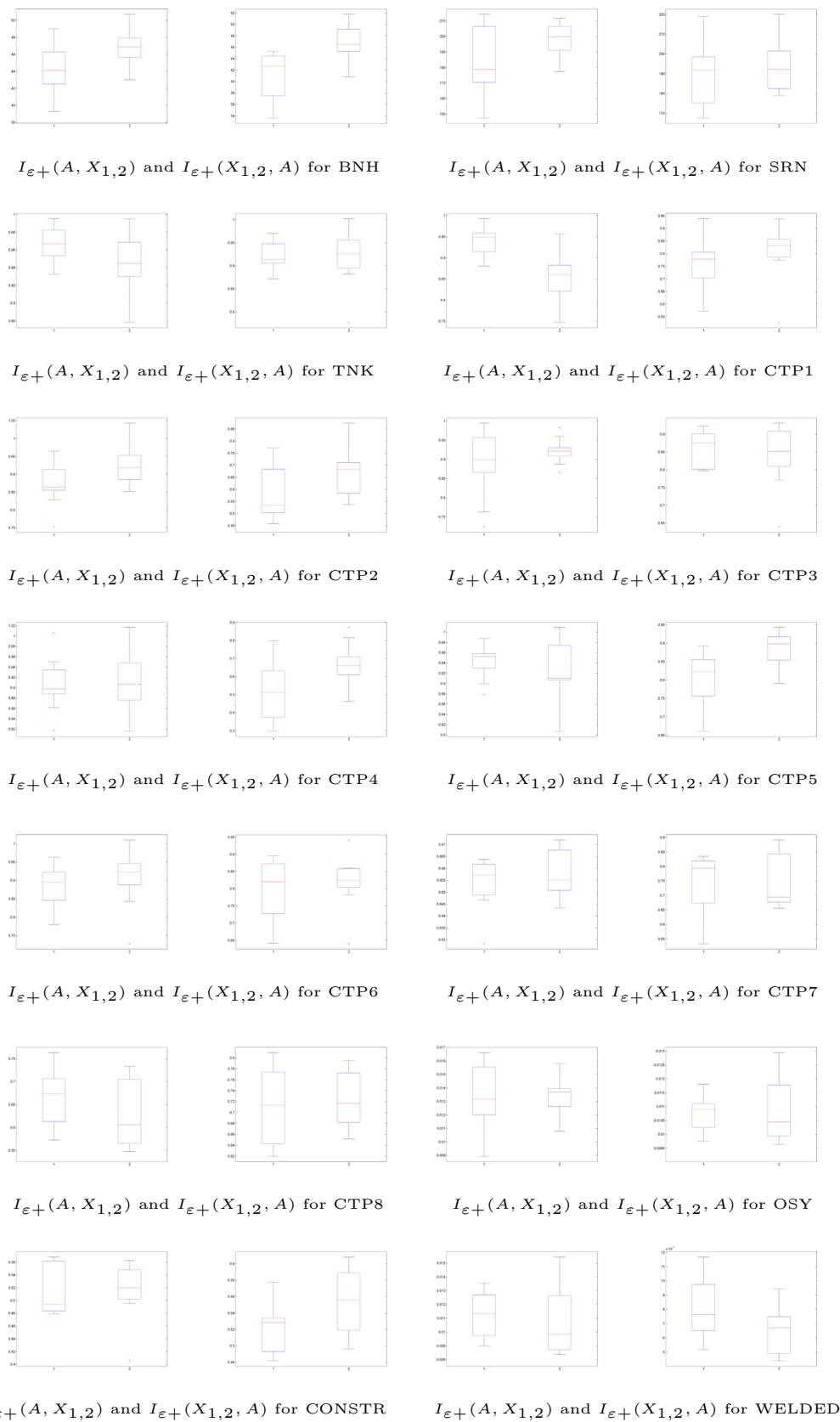
**Table 2** The distribution of  $I_H$  values tested using Mann–Whitney rank-sum test (Knowles et al. 2006)

Test functions	BNH	SRN	TNK	CONSTR	CTP1	CTP2	CTP3
$I_H(\text{Proposed}, \text{Woldesenbet})$	1.82E-4	1.70E-4	1.83E-4	1.83E-4	1.82E-4	5.83E-4	0.8155
$I_H(\text{Proposed}, \text{NSGA-II})$	1.35E-4	2.32E-3	1.83E-4	1.83E-4	4.39E-4	1.83E-4	0.9397
Test functions	CTP4	CTP5	CTP6	CTP7	CTP8	OSY	Welded Beam
$I_H(\text{Proposed}, \text{Woldesenbet})$	0.0164	0.0469	1.83E-4	2.57E-4	2.84E-3	0.0642	0.3267
$I_H(\text{Proposed}, \text{NSGA-II})$	5.49E-3	1.83E-4	2.57E-4	1.96E-4	0.0435	7.69E-4	0.0325

The table presents the  $\rho$ -values with respect to the alternative hypothesis (i.e.,  $\rho$ -value  $< \alpha = 0.05$ ) for each pair of the proposed algorithm and a selected CMOEA. The distribution of the proposed algorithm has significant differences than those selected CMOEA unless stated

of-the-art constraint handling multi-objective algorithms. Both qualitative and quantitative results were presented. For qualitative comparison, the plots of final non-dominated fronts were presented and the quantitative comparison was

performed using hyper-volume indicator ( $I_H$ ) and additive epsilon indicator ( $I_{\epsilon+}$ ). The experimental results showed that the proposed algorithm is highly competitive to solve the benchmark problems compared to the other algorithms.



**Fig. 13** Box plot of additive epsilon indicator ( $I_{\epsilon+}$  values) ('A' corresponds to the proposed algorithm, while 'X<sub>1,2</sub>' refers to Woldesenbet's and NSGA-II, respectively)

**Table 3** The distribution of  $I_{\varepsilon_+}$  values tested using Mann–Whitney rank-sum test (Knowles et al. 2006)

Test functions	BNH	SRN	TNK	CONSTR	CTP1	CTP2	CTP3
$I_{\varepsilon_+}(\text{Proposed}, \text{Woldesenbet})$	0.3721	0.4219	0.0162	0.1819	4.10E-4	1.83E-4	0.0395
$I_{\varepsilon_+}(\text{Proposed}, \text{NSGA-II})$	0.7311	0.6105	0.0271	0.0755	0.1233	1.83E-4	0.0287
Test functions	CTP4	CTP5	CTP6	CTP7	CTP8	OSY	Welded Beam
$I_{\varepsilon_+}(\text{Proposed}, \text{Woldesenbet})$	1.83E-4	3.30E-4	0.0311	2.51E-4	1.97E-4	2.11E-4	1.83E-3
$I_{\varepsilon_+}(\text{Proposed}, \text{NSGA-II})$	3.61E-4	0.0732	0.0109	2.76E-3	1.89E-4	3.67E-4	4.19E-4

The table presents the  $\rho$ -values with respect to the alternative hypothesis (i.e.,  $\rho$ -value  $< \alpha = 0.05$ ) for each pair of the proposed algorithm and a selected CMOEA. The distribution of the proposed algorithm has significant differences than those selected CMOEA unless stated

Indeed, it is able to obtain quality Pareto fronts for all of the test problems.

### Compliance with ethical standards

**Conflict of interest** The authors declare that they have no conflict of interest.

**Human and animal rights** This article does not contain any studies with human participants or animals performed by any of the authors.

### References

- Angantyr A, Andersson J, Aidanpaa J (2003) Constrained optimization based on a multiobjective evolutionary algorithm. *IEEE Congr Evol Comput* 3:1560–1567
- Back T, Hoffmeister F, Schwefel HP (1991) A survey of evolution strategies. In: *Proceedings of the fourth international conference on genetic algorithms*. Morgan Kaufmann, pp 2–9
- Bean J, Hadj-Alouane A (1992) A dual genetic algorithm for bounded integer programs. *Tech. Rep. 92-53*, The University of Michigan
- Binh TT, Korn U (1997) Mobes: a multiobjective evolution strategy for constrained optimization problems. In: *Proceedings of the third international conference on genetic algorithms*, pp 176–182
- Cai Z, Wang Y (2006) A multiobjective optimization-based evolutionary algorithm for constrained optimization. *IEEE Trans Evol Comput* 10(6):658–675
- Carreno Jara E (2014) Multi-objective optimization by using evolutionary algorithms: the p-optimality criteria. *IEEE Trans Evol Comput* 18(2):167–179
- Chafekar D, Xuan J, Rasheed K (2003) Constrained multi-objective optimization using steady state genetic algorithms. In: *Proceedings of the genetic and evolutionary computation conference*. Springer, Berlin, pp 813–824
- Chankong V, Haimes Y (1983) *Multiobjective decision making theory and methodology*. Elsevier, New York
- Coello CC (2006) Evolutionary multi-objective optimization: a historical view of the field. *IEEE Comput Intell Mag* 1(1):28–36
- Coello CC, Pulido G, Lechuga M (2004) Handling multiple objectives with particle swarm optimization. *IEEE Trans Evol Comput* 8(3):256–279
- Coello CC, Dhaenens C, Jourdan L (2010) *Advances in multi-objective nature inspired computing*, 1 edn. *Studies in computational intelligence*. Springer, Berlin
- Deb K (2000) An efficient constraint handling method for genetic algorithms. *Comput Methods Appl Mech Eng* 186(2):311–338
- Deb K (2001) *Multi-objective optimization using evolutionary algorithms*, 1 edn. Wiley, New York. <http://www.worldcat.org/isbn/047187339X>
- Deb K (2005) A population-based algorithm-generator for real-parameter optimization. *Soft Comput* 9(4):236–253
- Deb K, Pratap A, Agarwal S, Meyarivan T (2000) A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput* 6:182–197
- Farmani R, Wright J (2003) Self-adaptive fitness formulation for constrained optimization. *IEEE Trans Evol Comput* 7(5):445–455
- Fonseca CM, Fleming PJ (1993) Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In: *Proceedings of the fifth international conference in genetic algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, pp 416–423
- Hu W, Yen G (2015) Adaptive multiobjective particle swarm optimization based on parallel cell coordinate system. *IEEE Trans Evol Comput* 19(1):1–18
- Joines J, Houck C (1994) On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In: *Proceedings of the first IEEE conference on evolutionary computation*, pp 579–584
- Kennedy J, Eberhart R (1995) Particle swarm optimization. In: *Proceedings of the IEEE international conference on neural networks*, 1995, vol 4, pp 1942–1948
- Knowles J, Corne D (1999) Approximating the non-dominated front using the pareto archived evolution strategy. *Evol Comput* 8:149–172
- Knowles J, Thiele L, Zitzler E (2006) A tutorial on performance assessment of stochastic multiobjective optimizers (**revised**)
- Krohling RA, dos Santos Coelho L (2006) Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. *IEEE Trans Syst Man Cybern Part B Cybern* 36(6):1407–1416
- Liu H, Cai Z, Wang Y (2010) Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Appl Soft Comput* 10(2):629–640
- Liu HL, Gu F, Zhang Q (2014) Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE Trans Evol Comput* 18(3):450–455
- Lu H, Yen G (2003) Rank-density-based multiobjective genetic algorithm and benchmark test function study. *IEEE Trans Evol Comput* 7(4):325–343
- Min HQ, Zhou YR, Lu YS, Zhi JJ (2006) An evolutionary algorithm for constrained multi-objective optimization problems. In: *IEEE Asia-Pacific conference on services computing*, pp 667–670
- Mukhopadhyay A, Maulik U, Bandyopadhyay S, Coello Coello C (2014a) A survey of multiobjective evolutionary algorithms for data mining: part I. *IEEE Trans Evol Comput* 18(1):4–19
- Mukhopadhyay A, Maulik U, Bandyopadhyay S, CoelloCoello C (2014b) A survey of multiobjective evolutionary algorithms for data mining: part II. *IEEE Trans Evol Comput* 18(1):20–35
- Schutze O, Lara A, Coello Coello C (2011) On the influence of the number of objectives on the hardness of a multiobjective optimization problem. *IEEE Trans Evol Comput* 15(4):444–455
- Srinivas N, Deb K (1994) Multiobjective optimization using nondominated sorting in genetic algorithms. *Evol Comput* 2:221–248



- Takahama T, Sakai S (2005) Constrained optimization by applying the alpha; constrained method to the nonlinear simplex method with mutations. *IEEE Trans Evol Comput* 9(5):437–451
- Takahama T, Sakai S (2006) Constrained optimization by the epsilon constrained differential evolution with gradient-based mutation and feasible elite. In: *IEEE congress on evolutionary computation*, pp 1–8
- Tanaka M, Watanabe H, Furukawa Y, Tanino T (1995) GA-based decision support system for multicriteria optimization. *IEEE Int Conf Syst Man Cybern* 2:1556–1561
- Tessema BG, Yen GG (2006) A self-adaptive constrained evolutionary algorithm. In: *IEEE congress on evolutionary computation*, Vancouver, Canada, pp 246–253
- Venkatraman S, Yen G (2005) A generic framework for constrained optimization using genetic algorithms. *IEEE Trans Evol Comput* 9(4):424–435
- Wang Y, Cai Z (2009) A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems. *Front Comput Sci China* 3(1):38–52
- Wang Y, Cai Z (2011) Constrained evolutionary optimization by means of  $(\mu + \lambda)$ -differential evolution and improved adaptive trade-off model. *Evol Comput* 19(2):249–285
- Wang Y, Cai Z (2012a) Combining multiobjective optimization with differential evolution to solve constrained optimization problems. *IEEE Trans Evol Comput* 16(1):117–134
- Wang Y, Cai Z (2012b) A dynamic hybrid framework for constrained evolutionary optimization. *IEEE Trans Syst Man Cybern Part B Cybern* 42(1):203–217
- Wang Y, Wang BC, Li HX, Yen GG (2015) Incorporating objective function information into the feasibility rule for constrained evolutionary optimization. *IEEE Trans Cybern*. doi:[10.1109/TCYB.2015.2493239](https://doi.org/10.1109/TCYB.2015.2493239)
- Woldeesenbet Y, Yen G, Tessema B (2009) Constraint handling in multiobjective evolutionary optimization. *IEEE Trans Evol Comput* 13(3):514–525
- Yen G, Haiming L (2003) Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation. *IEEE Trans Evol Comput* 7(3):253–274
- Yen G, He Z (2014) Performance metric ensemble for multiobjective evolutionary algorithms. *IEEE Trans Evol Comput* 18(1):131–144
- Yeniay A (2005) Penalty function methods for constrained optimization with genetic algorithms. *Math Comput Appl* 10(6):45–56
- Zitzler E (1999) Evolutionary algorithms for multiobjective optimization: methods and applications. Institut für Technische Informatik und Kommunikationsnetze Computer Engineering and Networks Laboratory
- Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Trans Evol Comput* 3(4):257–271
- Zitzler E, Thiele L, Laumanns M, Fonseca C, Fonseca V (2003) Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Trans Evol Comput* 7(2):117–132