Swarm Intelligence
https://doi.org/10.1007/s11721-019-00170-1

A simplified multi-objective particle swarm optimization algorithm

Vibhu Trivedi[1] · **Pushkar Varshney**[2] · **Manojkumar Ramteke**[1]

Received: 13 May 2019 / Accepted: 8 July 2019

Abstract
Particle swarm optimization is a popular nature-inspired metaheuristic algorithm and has been used extensively to solve single- and multi-objective optimization problems over the last two decades. Several local and global search strategies, and learning and parameter adaptation strategies have been included in particle swarm optimization to improve its performance over the years. Most of these approaches are observed to increase the number of user-defined parameters and algorithmic steps resulting in an increased complexity of the algorithm. This paper presents a simplified multi-objective particle swarm optimization algorithm in which the exploitation (guided) and exploration (random) moves are simplified using a detailed qualitative analysis of similar existing operators present in the real-coded elitist non-dominated sorting genetic algorithm and the particle swarm optimization algorithm. The developed algorithm is then tested quantitatively on 30 well-known benchmark problems and compared with a real-coded elitist non-dominated sorting genetic algorithm, and its variant with a simulated binary jumping gene operator and multi-objective non-dominated sorting particle swarm optimization algorithm. In the comparison, the developed algorithm is found to be superior in terms of convergence speed. It is also found to be better with respect to four recent multi-objective particle swarm optimization algorithms and four differential evolution variants in an extended comparative analysis. Finally, it is applied to a newly formulated industrial multi-objective optimization problem of a residue (bottom product from the crude distillation unit) fluid catalytic cracking unit where it shows a better performance than the other compared algorithms.

Keywords Particle swarm optimization · Genetic algorithm · Multi-objective optimization · Residue fluid catalytic cracking unit

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s11721-019-00170-1) contains supplementary material, which is available to authorized users.

✉ Manojkumar Ramteke
ramtekemanoj@gmail.com; mcramteke@chemical.iitd.ac.in

[1] Department of Chemical Engineering, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India

[2] Department of Chemical Engineering, Indian Institute of Technology, Kanpur 208016, India

Published online: 15 July 2019

1 Introduction

In most real-life optimization problems, multiple objectives are commonly encountered. Very often, these are of conflicting nature, i.e., one improves only at the cost of others. Problems with two or more such objectives are termed as multi-objective optimization (MOO) problems. These result in multiple optimal solutions providing a trade-off among all the objectives. The advent of metaheuristic algorithms has completely revamped the procedure of solving MOO problems in the last two decades. Some of the leading examples of metaheuristic algorithms are genetic algorithms (GAs) (Holland 1975), particle swarm optimization (PSO) (Kennedy and Eberhart 1995), ant colony optimization (ACO) (Dorigo et al. 1996), and simulated annealing (SA) (Kirkpatrick et al. 1983). Despite their phenomenal success in solving MOO problems, the use of existing metaheuristic algorithms is hindered by premature convergence, slow convergence speed, complex algorithmic structures, and the requirement of tuning several parameters (Meng et al. 2014). Therefore, any step toward developing a metaheuristic algorithm which has a simple structure that escapes premature convergence while retaining fast convergence speed, is of a significant importance. In order to improve upon the aforementioned shortcomings of existing algorithms, a real-coded metaheuristic algorithm called *simplified multi-objective particle swarm optimization* (SMPSO) is developed in the present study.

In optimization, heuristics for guided and random moves are regularly used in different forms for the perturbation of solutions in various metaheuristics. For example, the real-coded elitist non-dominated sorting genetic algorithm (RNSGA-II) (Deb et al. 2002a) uses guided moves in the form of tournament selection (TS) and simulated binary crossover (SBX) (Deb and Agrawal 1994) operations and random moves in the form of a polynomial mutation operation (PLM) (Deb and Agarwal 1999). Similarly, PSO uses guided moves in the form of perturbations of a global best position (for a fully connected topology), personal best positions, and the addition of an inertia component (Kennedy and Eberhart 1995) which sets the velocity of a particle (solution) in a particular direction. The above moves also contribute to exploration since the new solutions are explored along the resultant direction of the global best, personal best, and inertia components. The bigger is the angle between the global best component and the other components (i.e., personal best component and the inertia component), the larger is the range of exploration (Reyes-sierra and Coello-coello 2006). Unlike these, random moves in the form of a turbulence operation (Reyes-sierra and Coello-coello 2006) which set the velocity of a particle in a random direction, an operation similar to a mutation operation adopted in evolutionary algorithms, are also used in different variants of multi-objective PSO to improve the performance. The ultimate aim of all these components is to provide a proper balance between exploration and exploitation. The present paper focuses on a *simplified* yet *optimized implementation* of these components used for the perturbation of solutions for *specifically* solving computationally intensive problems. For this purpose, a detailed qualitative analysis of guided and random moves used in RNSGA-II (i.e., simulated binary crossover and polynomial mutation) and PSO (i.e., perturbation by inertia, personal best position, and global best position) is performed. Based on this, simplified and effective forms of these moves are formulated for the perturbation of solutions to constitute SMPSO by *including* other commonly used components of multi-objective optimization such as initialization, ranking based on non-dominated sorting, and elitism from non-dominated sorting GA literature (Deb et al. 2002a).

The performance of SMPSO is verified quantitatively by solving 30 MOO benchmark problems from three well-known benchmark suites (Huband et al. 2006). Obtained results

for all problems are compared to those of RNSGA-II (Deb et al. 2002a), RNSGA-II with simulated binary jumping gene operator (RNSGA-II-SBJG) (Ramteke et al. 2015), and non-dominated sorting PSO (NSPSO) (Li 2003) and found to be superior, particularly for a restricted number of function calculations (i.e., number of generations or iterations × population or swarm size). This illustrates the suitability of the developed algorithm for solving computationally intensive problems. The performance of the proposed algorithm is also compared with four state-of-the-art multi-objective PSO variants, i.e., MOPSO (Coello-coello et al. 2004), TV-MOPSO (Tripathi et al. 2007), CCS-MOPSO (Kaveh and Laknejadi 2011), and MOLS-MOPSO (Xu et al. 2015) on six problems from the ZDT (Zitzler et al. 2000) and DTLZ (Deb et al. 2002b) benchmark suites and is found to be better. Next, SMPSO is applied to a MOO problem of residue fluid catalytic cracking (RFCC), which is an important process of the refining industry. The complex model equations (Varshney et al. 2015) present in the industrial problem restrict the number of function calculations that can be allowed for the optimization algorithm. The results for these problems demonstrate the usefulness of SMPSO in solving benchmark problems as well as complex industrial optimization problems under a strict computational budget.

The rest of the paper is organized as follows: A brief review of recent multi-objective PSO variants is presented in Sect. 2, a qualitative analysis of RNSGA-II and PSO is presented in Sect. 3, structure and qualitative analysis of the SMPSO algorithm are discussed in Sect. 4, the details of benchmark and industrial MOO problems are given in Sect. 5, the details of performance metrics and statistical tests are given in Sect. 6, results and discussion are provided in Sect. 7, and conclusions are provided in Sect. 8.

2 Review of recent variants of multi-objective PSO

PSO is a stochastic population-based optimization algorithm which mimics the social behavior of bird flocks. PSO was introduced about two decades ago by Kennedy and Eberhart (1995) for single-objective optimization (SOO) problems. Since then, several attempts to extend PSO for MOO problems have been reported (Reyes-sierra and Coello-coello 2006; Sengupta et al. 2019). The first attempt in this direction was made by Moore and Chapman (1999). They have used the concept of Pareto dominance to produce a list of the best solutions that guides the search. Coello-coello and Lechuga (2002) proposed a multi-objective PSO (MOPSO) which utilizes an external memory and a geography-based methodology to preserve the diversity. Subsequently, a large number of MOPSO variants were developed.

Several MOPSO variants were developed in the direction of improving the selection of personal and global best solutions for MOO problems. Tripathi et al. (2007) used a roulette wheel selection to select the global best solution from the set of non-dominated solutions. Liu et al. (2008) used a tournament niche scheme to select the global best solution and the concept of Pareto dominance to update the local best solutions. Wang and Yang (2009) ranked all solutions in a preference order to identify the global best solution. Elhossini et al. (2010) used tournament selection to select the global best solution from the external archive.

Improving the convergence of solutions to global Pareto optimal front in objective space while maintaining sufficient diversity is a challenging task. This direction is explored extensively in MOPSO studies. Coello-coello et al. (2004) implemented different clustering techniques to split the population into several swarms to maintain a better diversity of solutions in the objective space. Parsopoulos et al. (2004) developed a multi-swarm PSO, namely the vector evaluated PSO (VEPSO) in which each swarm was optimizing only one of the objective functions of the underlying MOO problem and the best experience of each swarm

was used to guide other swarms. Later on, VEPSO has been modified by several researchers (Grobler and Engelbrecht 2009; Harrison et al. 2013; Lim et al. 2013, 2014) with improved knowledge transfer strategies. Subsequently, several variants (Janson et al. 2008; Leong and Yen 2008; Liang et al. 2012; Hu et al. 2015) utilizing multiple sub-swarms have been reported. Liang et al. (2006) developed a comprehensive learning PSO (CLPSO) in which the velocity of a particle is updated using the personal best positions of the other particles. Cheng and Jin (2015) proposed a social learning PSO (SL-PSO) by incorporating a social learning mechanism in PSO. Other recent learning–strategy-based MOPSO variants include MOPSO with multiple search strategies (MMOPSO) (Lin et al. 2015), multiple learning PSO with space transformation perturbation (MLPSO-STP) (Yu et al. 2016), MOPSO based on sharing–learning and dynamic crowding distance (MOPSO-SDCD) (Peng et al. 2016), and multi-objective sorting-based learning PSO (MSLPSO) (Xu et al. 2019).

In addition to the above studies, PSO has also been hybridized with several global and local search techniques to produce a good blend of exploration and exploitation which is essential for a better performance. This includes DEPSO (Zhang and Xie 2003) and other hybrid variants (Das and Suganthan 2011) of PSO and a differential evolution (DE). Mirjalili and Hashim (2010) developed PSOGSA by combining a PSO with the gravitational search algorithm (GSA). Lim and Isa (2014) developed teaching and peer-learning PSO (TPLPSO) by combining PSO with a teaching- and learning-based algorithm (TLBO). Beheshti and Shamsuddin (2014) developed centripetal accelerated PSO (CAPSO) by combining PSO with Newton's law of motion. Khoshahval et al. (2014) developed parallel PSO-SA by combining PSO with SA. Xu et al. (2015) developed MOLS-MOPSO by combining PSO with multi-objective dichotomy line search (MOLS). Luo et al. (2015) developed MOPSO-EDA by combining PSO with an estimation of distribution algorithm (EDA).

From the aforementioned studies, it is clear that several local and global search strategies, and learning and parameter adaptation strategies have been included in PSO to improve its performance. However, most of these approaches involved the addition of components other than the three regular ones (i.e., perturbation by global best position, perturbation by personal best position and the inertia component) in the perturbation step along with their user-defined parameters. On the contrary, the aim of the present study is to develop a simplified multi-objective PSO variant to reduce the number of perturbation components while also improving upon the performance. For this purpose, a component-based qualitative analysis is performed for various components or operations used in PSO and RNSGA-II for perturbation. PSO and RNSGA-II are selected as model algorithms because these are among the most popular members of the respective classes of swarm-based and evolutionary algorithms. Apart from this, their components have been used in several other metaheuristics. This qualitative analysis provides a basis for informing on a simplified yet optimized use of different components in order to constitute an efficient hybrid algorithm for computationally intensive problems as described next.

3 Qualitative analysis of exploitation and exploration moves used in RNSGA-II and PSO

The genetic operators used in RNSGA-II are tournament selection, simulated binary crossover (SBX) and polynomial mutation (PLM) (Ramteke et al. 2015). Similarly, the perturbation steps used in PSO are the perturbation by a global best position, perturbation by a personal best position, and an addition of the inertia component (Kennedy and Eberhart 1995). The detailed

mathematical formulations of the genetic operators used in RNSGA-II and the perturbation steps used in PSO are given in Table S1 (supplementary material). In order to analyze the effect of each of these components (steps) of RNSGA-II and PSO on the perturbation of a solution in the presence of a current best solution (CB) (i.e., a solution with best objective value in the current population of the solutions), a hypothetical illustrative example of single-objective optimization is considered in which a single variable, X, is assumed for simplicity. The illustrative example is extended for a multi-objective optimization problem in the latter part of this section. The bounds on this variable, X^{Low} and X^{High}, are assumed to be 0 and 1, respectively. Besides this, the current best (CB) value of this variable is assumed 0.7. For single-objective optimization, a single CB value is assumed for X, whereas in the subsequent analysis for multi-objective optimization, multiple CB values (i.e., $\text{CB}_1 = 0.9$, $\text{CB}_2 = 0.7$, $\text{CB}_3 = 0.5$, and $\text{CB}_4 = 0.2$) are assumed. Though the mathematical form of the objective function is not given explicitly, it is assumed that the objective function has its optimum at $X = \text{CB} = 0.7$ for single-objective optimization and the solution with a variable value closer to 0.7 is considered to be better in relative comparison. For a multi-objective optimization, the objective functions are assumed to have optima at multiple current best solutions and the solution is better if it is relatively close to any one of these in variable space. In the analysis, the solutions are generated for a large number of random instances. These generated solutions are perturbed using the components of RNSGA-II and PSO one after another in the same sequence as that occurring in the algorithms. Perturbations due to each of these components on solutions are analyzed to capture their effect in the respective algorithms in terms of movement of solutions toward the CB (i.e., exploitation) and of the extent of perturbation (i.e., exploration). This is helpful in understanding the interlinking of different components used in RNSGA-II and PSO, providing insight into their relevance to single- and multi-objective optimization. In addition, it paves the way to devise a better hybrid algorithm for multi-objective optimization.

In order to demonstrate the analysis, 5000 random values of the variable X, X_k^{Old} ($k \in [1, 5000]$), are generated. The distribution of these random initial values is arranged in a decreasing order and plotted in Fig. 1. A near-straight line curve with no inflation was observed, which indicates an absolute randomness with a negligible bias and serves as a baseline for analyzing the effects of different components of RNSGA-II and PSO. These initial variable values, X_k^{Old}, are perturbed sequentially one after another in the given sequence of steps of RNSGA-II and PSO to give new values, X_k^{New}, at the end of all perturbation steps. The values obtained after any intermediate perturbation step are referred by X_k^{Step}, whereas the values after its predecessor step are referred by $X_k^{\text{Previous Step}}$. Further, if the variable values cross the given bounds in any of the steps, then these values are brought back to the given limits. After each step, the distribution function $\alpha_{k,\text{Step}}$ $[(X_k^{\text{Step}} - X_k^{\text{Previous Step}})/(X^{\text{High}} - X^{\text{Low}})]$ is obtained for all 5000 instances. These $\alpha_{k,\text{Step}}$ values capture the extent of perturbation added by the current step on the variable values obtained by the previous step, $X_k^{\text{Previous Step}}$. In this, an $\alpha_{k,\text{Step}}$ equal to 0 signifies no perturbation, whereas a value close to -1 or 1 signifies a complete perturbation. Furthermore, the new values after each step, X_k^{Step}, arranged in decreasing order, are plotted against the number of instances to obtain the X_k^{Step}-curve. Such rearranging of X_k^{Step} values in decreasing order filters the effect of random perturbations on X_k^{Step}-curve. Thus, the deviation of X_k^{Step}-curve from the previous $X_k^{\text{Previous Step}}$-curve demonstrates only the biased perturbations added in the respective step. After the perturbation, the solutions tend to move toward the best solutions leading to inflation points on X_k^{Step}-curve coinciding exactly at the current best values. The bias toward the current best solution added

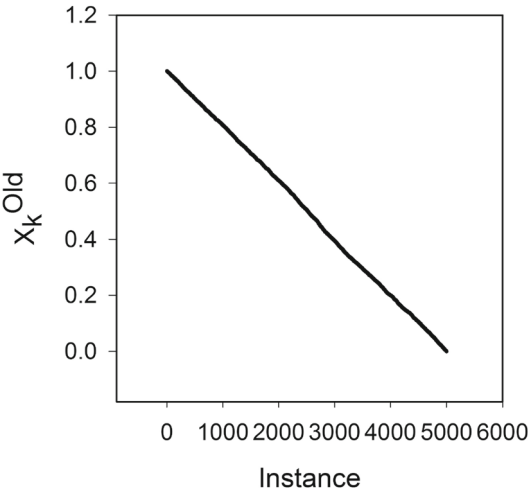

Fig. 1 Initial values of variable, X_k^{Old}, generated randomly and arranged in a decreasing order with the number of instances

in subsequent steps further increases the inflation in the X-curve marginally. A detailed analysis of components of RNSGA-II for single-objective optimization is presented next.

The effect of tournament selection (TS) (with two contenders competing at a time in the tournament) in RNSGA-II is evaluated first. This operation does not perturb the solutions as such, but copies the better solutions with higher probability (Deb et al. 2002a) and only these solutions subsequently undergo the SBX and PLM operations. Figure 2a shows the distribution of variable values (X_k^{TS}) after a tournament selection for the given single-objective illustrative example. The distribution illustrates that the values of the variable are biased toward the CB of 0.7 which is confirmed by an inflation exactly at 0.7 in contrast to the initial random values shown in Fig. 1, where the values represent a straight line (with no inflation). Since no perturbation occurs in this operation, $\alpha_{k,\text{TS}}\ [(X_k^{\text{TS}} - X_k^{\text{Old}})/(X^{\text{High}} - X^{\text{Low}})]$ is exactly 0. This population of preferred solutions, commonly referred as the mating pool population, undergoes SBX with a probability (P_{cross}) of 0.9 and index $\eta_{\text{cross}} = 20$ (Ramteke et al. 2015). In this operation, the extent of the perturbation in the new values (X_k^{SBX}) is maintained at a local level (i.e., the new perturbed values of variables are very close to the previous values) as indicated by $\alpha_{k,\text{SBX}}\ [(X_k^{\text{SBX}} - X_k^{\text{TS}})/(X^{\text{High}} - X^{\text{Low}})]$ varying close to 0 in Fig. 2b. It is to be noted that $\eta_{\text{cross}} = 1$ results in a random perturbation, whereas higher values of η_{cross} will increasingly favor local perturbations around the current values of the variable (Deb and Agrawal 1994). This operation does not add any additional bias toward the CB as both the parents are randomly picked from the mating pool. Therefore, the inflation in the variable curve (X_k^{SBX}-curve) is maintained at the same level as shown in Fig. 2a and has not been shown separately for a better readability.

The subsequent perturbation step is a PLM operation. In this operation, a PLM index, $\eta_{\text{mut}} = 20$ (Ramteke et al. 2015) and a probability (P_{mut}) of 0.1 [instead of (1/number of variables) as only one variable is perturbed] are assumed. For this operation, an obtained variable curve (X_k^{New}-curve) is the same as that of Fig. 2a and $\alpha_{k,\text{PLM}}\ [(X_k^{PLM} - X_k^{\text{SBX}})/(X^{\text{High}} - X^{\text{Low}})]$ vary close to zero signifying no additional bias toward the best solution in this step, while perturbation is maintained at the local level. Though the SBX and PLM do not add any bias toward the best value ($X = \text{CB} = 0.7$), the overall distribution of X_k^{New} is biased toward the best value due to the effect of the tournament selection operation as shown in Fig. 2d. Furthermore, the overall perturbation is maintained at a local level as indicated by $\alpha_{k,\text{OV}}\ [(X_k^{\text{New}} - X_k^{\text{Old}})/(X^{\text{High}} - X^{\text{Low}})]$ varying close to 0.

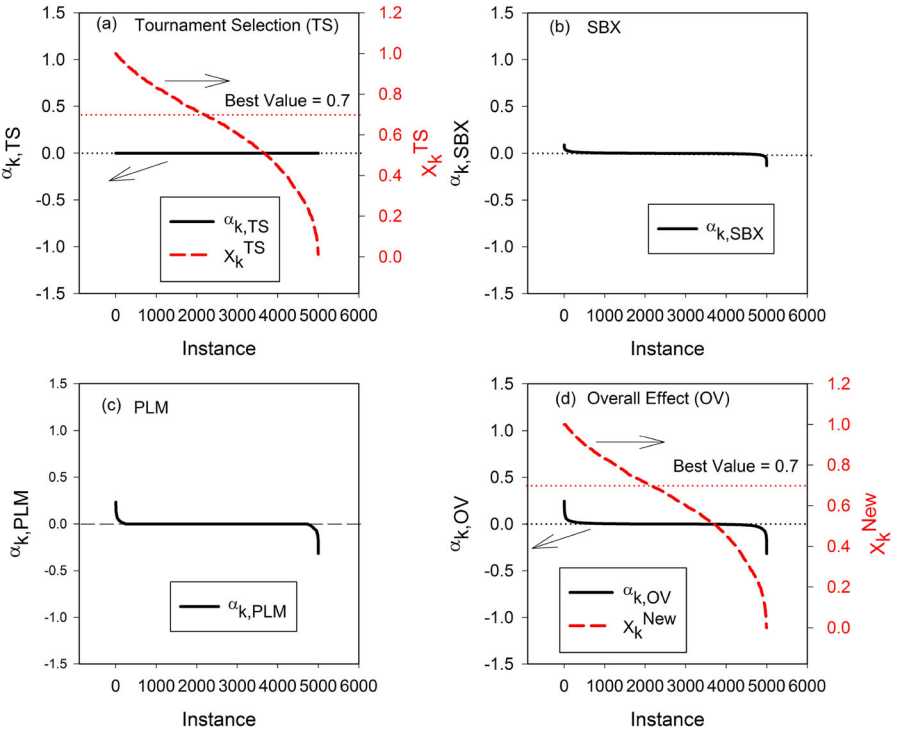

Fig. 2 Effect of **a** tournament selection (TS), **b** simulated binary crossover (SBX), **c** polynomial mutation (PLM), and **d** their overall (OV) effect on perturbation of the variable is shown in terms of the extent of perturbation ($\alpha_{k,\text{Step}}$; Step = TS, SBX, PLM, and OV) and movement of variable values [X_k^{Step}; Step = TS and New (values after all perturbation components)] toward the current best solution in RNSGA-II (*Note* The Y-axes for corresponding curves of $\alpha_{k,\text{Step}}$ and X_k^{Step} are indicated by arrows)

In a single-objective PSO, the solutions are perturbed by the global best position (gbest), the personal best position (pbest), and the addition of the inertia component. The generic term *personal best solution* refers to the best position achieved by a solution over its personal history of perturbation, whereas *global best solution* (Kennedy and Eberhart 1995) refers to the best solutions obtained so far (i.e., current best solution) in a fully connected topology (Reyes-sierra and Coello-coello 2006). For the perturbation by global best position, a given solution is perturbed using its stochastic weighted difference with the current best solution. For the perturbation by personal best position, the stochastically weighted difference between the underlying solution and its personal best solution is used. However, the inertia component adds the effect of previous perturbations. The effect of each of these components is evaluated in terms of the extent of perturbation ($\alpha_{k,\text{Step}}$) and the exploitation of the current best solution for the given illustrative example. The new variable values, X_k^{gbest}, obtained after the perturbation by global best (gbest) position show (see Fig. 3a) a bias toward the current best value ($X = \text{CB} = 0.7$) as demonstrated by an inflation at a value equal to 0.7 as compared to the initial random values shown in Fig 1. Further, Fig. 3 shows the horizontal line at the top (i.e., at $X = 1$) on the X_k^{gbest}-curve, which indicates that the values of the variable are brought back to their upper bound for the cases where the perturbation leads to the values beyond 1. In addition to the bias toward the current best solution, this step also adds the perturbation as illustrated by the distribution of $\alpha_{k,\text{gbest}}$ $[(X_k^{\text{gbest}} - X_k^{\text{Old}})/(X^{\text{High}} - X^{\text{Low}})]$ values as shown in Fig. 3a.

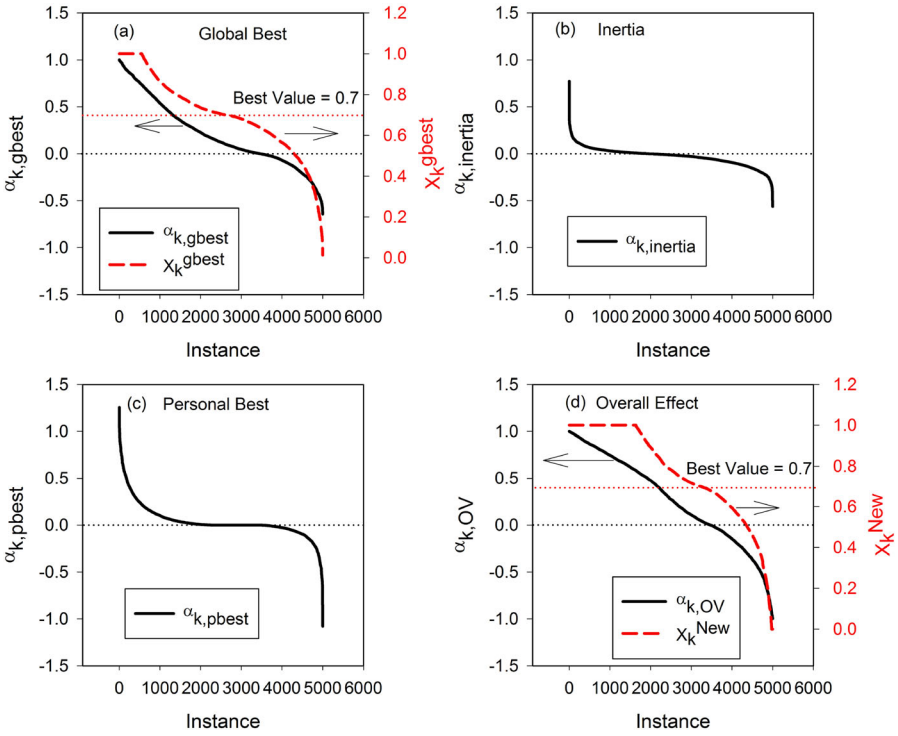

Fig. 3 Effect of **a** perturbation by global best position (gbest), **b** addition of inertia component, **c** perturbation by personal best position (pbest), and **d** their overall (OV) effect on perturbation of the variable is shown in terms of the extent of perturbation ($\alpha_{k,\text{Step}}$; Step = gbest, inertia, pbest, and OV) and movement of variable values [X_k^{Step}; Step = gbest and New (values after all perturbation components)] toward the current best solution in single-objective PSO (*Note* The Y-axes for corresponding curves of $\alpha_{k,\text{Step}}$ and X_k^{Step} are indicated by arrows)

In contrast to the perturbation by a global best position, the inertia component adds only the perturbation in new variable values, X_k^{inertia} and does not add any bias toward the current best value. Therefore, only $\alpha_{k,\text{inertia}}$ $[(X_k^{\text{inertia}} - X_k^{\text{gbest}})/(X^{\text{High}} - X^{\text{Low}})]$ values indicating the variable perturbation are shown in Fig. 3b as the X_k^{inertia}-curve is expected to be nearly the same as X_k^{gbest}-curve shown in Fig. 3a. Similarly, the perturbation by personal best (pbest) position primarily adds to the extent of perturbation with a very little bias toward the current best value ($X = \text{CB} = 0.7$). This is because, the personal best positions tend to move toward the global best position over the iterations. However, this movement is quite slow. Owing to this, the effect of this additional bias on the X_k^{New}-curve is expected to be very low and therefore the X_k^{New}-curve for this step is also similar to that shown in Fig. 3a. For a better readability, only the extent of perturbation indicated by $\alpha_{k,\text{pbest}}$ $[(X_k^{\text{pbest}} - X_k^{\text{inertia}})/(X^{\text{High}} - X^{\text{Low}})]$ is shown in Fig. 3c. The overall extent of perturbation $[\alpha_{k,\text{OV}} = (X_k^{\text{New}} - X_k^{\text{Old}})/(X^{\text{High}} - X^{\text{Low}})]$ includes the effect of all components and shows a substantial perturbation (i.e., $\alpha_{k,\text{OV}}$ is nonzero in Fig. 3d). Further, the X_k^{New}-curve shows a bias toward the best value i.e., the X_k^{New}-curve shows the inflation at the value of 0.7 in Fig. 3d, which is similar to the global best step. Subsequently, due to the addition of the inertia component and perturbation by personal best position, the horizontal plateau at the top (at $X_k^{\text{New}} = 1$), signifying the number of solutions crossing the upper bounds, also increases in the length as compared to that shown in Fig. 3a.

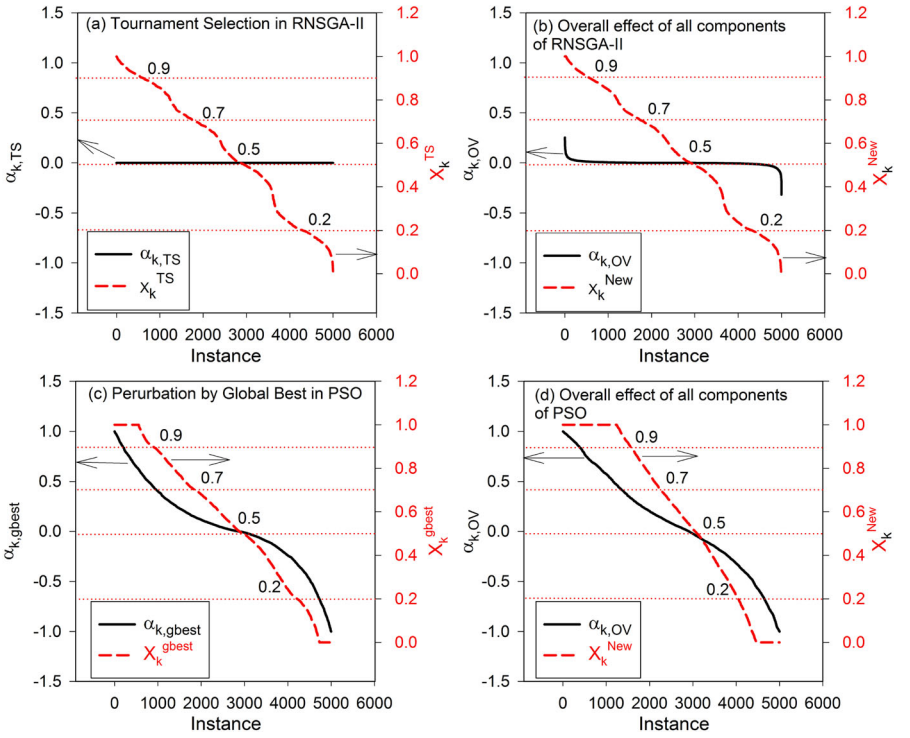

Fig. 4 **a** Effect of tournament selection (TS), **b** overall effect of all components of RNSGA-II, **c** effect of perturbation by the global best position (gbest), and **d** overall (OV) effect of all components of PSO on perturbation of the variable in multi-objective optimization involving multiple current best solutions indicated in terms of the extent of perturbation ($\alpha_{k,\text{Step}}$, Step = TS, gbest, OV) and movement of variable values [X_k^{Step}, Step = TS, gbest, New (values after all perturbation components)] toward the current best solution (*Note* The Y-axes for corresponding curves of $\alpha_{k,\text{Step}}$ and X_k^{Step} are indicated by arrows, and four current best solutions are shown by dashed lines)

The above analysis is extended to a multi-objective RNSGA-II and multi-objective PSO by considering four best values of the variable ($\text{CB}_1 = 0.9, \text{CB}_2 = 0.7, \text{CB}_3 = 0.5$, and $\text{CB}_4 = 0.2$) for the given illustrative example while keeping all other data the same. With this modification, the selection operation in RNSGA-II captures the inflation in the X_k^{TS}-curve at different best values as shown in Fig. 4a. The effect of SBX and PLM is the same as described the above. A corresponding overall effect is shown in Fig. 4b which is in accordance with the single-objective case as described the above. Similar to this, the perturbation by global best position in multi-objective PSO captures the inflation in the X_k^{gbest}-curve at different best values as shown in Fig. 4c. However, the amount of bias toward the current best solutions for the multi-objective PSO decreased considerably compared to that observed for the single-objective case (see Fig. 3a). This is primarily because, the global best position used in the global best perturbation is selected randomly from CB_1 to CB_4 for each instance which adds to the diversity. This bias is further diluted in the overall effect shown in Fig. 4d due to the additional perturbation by the personal best position and the inertia component.

Figures 2d and 3d which represent the qualitative nature of perturbation in RNSGA-II and PSO for single-objective optimization (and also Fig. 4b, d which represents the qualitative nature of perturbation in RNSGA-II and PSO for multi-objective optimization) illustrate

that two important aspects are required for the success of RNSGA-II and PSO. The first aspect is a biased perturbation, or a *guided move* toward the best solutions (i.e., exploitation), whereas the second aspect is a random perturbation, or a *random move* (i.e., exploration). Random moves are able to maintain a sufficient diversity in the swarm and inhibit the tendency to get trapped in local optima. In RNSGA-II and PSO, several operations are used for both exploitation and exploration (e.g., perturbation by global best position, personal best position, and the inertia component are used in PSO for exploitation). Thus, there is a possibility to reduce the number of operations and to simplify the algorithm. On perturbing the variable, the use of several operations has provided additional exploration or exploitation capabilities. Besides this, more flexibility is provided in tuning the balance between exploration and exploitation through respective adjustable parameters (e.g., k_1, k_2, and w in PSO and NSPSO for perturbation by personal best position, perturbation by global best position, and the inertia component, respectively). However, the applications of several operations also increase the number of adjustable parameters, which makes their tuning difficult. Ideally, just two perturbation operations (components), one performing the exploration and the other performing the exploitation, are sufficient for optimization. For instance, a guided move similar to the perturbation by the global best position can be used in place of selection and crossover operations in RNSGA-II as illustrated by the similarity of the X-curves in Figs. 2a, b and 3a. Similarly, the perturbation by personal best position and the inertia component in PSO, which perform the directional exploration and maintain the diversity in the swarm, can be replaced by a more refined random move that can set the velocities of the particles in random directions. Moreover, the significance of perturbation by personal best position and inertia component is reduced in a multi-objective PSO as multiple global best solutions are naturally present, adding a significant diversity in the swarm (see Fig, 4d). An algorithm is, thus, constituted using two perturbation components, guided moves and random moves as described the above and other non-perturbative components such as initiation, non-dominated ranking, and elitism from GA studies (Deb et al. 2002a) and is discussed in the next section (Sect. 4).

Operations in various basic algorithms (e.g., binary coded GA and single-objective PSO) are developed for some specific purposes. However, over the years, these basic algorithms were adapted for a multi-objective optimization (e.g., RNSGA-II and multi-objective PSO), which is further improved by incorporating new features, e.g., binary coded GA is improved by incorporating the real coding in RNSGA-II. In these adaptations, the operations that were developed for the original algorithms are retained with some changes, primarily to mimic the underlying concept, without analyzing whether some of these operations are actually required in the adapted version. For instance, the mutation operation developed for the original binary coded GA is absolutely necessary for the convergence of solutions to a global optimum, particularly for the case when the global optimum has a different genetic content than the variations present in the entire gene pool. This is because such a new genetic content cannot be obtained purely by a binary crossover, which involves only the swapping of genetic content between the parent solutions. Therefore, the mutation operation not only represents the biomimicking of one of the important aspects of the evolution process but also is absolutely necessary for the convergence of solutions to global optimal solutions. Unlike a binary crossover, the simulated binary crossover operation used in real-coded GA, an advanced version of binary crossover, perturbs the variables randomly in the neighborhood of the parent solutions, where all possible genetic contents can be visited by this operation. Therefore, the use of polynomial mutation operation in real-coded GA, an advanced version of mutation used in binary coded GA, is not necessary. This operation is used primarily to retain the biomimicking of one of the important aspects of the evolution process and for an additional exploration. Similarly, the addition of an inertia component and perturbation

by personal best position in PSO are absolutely necessary for single-objective optimization, particularly for multimodal problems. This is because a single global best solution, present in each iteration, tends to make the algorithm prone to entrapment in a local optimum (Reyes-sierra and Coello-coello 2006). However, these operations are not absolutely necessary in multi-objective optimization in which *multiple* global best solutions are naturally present in each iteration since perturbations using these itself provide the diversity of solutions in objective space.

4 SMPSO and its qualitative analysis

Based on the above analysis, a hybrid algorithm, SMPSO, is developed by combining the suitable components of RNSGA-II and PSO in an adapted form to simplify the overall structure of the algorithm, while maintaining the balance between exploration and exploitation.

4.1 Algorithm development

The SMPSO algorithm proceeds in the following order of steps: initialization, objective function calculation, ranking of solutions based on non-dominated sorting (Deb et al. 2002a), guided move (GM) adopted from the global best perturbation used in PSO, random move (RM) adopted from the polynomial mutation operation used in RNSGA-II, and elitism (Thierens and Goldberg 1994). In this algorithm, the number of perturbation components is reduced to two (GM and RM) from three components (i.e., perturbation by global best position, perturbation by personal best position, and addition of the inertia component) used regularly in PSO variants (Reyes-sierra and Coello-coello 2006), such as NSPSO. Also, the algorithm follows a fully connected topology. The flowchart of SMPSO is shown in Fig. 5.

The algorithm initializes with the generation of a swarm of N_P feasible solutions. Each solution is a vector of decision variables. The quality of a solution depends on the values of these variables. The variables of a solution lie between the respective predefined lower and upper bounds, X_j^{Low} and X_j^{High}, and are generated (see Box P in Fig. 5) randomly as follows:

$$X_{i,j} = X_{i,j}^{\text{Low}} + \text{RN}\left(X_j^{\text{High}} - X_j^{\text{Low}}\right) \tag{1}$$

where RN is a random number uniformly sampled from (0, 1) and $X_{i,j}$ is a jth variable of the ith solution. These variable values are then used in the objective function to calculate the quality of the solution. The real-world problems often involve constraints. In the developed algorithm, constraints are handled using a penalty function (Deb 2001), where the penalties for constraint violation are added to the objective function. Such addition of penalties in objective function for constraint violations drives the search toward the feasible solutions. This process is repeated for all N_P solutions. Each solution is then ranked using the non-dominated sorting procedure (see Box P' in Fig. 5) (Deb et al. 2002a). In this procedure, a solution from the population is transferred to a separate set designated to store the solutions. Each of the remaining $N_P - 1$ solutions is then checked for non-dominance. In this checking, if a solution from the original set is found to be non-dominated with respect to all solutions present in the new set, the solution is transferred to the new set. Otherwise, the solution is retained in the original set. Also, all solutions from the new set that are dominated by this solution are transferred back to the original set. After checking all N_P solutions, only non-dominated solutions exist in the new set. These are assigned rank 1. The entire procedure is then repeated for the solutions still present in the original set until all the solutions are assigned the ranks 2, 3, and so on.

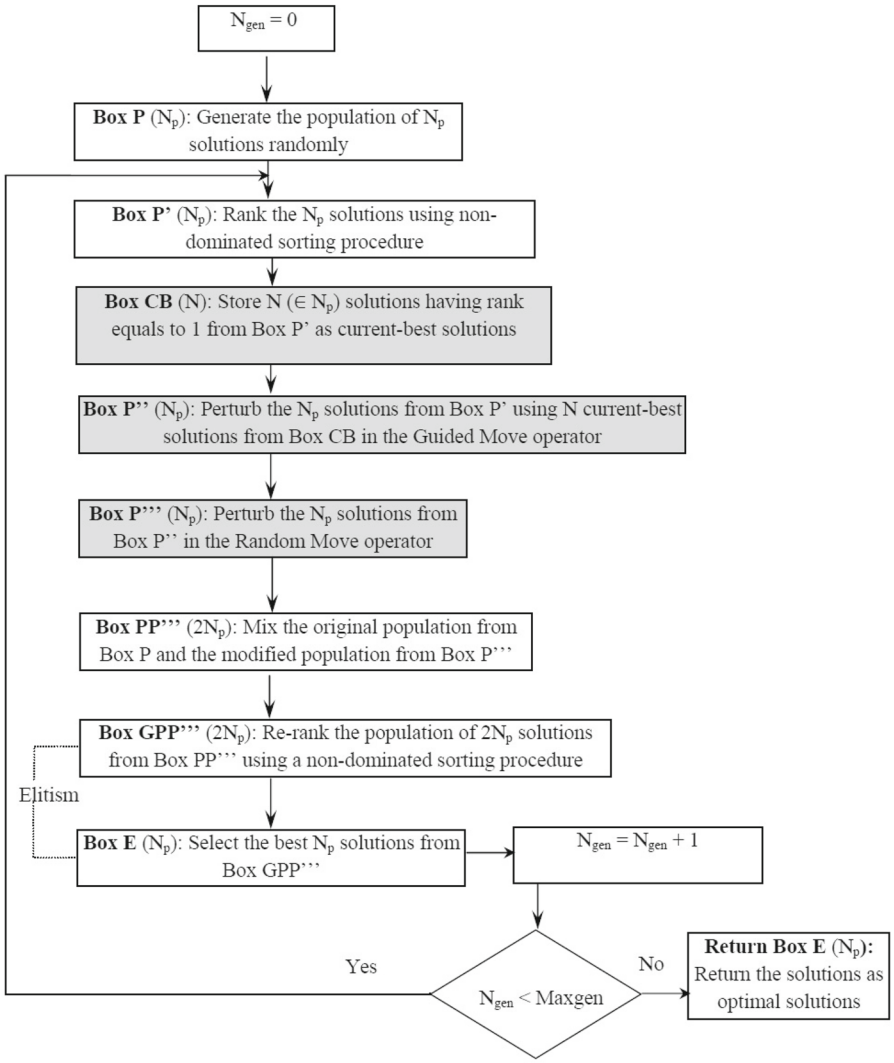

Fig. 5 A flowchart of SMPSO. Gray-colored boxes highlight the modifications added in SMPSO, whereas other boxes represent the commonly used operations in non-dominated sorting-based multi-objective algorithms

Thereafter, the solutions having rank 1 are designated as current best solutions (see Box CB in Fig. 5) for the corresponding generation. The GM operator moves all solutions in the swarm (including the current best solutions) in the direction of randomly selected current best solutions, one after another, with a probability of P_{GM}. More precisely, this operation is performed variable-wise where the underlying variable moves to a new value by adding its weighted (i.e., multiplied by a factor $\beta \in [0, 1]$) difference from the corresponding variable of a randomly selected current best solution from the set of N current best solutions. The jth variable of the ith solution is moved to a new value as follows:

$$X_{i,j}^{\text{New}} = X_{i,j}^{\text{Old}} + \beta \left(X_{n,j}^{\text{CB}} - X_{i,j}^{\text{Old}} \right); \quad i \in [1, N_{\text{P}}]; \quad j \in [1, N_{\text{V}}]; \quad n \in [1, N] \tag{2}$$

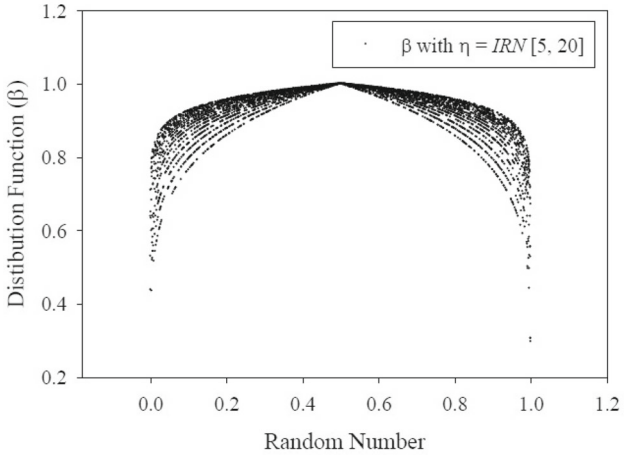

Fig. 6 Variable perturbation function β (with $\eta = \text{IRN} \in [5, 20]$) versus random number (RN)

where $X_{n,j}^{\text{CB}}$ is the jth variable of the randomly selected nth current best solution, β is a scalar value [$\in (0, 1)$], N_{V} is the total number of variables, and N is the total number of current best solutions. The specific range of β restricts the perturbation of the variable between the range $[X_{i,j}^{\text{Old}}, X_{n,j}^{\text{CB}}]$. Perturbation beyond this range is obtained in a random move described later in this section. The distribution function for generating β in SMPSO is similar to that of the PLM operation (Deb and Agarwal 1999) and is given as follows:

$$\beta = \begin{cases} (2\text{RN})^{\frac{1}{\eta+1}} & \text{for RN}(\sim u[0,1]) < 0.5 \\ [2(1-\text{RN})]^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \tag{3}$$

Here, the value of η is selected as a random integer number (IRN) between the lower limit, L_1, and the higher limit, L_2 (i.e., $\eta \sim \text{IRN}[L_1, L_2]$, $L_2 \geq L_1$). It is to be noted that if L_1 and L_2 equal to 1, the β values are uniformly distributed between 0 and 1, whereas for higher values of L_1 and L_2, the β values are biased toward 1. The distribution of β for $\eta \sim \text{IRN}[5, 20]$ is shown in Fig. 6 in which the values of β are plotted against the corresponding random numbers for 5000 random instances. It is clear that β varies close to 1 which ensures that the improved value of $X_{i,j}^{\text{New}}$ becomes closer to that of the current best solution, $X_{n,j}^{\text{CB}}$, as per Eq. (2). The presence of multiple trajectories for β gives a wider range of perturbation in Eq. (2). The expression for β [Eq. (3)] can take two forms for RN in the range 0 to 0.5 and 0.5 to 1. The expression gives the symmetric distribution of β over the entire range of 0 to 1 of RN with a mean laying exactly at 0.5 as shown in Fig. 6. Further, this range can be changed to have an asymmetric distribution of β. However, a symmetric distribution of β is more favorable when the algorithm is applied with the common parameter setting to a variety of problems. Such a symmetric distribution of index is also used in simulated binary crossover (Deb and Agrawal 1994), polynomial mutation (Deb and Agarwal 1999), and simulated binary jumping (Ramteke et al. 2015) gene operations used in RNSGA-II and RNSGA-II-SBJG (see Table S1, supplementary material).

The term $\beta(X_{n,j}^{\text{CB}} - X_{i,j}^{\text{Old}})$ in Eq. (2) of the GM operation resembles the global best position perturbation of the velocity equation. This velocity contribution is added to the old position, $X_{i,j}^{\text{Old}}$, for calculating the new position, $X_{i,j}^{\text{New}}$, similar to PSO. However, unlike PSO, the developed SMPSO does not involve the velocity contribution from perturbation by personal

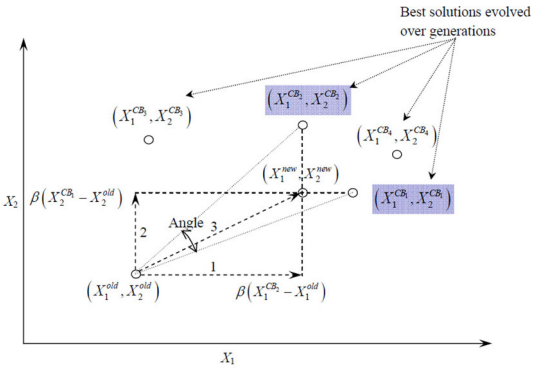

Fig. 7 A guided move operation illustrated for a two-variable problem with variables X_1 and X_2

best position and the inertia component (see Table S1, supplementary material). Thus, the use of GM operation simplifies the perturbation in SMPSO as compared to PSO.

The effect of the GM operation on a trial solution (selected arbitrarily for illustration) comprising two variables: X_1^{old} and X_2^{old}, is shown in Fig. 7. Consider four current best solutions having current best position 1 as $(X_1^{\text{CB}_1}, X_2^{\text{CB}_1})$, current best position 2 as $(X_1^{\text{CB}_2}, X_2^{\text{CB}_2})$, current best position 3 as $(X_1^{\text{CB}_3}, X_2^{\text{CB}_3})$, and current best position 4 as $(X_1^{\text{CB}_4}, X_2^{\text{CB}_4})$ present in the swarm. Suppose, out of these four current best solutions, current best position 2 is randomly selected to perturb X_1^{old} and current best position 1 is randomly selected to perturb X_2^{old}. Thus, X_1^{old} and X_2^{old} will be perturbed in the directions of the corresponding variables $X_1^{\text{CB}_2}$ and $X_2^{\text{CB}_1}$ using β as shown by arrows 1 and 2. The resultant solution, comprising variables X_1^{new} and X_2^{new}, is shown by arrow 3 which is closer to both current best position 1 and current best position 2. It is to be noted that the deviation of the resultant solution from the corresponding current best solutions (i.e., current best position 1 and current best position 2) used in the perturbation increases with the increase in deviation between the used current best solutions (i.e., the angle between current best position 1 and current best position 2) and with the decrease in value of β. Thus, the GM operation with multiple current best solutions leads to both exploration and exploitation.

Because the GM operator is probabilistically applied to variables of the solutions in the swarm, not all of the variables undergo perturbation. Those not perturbed by GM are perturbed using the RM operator to maintain diversity in the swarm and to obtain the improved set of current best solutions for the next generation (see Box P''' in Fig. 5). This operator perturbs a variable randomly between prescribed lower and upper limits as follows:

$$X_{i,j}^{\text{New}} = X_{i,j}^{\text{Old}} + \phi \left(X_j^{\text{High}} - X_j^{\text{Low}} \right) \tag{4}$$

where ϕ is a random number between -1 and 1. This operation is a generalized version of the PLM operation in RNSGA-II. In PLM, $\alpha_{k,\text{PLM}}$ is usually distributed around zero as shown in Fig. 2c. In the proposed algorithm, the distribution function, $\alpha_{k,\text{PLM}}$, is replaced by a more generalized distribution function, ϕ, for which the values are distributed uniformly between -1 to 1. Using this new distribution function, the variables may go beyond the given bounds after application of the RM operation. For such cases, values are restricted to the nearest bounds. This operator may lead to both superior (in terms of quality) and inferior solutions as compared to solutions before the perturbation. Superior solutions constitute the population of current best solutions in the next generation and lead to the better direction of search in the GM operation, whereas inferior solutions are eliminated in the subsequent elitism operation of the current generation. Further, fluctuations due to this operation are restricted to a few

solutions because the probability of this operation is low [(1- P_{GM}) $\approx$ 0.1]. This operation is essential to maintain diversity, particularly when the GM operation is greedy in nature (i.e., perturbation is directed toward the current best solution). Thus, the RM operation is complementary to the GM operation as it tends to perturb the solution in a random direction.

The elitism operator (Deb et al. 2002a; Thierens and Goldberg 1994) mixes the modified population and the original population of solutions to form a population comprising of $2N_P$ solutions (see Box PP''' in Fig. 5). These $2N_P$ solutions are re-ranked (see Box GPP''' in Fig. 5) using the non-dominated ranking procedure (Deb et al. 2002a) and the best N_P solutions from this mixed population are selected as elites (see Box E in Fig. 5). These N_P elites of the current iteration (N_{gen}) form the starting population for the next iteration ($N_{gen} + 1$). This process is repeated until the algorithm converges to the optimal solutions or till the user-defined maximum number of iterations (Max_{gen}) is executed.

The developed algorithm has three adjustable parameters: P_{GM}, L_1, and L_2. In literature, packages such as ParamILS (Hutter et al. 2009) and F-Race (Birattari et al. 2002; Birattari 2009) are used for a rigorous tuning of the parameters of the algorithm for an individual problem such as TSP and satisfiability. However, the use of these rigorous methods for tuning the algorithm parameters to obtain a common set of best values for 30 benchmark problems, as that in the present study, is not only difficult but also computationally prohibitive (Tang et al. 2014). This is primarily because, all 30 benchmark problems have to be solved simultaneously for each trial set of parameters, and the procedure has to be continued over several iterations with varying parameter values in order to get the best set. These benchmark problems are described in Sect. 5.1. In the present study, a partial tuning methodology (Neumuller et al. 2012; Ramesh et al. 2012) is used. In this, some of the parameters are fixed based on the nature of the associated operations empirically or based on the best values reported in literature, whereas the remaining ones are tuned rigorously over several benchmark problems. This not only simplifies the tuning to a manageable extent but also provides reasonably good values for parameters. In the present study, the values of parameters $L_1(= 5)$ and $L_2(= 20)$ are kept the same as that used in the recently developed simulated binary jumping gene operator (Ramteke et al. 2015) to maintain the range of perturbation at the local level and P_{GM} is tuned by varying it from 0.5 to 1.0 in intervals of 0.1. For this, the hyper-volume ratio (HR) metric for all 30 benchmark problems is calculated for $P_{GM} = 0.5, 0.6, 0.7, 0.8, 0.9$, and 1.0 for 50 iterations and 50 different runs (with different random seeds). The reference points used for calculating HR values for all 30 problems are given in Table S2 of supplementary material. A comparison of HR values shows that $P_{GM} = 0.9$ is the most suitable choice because, with this value, the performance of the algorithm is superior for 21 problems out of the total of 30. Corresponding statistical analysis of HR values using the Friedman test is given in Table S3 of supplementary material. The suitability of the selected values of P_{GM}, L_1, and L_2 is further established with tuning on the illustrative example in Sect. 4.2 (see Fig. S1 of supplementary material). These tuned parameter values are used throughout the paper.

The simplification of PSO operators in SMPSO makes the perturbation [shown in Eq. (2)] look similar to a mutant solution (vector) generation in DE (Mezura-Montes et al. 2008). However, except for this similarity, SMPSO differs completely from DE. These differences are highlighted in the following points: (1) In DE, the mutant solution is generated first. In the DE/rand/1/bin strategy, a mutant solution corresponding to each target solution is generated by adding a scaled difference in variables of *two randomly* selected solutions (r_1 and r_2) to that of a *third randomly* selected solution (r_3). In the DE/best/1/bin strategy, a scaled difference in variables of *two randomly* selected solutions (r_1 and r_2) is added to the best solution. In the DE/current-to-best/1/bin strategy, a scaled difference in variables of the best solution with that of the target solution is added to the target solution in addition

to the usual difference in variables of *two randomly* selected solutions (r_1 and r_2). This mutant solution is then used for perturbation of a corresponding target solution in subsequent steps of crossover and selection. In contrast to this, a *biased* scaled (i.e., due to use of β) difference between variables of a *randomly selected current best solution* and that of a *target solution* is added into the corresponding variable of a *target solution* for perturbation in SMPSO without any further steps, such as crossover and selection thereafter. (2) In DE, the crossover is performed after the mutant solution generation, where some of the variables of target solutions are replaced by the corresponding variables of the mutant solution to generate the trial solution. No such step is performed in SMPSO. (3) In DE, the trial solution undergoes the selection operation, where the quality of trial solution is compared with that of the target solution. If only the former is higher, the perturbed solution is copied into the modified population; otherwise, the target solution is copied into the modified population as it is. No such step is performed in SMPSO as all perturbed solutions constitute the modified population irrespective of their qualities. Therefore, DE is greedier and more complex than the SMPSO. Clearly, the operating procedure of SMPSO differs significantly from DE.

4.2 Qualitative analysis of guided and random moves

Similar to the qualitative analysis of components of PSO and RNSGA-II presented in Sect. 3, the qualitative analysis of the developed guided and random moves, on variable perturbation, is performed for both single- and multi-objective optimization. The same illustrative example described in Sect. 3 is used for the analysis. The distribution function, $\alpha_{k,\text{Step}}$, and the new variable values after perturbation, X_k^{Step}, are plotted against the number of instances for the perturbation operations (steps) of SMPSO: guided move and random move. The effect of the guided move on variable perturbation is shown in Fig. 8a. The $\alpha_{k,\text{GM}}$ values shown in Fig. 8a are consistently higher than zero, except for a few ($\sim$ 10%) instances for which the variable remains unperturbed (i.e., $\alpha_{k,\text{GM}} = 0$ in Fig. 8a as indicated by a small plateau) due to a probability of 0.9 used for the guided move. This indicates that the guided move leads to a significant extent of perturbation. Further, the X_k^{GM} values in Fig. 8a show that a significant portion of the X_k^{GM}-curve tends to match the best value of 0.7. This indicates that the perturbations in the guided move are considerably biased toward the current best value of the variable. Unlike this, the random move perturbs the variable randomly. However, only the 10% instances missed out in the guided move undergo the random move perturbation. This is indicated by nonzero $\alpha_{k,\text{RM}}$ values for $\sim$ 10 % instances in Fig. 8b. In this operation, no additional bias toward the current best solution is added and therefore the X_k^{RM}-curve remains the same as that of Fig. 8a. The overall effect of both guided and random moves in terms of extent of perturbation, $\alpha_{k,\text{OV}}$, and new variable values, X_k^{New}, is shown in Fig. 8c. The results indicate that the combination of just two perturbation operations, i.e., the guided and random moves, leads to a biased perturbation toward the current best value similar to that obtained for RNSGA-II and PSO (see Figs. 2d, 3d) which involve three operations each for perturbation.

In contrast to RNSGA-II and PSO (see Figs. 2d, 3d), the bias toward the current best solution shown in Fig. 8c is excessive (shown by a large plateau at $X = 0.7$) which is undesired because such greedy approach leads to the entrapment in local optimum (Reyes-sierra and Coello-coello 2006). This occurs primarily due to the presence of a single current best solution (CB = 0.7) in the illustrative example. However, for a multi-objective optimization, several current best solutions are naturally present and therefore the effect tones down significantly. This is demonstrated using the same illustrative example with four current best solutions

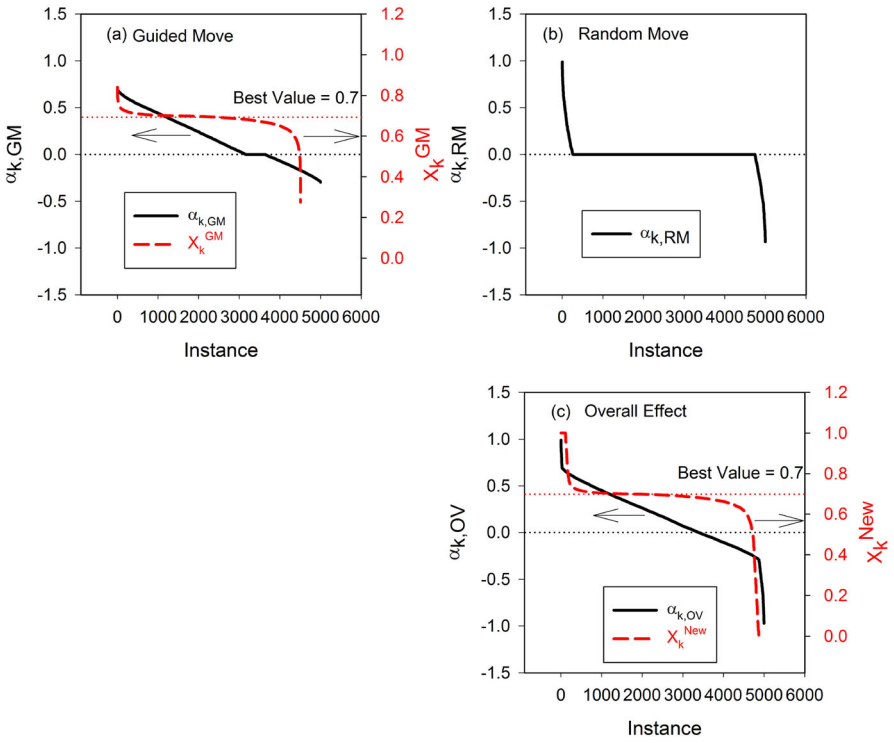

Fig. 8 Effect of **a** guided move (GM) with probability equals to 0.9, **b** random move (RM) with probability equals to 0.1, and **c** their overall (OV) effect on perturbation of the variable in terms of the extent of perturbation ($\alpha_{k,\text{Step}}$; Step = GM, RM, and OV) and movement of variable values [X_k^{Step}; Step = GM and New (values after all perturbation components)] toward the current best solution for single-objective optimization (*Note* The Y-axes for corresponding curves of $\alpha_{k,\text{Step}}$ and X_k^{Step} are indicated by arrows, and four current best solutions are shown by dashed lines)

with the values of $\text{CB}_1 = 0.9$, $\text{CB}_2 = 0.7$, $\text{CB}_3 = 0.5$, and $\text{CB}_4 = 0.2$. With these current best solutions, the $\alpha_{k,\text{GM}}$ and the X_k^{GM} curves for GM operation are shown in Fig. 9a. The distribution function, $\alpha_{k,\text{GM}}$, indicates the reasonable extent of perturbation (except for a small plateau as the probability of GM is 0.9), whereas the X_k^{GM}-curve shows a bias toward the four current best values as indicated by four inflation points at 0.9, 0.7, 0.5, and 0.2. However, unlike the GM operation with a single current best solution (see Fig. 8a), there is no significant plateau in X_k^{GM}-curve which indicates that the bias toward the current best solutions is toned down significantly. The effect of the RM operation (performed on the 10 % instances missed in the GM operation) is the same as that shown in Fig. 8b. This random perturbation helps in evolving better solutions in the new population which pay higher dividends in the next iterations as these are subsequently used as current best solutions. The overall effect in terms of extent of perturbation, $\alpha_{k,\text{OV}}$, and new variable values, X_k^{New}, of both operations is shown in Fig. 9b. The $\alpha_{k,\text{OV}}$-curve is continuous and resembles partially the overall effect shown in Fig. 4d for multi-objective PSO. Further, the X_k^{New}-curve is biased toward the multiple best solutions. Since the solutions will be perturbed by different best solutions over the iterations, the diversity is maintained even with the greedy approach. Unlike PSO, the overall effect of the new algorithm does not have an excessive perturbation as there is no plateau at $X = 1$ (see Figs. 3d, 4d). This clearly illustrates that the simple form of just two perturbation moves

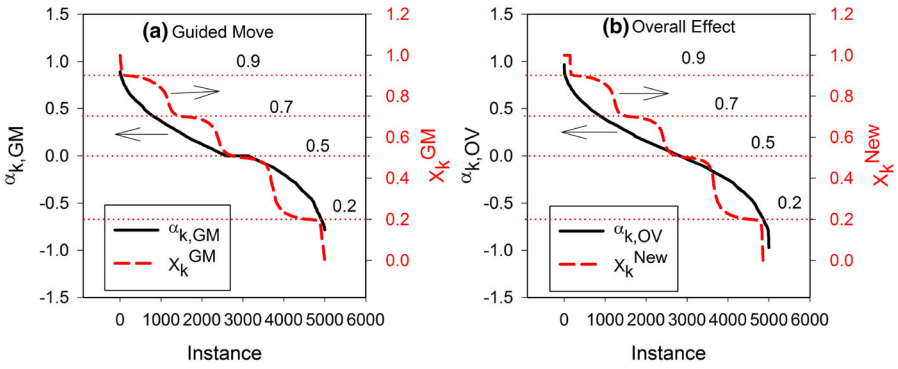

Fig. 9 Effect of **a** guided move (GM) with probability equals to 0.9 and **b** overall effect of guided move and random move (see Fig 8b) on perturbation of the variable in terms of the extent of perturbation ($\alpha_{k,\text{Step}}$; Step = GM and OV) and movement of variable values [X_k^{Step}; Step = GM and New (values after all perturbation components)] toward the multiple current best solutions ($\text{CB}_1 = 0.9$, $\text{CB}_2 = 0.7$, $\text{CB}_3 = 0.5$, and $\text{CB}_4 = 0.2$) in multi-objective optimization (*Note* The *Y*-axes for corresponding curves of $\alpha_{k,\text{Step}}$ and X_k^{Step} are indicated by arrows, and four current best solutions are shown by dashed lines)

(GM and RM moves) used in the SMPSO can yield the desired level of exploration and exploitation. Further, the extent of exploration and exploitation can be modulated by varying the adjustable parameters. This is shown by varying P_{GM} (= 0.5, 0.65, and 0.8) while keeping the η(= IRN[L_1, L_2]) fixed at IRN [5, 20] and varying η (= IRN [1, 1], IRN [1, 5], and IRN [5, 15]) while keeping the P_{GM} fixed at 0.9 in Fig. S1 (supplementary material).

5 Benchmark and industrial problems

5.1 Benchmark MOO problems

The ability of newly developed algorithms to deal with challenges associated with optimization problems is evaluated by applying these on benchmark problems. The features such as multimodality, many-to-one mapping, and biased formulation are often encountered in optimization problems. The benchmark problems are purposefully designed (Huband et al. 2006) to include these features. The feature of many-to-one mapping involves different sets of variable values giving the same value of objective function (Huband et al. 2006). The feature of biased formulation involves an uneven distribution of solutions in the Pareto optimal front for an even distribution of variables in the variable space (Huband et al. 2006). Therefore, solving the benchmark problems helps researchers in understanding the behavior of an algorithm in the presence of the above-mentioned features. Also, standard results for benchmark problems are available in the literature, which facilitates comparison of the algorithms. However, a single benchmark problem may not evaluate an algorithm completely and often a group or suite of benchmark problems is used for this task. In the present work, 30 MOO benchmark problems (Huband et al. 2006) from three such benchmark suites, namely ZDT (ZDT1–ZDT6 except ZDT5), DTLZ (DTLZ1–DTLZ7), and WFG (WFG1–WFG9), are solved using SMPSO under a strict computational budget similar to that of computationally intensive problems to evaluate its suitability for the latter. Among these benchmark problems, all the problems of the ZDT suite are scalable only in terms of the number of variables, whereas the problems of the DTLZ and WFG suites are scalable in terms of both

the number of objectives and the number of variables. To test the potential of the algorithm rigorously, the DTLZ problems with three objectives and the WFG problems with both two (2D) and three (3D) objectives are used in the present study. All problems of the ZDT suite are scaled to 10 variables, DTLZ1, DTLZ2–DTLZ6, and DTLZ7 are scaled to 7, 12, and 22 variables, respectively, and all the problems of the WFG suite are scaled to 24 variables. The mathematical formulations of all 30 benchmark problems can be obtained from Huband et al. (2006).

5.2 Industrial MOO problem

Several MOO problems of industrial importance belonging to the areas of polymerization, petrochemicals, catalytic reactors, pharmaceuticals, process design, and control are solved using different metaheuristic algorithms and are reviewed extensively in the literature (Bhaskar et al. 2000; Rangaiah 2009; Rangaiah and Petriciolet 2013; Valadi and Siarry 2014). In the present study, the performance of SMPSO for real-life optimization problems is evaluated by solving a MOO problem related to a complex industrial system of residue fluid catalytic cracking unit (RFCCU) (Sadeghbeii 2000; Yang 2003). The mathematical model (Varshney et al. 2015) of this system consists of several differential equations. These equations must be solved for each function calculation which requires inordinately large computational effort. Since the total number of function calculations is equal to the swarm size multiplied by the maximum number of user-defined iterations, these equations have to be solved repeatedly. Thus, to maintain the economy of computational cost, optimization is often carried out for a restricted number of iterations. In such cases, algorithms with a faster convergence speed are required. The present problem provides an ideal test to evaluate the convergence speed of the SMPSO. The description of the problem formulation is given in supplementary material (section S1).

6 Performance metrics and statistical test

The quality of the results obtained after MOO is evaluated in terms of two factors: the proximity of solutions to the global Pareto optimal front (PF_G) available in the literature (Durillo and Nebro 2011) and the distribution of solutions along the entire Pareto optimal region. To estimate quantitatively the quality of the results on the basis of these two factors, performance metrics are used. Several of these are reported in the literature (Coello-coello et al. 2007) and are frequently used in groups to evaluate the overall performance of an algorithm. In this work, three popular metrics, generational distance (GD), hyper-volume ratio (HR), and spacing (S) (Coello-coello et al. 2007), are used to evaluate the performance of SMPSO. For a better performance, the values of GD, HR, and S must be close to 0, 1, and 0, respectively. Further, the results of these metrics for all 30 problems are compared with those of other algorithms using the Wilcoxon signed rank test (Garcia et al. 2009).

7 Results and discussion

The newly developed hybrid algorithm, SMPSO, is first applied to the 30 MOO problems from the ZDT, DTLZ, and WFG benchmark suites. For each problem, the average values of the metrics GD, HR, and S and their standard deviations over 50 different runs are calculated.

Table 1 The values of parameters used in RNSGA-II, RNSGA-II-SBJG, NSPSO, and SMPSO

RNSGA-II		RNSGA-II-SBJG		NSPSO		SMPSO	
Param.	Value	Param.	Value	Param.	Value	Param.	Value
N_P	100	N_P	100	N_P	100	N_P	100
P_{cross}	0.9	P_{cross}	0.9	w	0.4–0.9	P_{GM}	0.9
P_{mut}	$1.0/N_V$	P_{mut}	$0.5/N_V$	k_1	2	η	IRN[5, 20]
η_{cross}	20	η_{cross}	20	k_2	2.2		
η_{mut}	20	η_{mut}	20				
		P_{JG}[a]	$0.5/N_V$				
		η_{JG}[b]	IRN[5, 20]				

[a] P_{JG} is a probability of simulated binary jumping gene operation
[b] η_{JG} is a index for simulated binary jumping gene operation

The global Pareto optimal fronts, PF_Gs, required for metrics calculation are adopted from the literature (Durillo and Nebro 2011). Additionally, for the calculation of HR values, a reference point for each problem is required. These reference points are identified using the worst values of each objective, associated with a given problem, obtained by all algorithms under comparison (i.e., SMPSO, RNSGA-II, RNSGA-II-SBJG, and NSPSO) over 500 iterations and 50 different runs. The reference points for all 30 problems are given in Table S2 (supplementary material).

Also, the parameters used in all algorithms under comparison are given in Table 1. Among these, crossover probabilities (P_{cross}) in RNSGA-II and RNSGA-II-SBJG and the range of inertia weight (w) in NSPSO are tuned as these are analogous to P_{GM} used in SMPSO by keeping other corresponding parameters constant to that reported in the literature (Deb 2001; Reyes-sierra and Coello-coello 2006; Coello-coello et al. 2007; Ramteke et al. 2015). For this, the hyper-volume (HR) metric for all 30 benchmark problems is calculated for $P_{cross} = 0.5, 0.6, 0.7, 0.8$, and 0.9 for 50 iterations and 50 different runs with different random seeds using both RNSGA-II and RNSGA-II-SBJG. The comparison of HR values shows that $P_{cross} = 0.9$ is the most suitable value for both algorithms because, with this value, the performance of RNSGA-II and RNSGA-II-SBJG is superior for more problems as compared to that for other values of P_{cross}. For $P_{cross} = 0.9$, the HR values of RNSGA-II and RNSGA-II-SBJG are best for 9 and 11 problems, respectively, whereas the second best choice is $P_{cross} = 0.8$ for which the HR values of RNSGA-II and RNSGA-II-SBJG are best for 8 and 9 problems, respectively. A similar analysis for NSPSO with ranges of inertia weight (w) of 0.9–0.5, 0.9–0.4, 0.9–0.3, 0.9–0.2, 0.9–0.1, and 0.9–0.0 shows that the range 0.9–0.4 is the best as it provides the best HR values for 11 problems, a maximum number compared to that obtained for other ranges. The second best choice of the range is 0.9–0.5 for which the HR values are best for 9 problems. The comparisons of HR values are given in Tables S4–S6 of supplementary material, where the best HR values are given in the boldface font. These tuned parameter values are found to be in accordance with those reported in the literature (Deb 2001; Reyes-sierra and Coello-coello 2006; Coello-coello et al. 2007; Ramteke et al. 2015).

The comparisons of metrics obtained using SMPSO, RNSGA-II, RNSGA-II-SBJG, and NSPSO for 50, 100, 200, and 500 iterations are shown in Figs. 10, 11, and 12 and Figs. S2–S7 of supplementary material. Also, these results are given in a tabulated form (Tables S7–S13) in supplementary material. In Fig. 10, the bar charts of HR values obtained in 50 iterations are shown. In these, the bars with values closer to 1 indicate that the corresponding algorithm performs better in terms of HR value. In other words, for each problem, the algorithm with

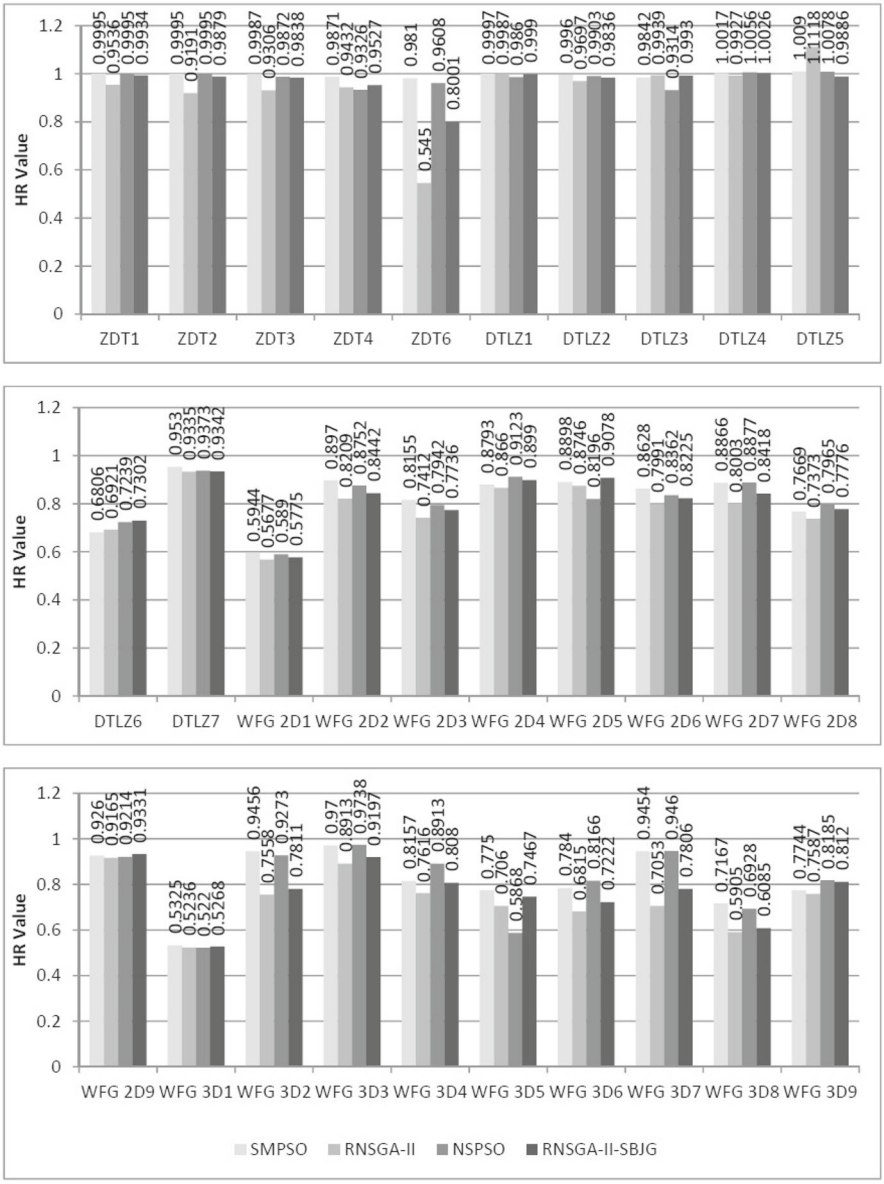

Fig. 10 Bar charts of HR values obtained in 50 generations (or iterations). The HR values are also shown above the bars. Average standard deviations in HR over 30 problems for SMPSO, RNSGA-II, NSPSO, and RNSGA-II-SBJG are 0.0156, 0.0408, 0.0226, and 0.0292, respectively. For standard deviations of individual problems, refer to Tables S7–S9 of supplementary material

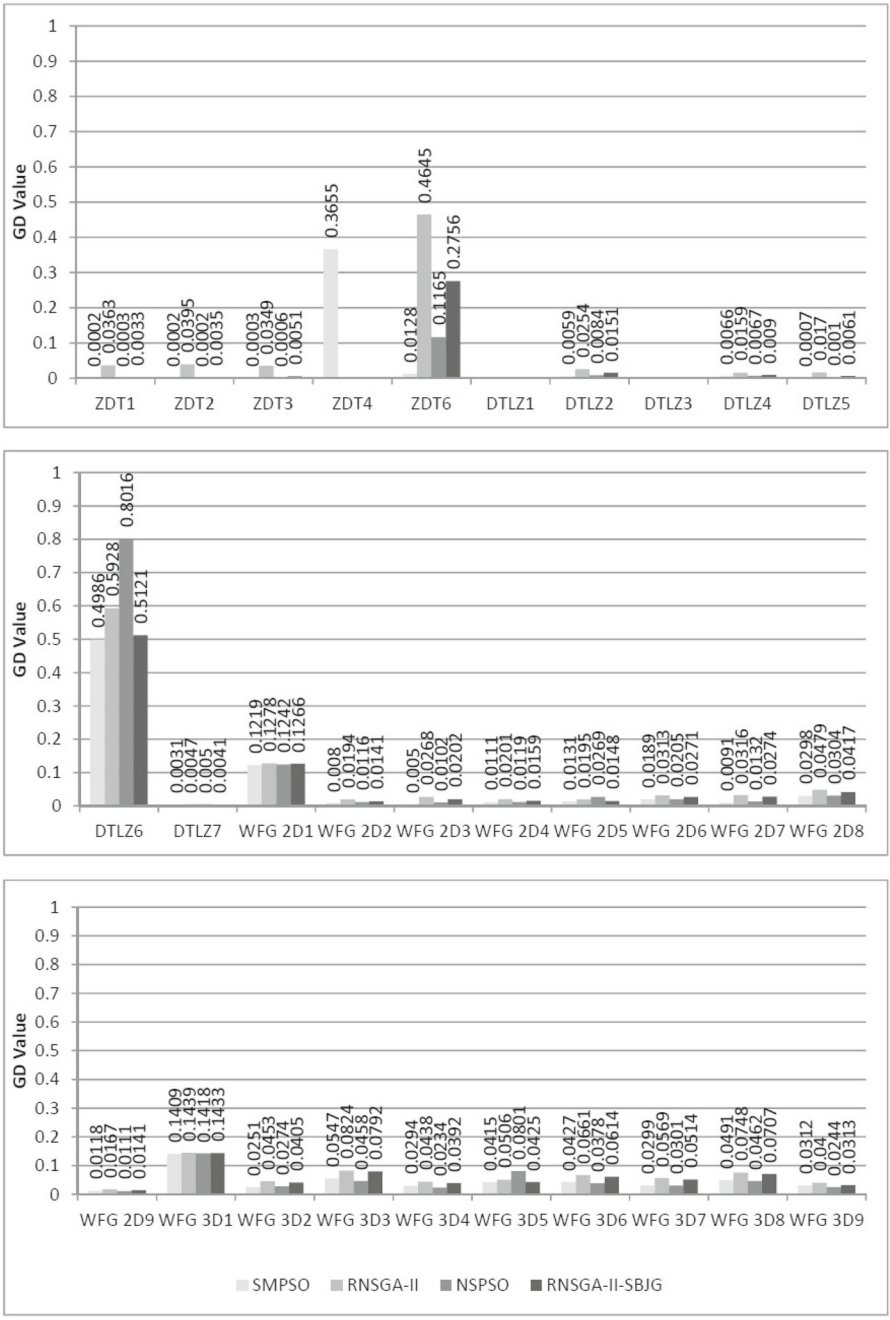

Fig. 11 Bar charts of GD values obtained in 50 generations (or iterations). The GD values are also shown above the bars. Average standard deviations in GD over 27 problems for SMPSO, RNSGA-II, NSPSO, and RNSGA-II-SBJG are 0.0024, 0.0110, 0.0064, and 0.0093, respectively. For standard deviations of individual problems, refer to Tables S10–S12 of supplementary material. The values for ZDT4 using RNSGA-II, NSPSO, and RNSGA-II-SBJG and for DTLZ1 and DTLZ3 using all four algorithms are larger than 1, and therefore, these are not shown in the figure

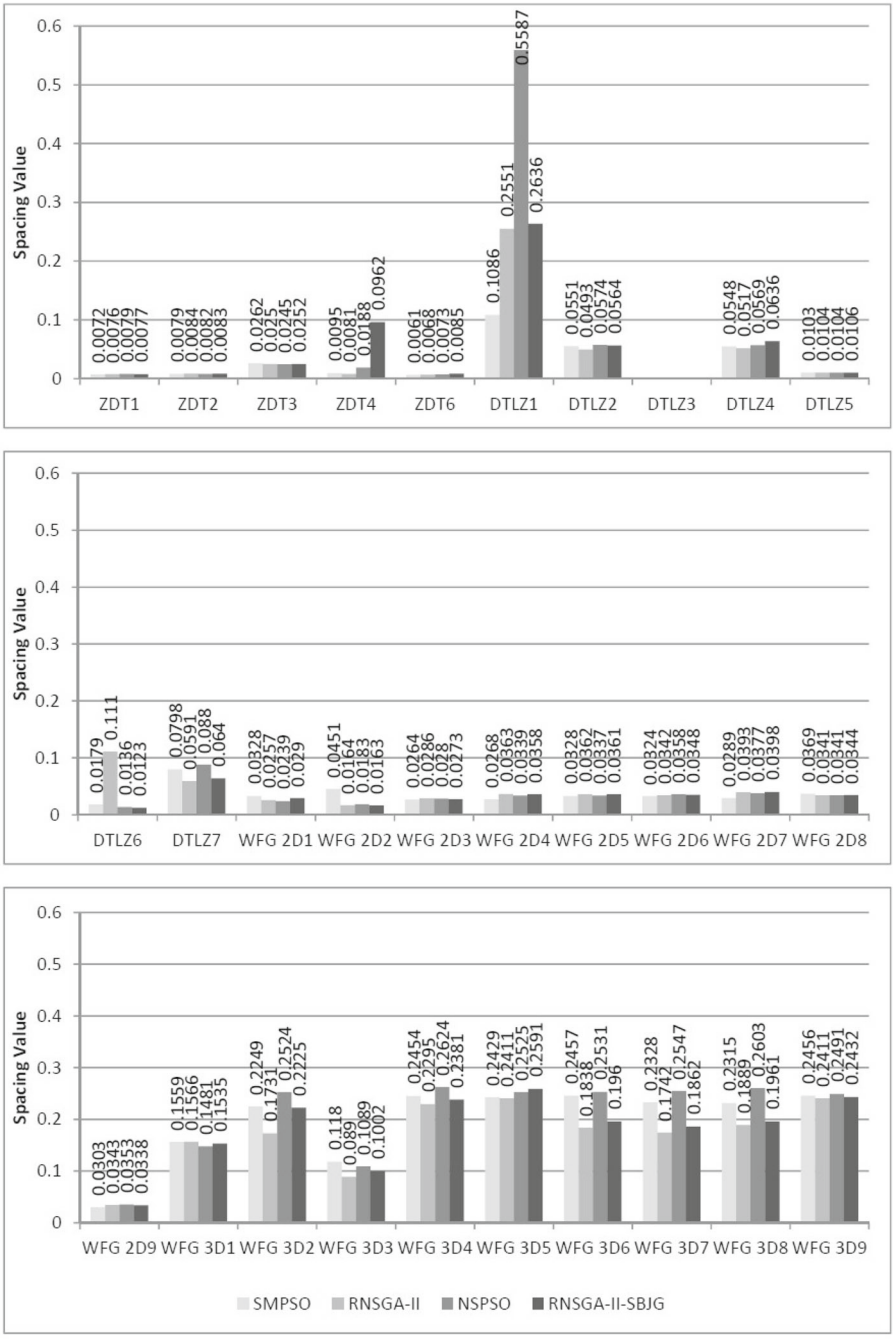

Fig. 12 Bar charts of spacing values obtained in 500 generations (or iterations). The spacing values are also shown above the bars. Average standard deviations in spacing over 29 problems for SMPSO, RNSGA-II, NSPSO, and RNSGA-II-SBJG are 0.0222, 0.0516, 0.0418, and 0.0815, respectively. For standard deviations of individual problems, refer to Table S13 of supplementary material. The values for DTLZ3 using all four algorithms are larger than 1, and therefore, these are not shown in the figure

Table 2 Results of Wilcoxon test for HR metric (number of positive ranks, number of negative ranks, and number of ties give the total number of problems out of 30 for which SMPSO is superior, inferior, and equivalent to the compared algorithm)

Iterations	SMPSO vs.	No. of positive ranks[a]	No. of negative ranks	No. of ties	*p* value
50	RNSGA-II	**27**	3	0	0.000
50	NSPSO	**18**	10	2	0.179
50	RNSGA-II-SBJG	**22**	8	0	0.004
100	RNSGA-II	**24**	6	0	0.000
100	NSPSO	**16**	13	1	0.469
100	RNSGA-II-SBJG	**19**	9	2	0.076
200	RNSGA-II	**25**	5	0	0.002
200	NSPSO	**14**	13	3	0.614
200	RNSGA-II-SBJG	**16**	14	0	0.734
500	RNSGA-II	**17**	13	0	0.304
500	NSPSO	13	14	3	0.943
500	RNSGA-II-SBJG	13	17	0	0.271

[a]Boldface font indicates the superior performance of SMPSO

the highest bar is the best among the four compared algorithms in terms of HR value. Here, it can be seen that the bars of SMPSO are the highest for most of the problems in 50 iterations (see Fig. 10) and also there is a clear visible difference among the heights of bars of different algorithms with NSPSO being the second highest followed by RNSGA-II-SBJG and RNSGA-II. However, this difference narrows down with the increase in the number of iterations [see Figs. S2 and S3 (supplementary material)] and for 500 iterations, the bars become almost equal in height (see Fig. S4 of supplementary material).

The observations made from the figures are further confirmed by the statistical analysis using the Wilcoxon test and are provided in Table 2. In this, the number of positive ranks denotes the number of problems for which SMPSO performs better than the compared algorithm (either RNSGA-II, NSPSO or RNSGA-II-SBJG), the number of negative ranks denotes the number of problems for which SMPSO performs worse than the compared algorithm, and the number of ties denotes the number of problems for which SMPSO performs equal to the compared algorithm. The p values of less than 0.05 also indicate a better performance. These statistical results for the HR metric clearly show that SMPSO outperforms the other three algorithms in 50 and 100 iterations. This trend is a clear indicator of the faster convergence speed of SMPSO which performs exceptionally well in the smaller number of iterations as compared to the rest of the algorithms. Though SMPSO is still better in 200 iterations, the difference in the performance starts narrowing down and becomes almost equal in 500 iterations. This is due to the fact that all the algorithms start converging to the global Pareto optimal front (i.e., PF_O approaches PF_G) for each problem as the number of iterations increases.

The bar charts of the GD values obtained in 50 iterations are shown in Fig. 11. The additional results for 100, 200, and 500 iterations are shown in Figs. S5–S7 of supplementary material for the sake of brevity. Since the GD metric should be close to zero, the bars with smaller heights indicate the better performing algorithm. In Fig. 11, the GD values of the ZDT4, DTLZ1, and DTLZ3 problems for some or all algorithms are not shown because these are very large (much larger than 1). The reason behind such large values is that, for a given problem, the corresponding algorithm is not able to converge. By observing all the bar

Table 3 Results of Wilcoxon test for GD metric (number of positive ranks, number of negative ranks, and number of ties give the total number of problems out of 30 for which SMPSO is superior, inferior, and equivalent to the compared algorithm)

Iterations	SMPSO vs.	No. of positive ranks[a]	No. of negative ranks	No. of ties	*p* value
50	RNSGA-II	**29**	1	0	0.000
50	NSPSO	**23**	6	1	0.013
50	RNSGA-II-SBJG	**29**	1	0	0.000
100	RNSGA-II	**29**	1	0	0.000
100	NSPSO	**21**	6	3	0.007
100	RNSGA-II-SBJG	**25**	4	1	0.000
200	RNSGA-II	**26**	2	2	0.000
200	NSPSO	**19**	7	4	0.011
200	RNSGA-II-SBJG	**25**	5	0	0.000
500	RNSGA-II	**19**	9	2	0.019
500	NSPSO	**15**	11	4	0.110
500	RNSGA-II-SBJG	**17**	10	3	0.428

[a]Boldface font indicates the superior performance of SMPSO

charts of GD values and analyzing the statistical results given in Table 3, it becomes clear that the algorithms follow the same pattern for GD values as that in the case of HR values. This further establishes the fact that SMPSO has a faster convergence speed than the rest of the compared algorithms.

In order to analyze the convergence speed of SMPSO with an increase in the number of iterations, the convergence curves of all algorithms under comparison are plotted in Fig. 13. These curves show the *average HR* values of 30 benchmark problems obtained by all compared algorithms in a single run with respect to the number of iterations (up to 200 iterations). From Fig. 13, it is clear that SMPSO shows faster convergence than the other compared algorithms, from initial iterations itself. Moreover, after 100 iterations, the curve of SMPSO becomes almost flat which shows its ability to find nearly converged solutions in a small number of iterations.

It has to be noted that the metric, S, measures the spread of solutions which naturally assumes the significance only after complete convergence (Ramteke et al. 2015). Thus, the

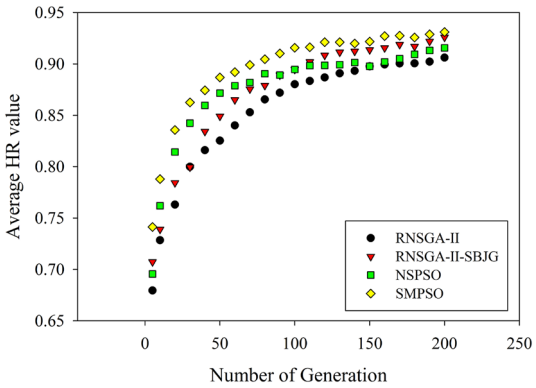

Fig. 13 Convergence curves of compared algorithms. Average standard deviations for RNSGA-II, RNSGA-II-SBJG, NSPSO, and SMPSO are 0.0272, 0.0197, 0.0179, and 0.0125, respectively

Table 4 Results of Wilcoxon test for spacing metric (number of positive ranks, number of negative ranks, and number of ties give the total number of problems out of 30 for which SMPSO is superior, inferior, and equivalent to the compared algorithm)

Iterations	SMPSO vs.	No. of positive ranks[a]	No. of negative ranks	No. of ties	*p* value
500	RNSGA-II	13	17	0	0.109
500	NSPSO	**23**	7	0	0.015
500	RNSGA-II-SBJG	**16**	14	0	0.910

[a]Boldface font indicates the superior performance of SMPSO

values of S are calculated only for 500 iterations in which each algorithm nearly converges to PF_G for most of the problems. Bar chart of S values obtained in 500 iterations is shown in Fig. 12. Similar to the GD, the metric, S, should also be close to zero and thus the smaller bars show the better performing algorithm. Again, the S values of DTLZ3 are not shown because these are outliers (much larger than 1). Both the bar charts and the statistical results (see Table 4) show that SMPSO performs inferior to RNSGA-II, superior to NSPSO and almost equal to RNSGA-II-SBJG in terms of the S values. Since real-life applications often require faster convergence in the smaller number of iterations, the values of the HR and GD metrics become more important than that of the S metric. Thus, the given results show that the SMPSO maintains a very high convergence speed with reasonably distributed Pareto optimal solutions.

A comparison of the CPU times taken by RNSGA-II, NSPSO, RNSGA-II-SBJG, and SMPSO for all 30 problems is given in Table S14 (supplementary material). These CPU times are calculated by executing 50 runs of each problem for 500 iterations on a desktop computer having Intel Xeon E3-1225 v3@3.20 GHz processor, 8 GB RAM, and Windows 7 operating system. Table S14 indicates that the CPU times taken by SMPSO, RNSGA-II, and RNSGA-II-SBJG are nearly equal for all problems, whereas NSPSO takes higher CPU time compared to the other three algorithms for all problems studied.

Performance comparison is further extended to four state-of-the-art multi-objective PSO variants, i.e., MOPSO, TV-MOPSO, CCS-MOPSO, and MOLS-MOPSO. For this, six benchmark problems (ZDT1–ZDT4, DTLZ1, and DTLZ2) are solved using SMPSO for 1000 iterations and 20 separate runs and the mean and standard deviation of GD and S values over these runs are calculated. These values are then compared with those reported by (Xu et al. 2015), under similar conditions, for the above four PSO variants. These values are provided in Tables 5 and 6 which show that SMPSO clearly outperforms these PSO variants in terms of both GD and S. The best values of GD and S are shown in bold letters in Tables 5 and 6, respectively. Also, the comparison is extended (see Table 7) to some of the recent DE variants (MODE-RMO, GDE3, NSGA-II-DE, and MODE) (Chen et al. 2014) for the same condition of 20 separate runs and 250 iterations for ZDT problems and 500 iterations for DTLZ 1 and DTLZ2 as reported in the literature. The results of GD are found to be better than multi-objective DE variants for three benchmark problems out of six.

Motivated from a significantly improved performance observed in benchmark problems, SMPSO is applied to an industrial MOO problem of RFCCU. In this, a RFCCU is optimized by maximizing the gasoline yield (I_1) and minimizing the coke formation on the catalyst (I_2). Optimization protocol is carried out only for 10 iterations with a swarm size of 20 due to the large CPU time required for each function calculation ($\sim$ 70 min using each of the four algorithms which is equivalent to almost 24 hr. for each iteration on the same desktop computer as mentioned above). The Pareto optimal fronts obtained using RNSGA-

Table 5 Comparison of SMPSO with multi-objective PSO variants in terms of GD

Problem	MOPSO	TV-MOPSO[a]	CSS-MOPSO	MOLS-MOPSO	SMPSO
ZDT1	0.00333 ± 0.00000	0.00199 ± 0.00000	0.00242 ± 0.00000	0.00181 ± 0.00000	**0.00008 ± 0.00002**[a]
ZDT2	0.00089 ± 0.00000	0.00078 ± 0.00000	0.00051 ± 0.00000	0.00074 ± 0.00000	**0.00005 ± 0.00000**
ZDT3	0.04186 ± 0.01648	0.00437 ± 0.00000	0.00482 ± 0.00000	0.00314 ± 0.00000	**0.00020 ± 0.00001**
ZDT4	0.86385 ± 0.05294	0.57430 ± 0.00000	0.00820 ± 0.00000	**0.00091 ± 0.00000**	0.00154 ± 0.00136
DTLZ1	0.25428 ± 0.04833	0.12643 ± 0.00687	0.11076 ± 0.00978	0.07339 ± 0.00065	**0.04407 ± 0.04632**
DTLZ2	0.14723 ± 0.00068	0.04677 ± 0.00073	0.05893 ± 0.00031	0.03176 ± 0.00021	**0.00400 ± 0.00200**

[a]Boldface font indicates the superior performance of SMPSO

Table 6 Comparison of SMPSO with multi-objective PSO variants in terms of S

Problem	MOPSO	TV-MOPSO[a]	CSS-MOPSO	MOLS-MOPSO	SMPSO
ZDT1	0.57451 ± 0.00835	0.37692 ± 0.00074	0.93645 ± 0.00000	0.25417 ± 0.00049	**0.00713 ± 0.00052**[a]
ZDT2	0.23922 ± 0.00114	0.18565 ± 0.00058	0.16156 ± 0.00033	0.17783 ± 0.00054	**0.00809 ± 0.00066**
ZDT3	0.79326 ± 0.00742	0.41683 ± 0.00054	0.45285 ± 0.00086	0.48427 ± 0.00031	**0.02588 ± 0.00278**
ZDT4	0.17941 ± 0.09114	0.19551 ± 0.01063	0.16462 ± 0.00156	0.11921 ± 0.00735	**0.00708 ± 0.00103**
DTLZ1	0.45157 ± 0.00641	0.41779 ± 0.00284	0.34826 ± 0.00347	0.21434 ± 0.00475	**0.04745 ± 0.02723**
DTLZ2	0.53587 ± 0.00479	0.35892 ± 0.00671	0.33572 ± 0.00276	0.21536 ± 0.00561	**0.05508 ± 0.00699**

[a]Boldface font indicates the superior performance of SMPSO

Table 7 Comparison of SMPSO with multi-objective DE variants in terms of GD

Problem	MODE-RMO	GDE3	NSGA-II-DE	MODE	SMPSO
ZDT1	0.00385 ± 0.00032	0.00240 ± 0.00056	0.00583 ± 0.00122	0.00532 ± 0.00109	**0.00010 ± 0.00000**[a]
ZDT2	0.00697 ± 0.00072	0.00820 ± 0.00233	0.00775 ± 0.00059	0.00981 ± 0.00128	**0.00000 ± 0.00000**
ZDT3	0.00476 ± 0.00054	0.00276 ± 0.00092	0.00531 ± 0.00092	0.00637 ± 0.00085	**0.00020 ± 0.00000**
ZDT4	0.07290 ± 0.16300	0.03870 ± 0.03770	**0.01160 ± 0.02150**	0.19800 ± 0.21800	0.04130 ± 0.01690
DTLZ1	**0.00025 ± 0.00001**	0.07080 ± 0.07210	0.00480 ± 0.01710	0.00026 ± 0.00001	0.09360 ± 0.04990
DTLZ2	0.00073 ± 0.00002	**0.00073 ± 0.00003**	0.00196 ± 0.00030	0.00074 ± 0.00002	0.00370 ± 0.00170

[a]Boldface font indicates the superior performance of SMPSO

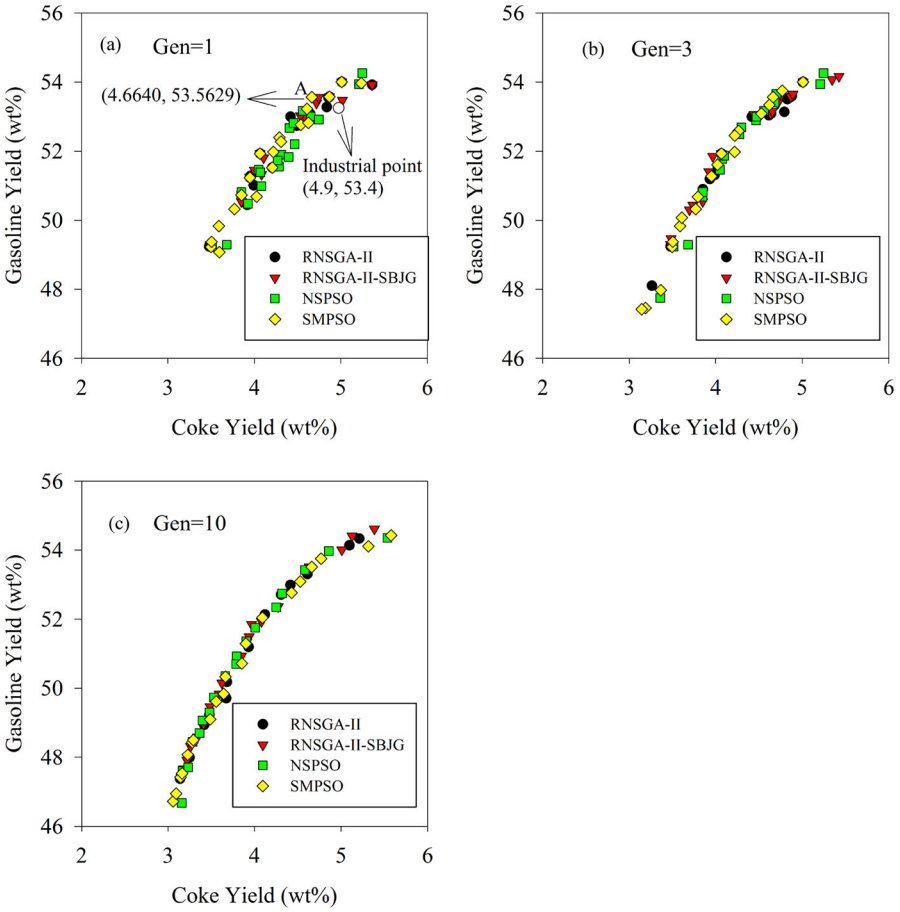

Fig. 14 Pareto optimal fronts of MOO problem of RFCCU for **a** 1, **b** 3, and **c** 10 generations (or iterations)

II, NSPSO, RNSGA-II-SBJG, and SMPSO for 1, 3, and 10 generations (iterations) are shown in Fig. 14a–c, respectively. It is clear from Fig. 14a that SMPSO outperforms the other algorithms in the first generation itself. This is also confirmed from the HR values of 0.8639, 0.8328, 0.8554, and 0.8847 obtained for RNSGA-II, NSPSO, RNSGA-II-SBJG, and SMPSO, respectively, using a pseudo-global Pareto optimal front extracted from the results of 10 iterations. Moreover, the industrial value (Varshney et al. 2015) of gasoline yield is reported as 53.4% corresponding to the coke formation of 4.9% (see Fig. 14a). SMPSO captures a point A (see Fig. 14a) in the first generation which gives a gasoline yield equivalent to the industrial value (53.56%) at a lower coke formation of 4.66%. For the same gasoline yield, NSPSO does not capture a point on the Pareto optimal front; however, RNSGA-II and RNSGA-II-SBJG capture a point with the coke formation of 4.83% and 4.72%, respectively. It shows that the percentage decrease in coke formation with respect to the industrial value using SMPSO, RNSGA-II, and RNSGA-II-SBJG is 4.90%, 1.43%, and 3.67%, respectively.

The results for the third generation (see Fig. 14b) obtained using all algorithms are better than those for the first generation; however, SMPSO shows significantly improved performance than the other algorithms in terms of convergence and also captures some additional points in the lower end of the Pareto optimal front. This is further confirmed by the HR values

of 0.9120, 0.9036, 0.9127, and 0.9438 obtained for RNSGA-II, NSPSO, RNSGA-II-SBJG, and SMPSO, respectively. This illustrates the use of SMPSO for optimization of complex industrial problems which are often associated with a strict computational budget.

8 Conclusions

In the present study, a simplified multi-objective variant of PSO, SMPSO, is developed. The algorithm incorporates two perturbation operators, guided move and random move, and involves three adjustable parameters. In addition, the algorithm does not include the inertia and personal best components in the velocity calculation. This is clearly a deviation from the standard PSO, but it leads to a reduction in the number of user-defined parameters and a simplified algorithmic structure.

The developed algorithm has been applied to 30 MOO benchmark problems. The results obtained are compared with those obtained using RNSGA-II, NSPSO, and RNSGA-II-SBJG. When the number of iterations is less than or equal to 200, SMPSO yields better or equal HR values for 84%, 60%, and 66% problems and better or equal GD values for 96%, 79%, and 89% as compared to RNSGA-II, NSPSO, and RNSGA-II-SBJG, respectively. It also showed significantly improved performance than the multi-objective PSO variants (MOPSO, TV-MOPSO, CCS-MOPSO, and MOLS-MOPSO) and multi-objective DE variants (MODE-RMO, GDE3, NSGA-II-DE, and MODE) for six benchmark problems. Besides the benchmark problems, the utility of SMPSO for complex real-world MOO problems has been investigated. For this purpose, the MOO problem of industrial residue fluid catalytic cracking unit is solved under a strict computational budget. Interestingly, the developed algorithm, SMPSO, showed significant improvement in performance than the other algorithms: RNSGA-II, NSPSO, and RNSGA-II-SBJG.

Acknowledgements The authors gratefully acknowledge the partial financial support from Science and Engineering Research Board, Government of India, New Delhi (through Grant SB/FTP/ETA-125/2013, dated June 5, 2014). Also, the authors acknowledge the detailed suggestions provided by editor, associate editor, reviewers, Prof. Ali Haider and Mr. Gautam Rangari for improving the technical and the linguistic quality of the paper.

References

Beheshti, Z., & Shamsuddin, S. M. H. (2014). CAPSO: Centripetal accelerated particle swarm optimization. *Information Sciences*, *258*, 54–79.

Bhaskar, V., Gupta, S. K., & Ray, A. K. (2000). Applications of multiobjective optimization in chemical engineering. *Reviews in Chemical Engineering*, *16*, 1–54.

Birattari, M. (2009). *Tuning metaheuristics: A machine learning perspective*. Berlin: Springer.

Birattari, M., Stutzle, T., Paquete, L., & Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In W. B. Langdon, E. Cantu-Paz, K. Mathias, et al. (Eds.), *GECCO 2002: Proceedings of the genetic and evolutionary computation conference* (pp. 11–18). San Francisco: Morgan Kaufmann.

Chen, X., Du, W., & Qian, F. (2014). Multi-objective differential evolution with ranking-based mutation operator and its application in chemical process optimization. *Chemometrics and Intelligent Laboratory Systems*, *136*, 85–96.

Cheng, R., & Jin, Y. (2015). A social learning particle swarm optimization algorithm for scalable optimization. *Information Sciences*, *291*, 43–60.

Coello-coello, C. A., Lamont, G. B., & van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems*. New York: Springer.

Coello-coello, C. A., & Lechuga, M. S. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on evolutionary computation part of the 2002 IEEE world congress of computational intelligence* (pp. 1051–1056). Hawaii: IEEE.

Coello-coello, C. A., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*, 256–279.

Das, S., & Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, *15*, 4–31.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Chichester: Wiley.

Deb, K., & Agrawal, R. B. (1994). Simulated binary crossover for continuous search space. *Complex Systems*, *9*, 1–34.

Deb, K., & Agarwal, S. (1999). A niched-penalty approach for constraint handling in genetic algorithms. In *Proceedings of the international conference on artificial neural nets and genetic algorithms* (pp. 235–243). Portoroz: Springer.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002a). A fast and elitist multiobjective genetic algorithm: NSGA-II. *EEE Transactions on Evolutionary Computation*, *6*, 182–197.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2002b). Scalable multi-objective optimization test problems. In D. B. Fogel, M. A. El-Sharkawi, X. Yao, et al. (Eds.), *Proceedings of the 2002 Congress on Evolutionary Computation* (pp. 825–830).

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B (Cybernetics)*, *26*, 29–41.

Durillo, J. J., & Nebro, A. J. (2011). JMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, *42*, 760–771.

Elhossini, A., Areibi, S., & Dony, R. (2010). Strength Pareto particle swarm optimization and hybrid EA-PSO for multi-objective optimization. *Evolutionary Computation*, *18*, 127–156.

Garcia, S., Molina, D., Lozano, M., & Herrera, F. (2009). A Study on the use of nonparametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics*, *15*(6), 617–644.

Grobler, J., & Engelbrecht, A. P. (2009). Hybridizing PSO and DE for improved vector evaluated multi-objective optimization. In *Proceedings of the IEEE congress on evolutionary computation, CEC 2009* (pp. 1255–1262). Trondheim: IEEE.

Harrison, K. R., Ombuki-Berman, B., & Engelbrecht, A. P. (2013). Knowledge transfer strategies for vector evaluated particle swarm optimization. In R. C. Purshouse, P. J. Fleming, C. M. Fonseca, et al. (Eds.), *Evolutionary multi-criterion optimization, EMO 2013* (Vol. 7811, pp. 171–184)., Lecture notes in computer science Berlin: Springer.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Cambridge: MIT Press.

Hu, Y.-F., Ding, Y.-S., Ren, L.-H., Hao, K.-R., & Han, H. (2015). An endocrine cooperative particle swarm optimization algorithm for routing recovery problem of wireless sensor networks with multiple mobile sinks. *Information Sciences*, *300*, 100–113.

Huband, S., Hingston, P., Barone, L., & While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, *10*, 477–506.

Hutter, F., Hoos, H. H., Leyton-Brown, K., & Stutzle, T. (2009). ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, *36*, 267–306.

Janson, S., Merkle, D., & Middendorf, M. (2008). Molecular docking with multi-objective particle swarm optimization. *Applied Soft Computing*, *8*, 666–675.

Kaveh, A., & Laknejadi, K. (2011). A novel hybrid charge system search and particle swarm optimization method for multi-objective optimization. *Expert Systems with Applications*, *38*, 15475–15488.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural network IV* (pp. 1942–1948). Piscataway: Springer

Khoshahval, F., Zolfaghari, A., Minuchehr, H., & Abbasi, M. R. (2014). A new hybrid method for multi-objective fuel management optimization using parallel PSO-SA. *Progress in Nuclear Energy*, *76*, 112–121.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Leong, W.-F., & Yen, G. G. (2008). PSO-based multiobjective optimization with dynamic population size and adaptive local archives. *IEEE Transactions on Systems, Man, and Cybernetics Part B (Cybernetics)*, *38*, 1270–1293.

Li, X. (2003). A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *Proceedings of the genetic and evolutionary computation conference, GECCO 2003* (pp. 37–48). Chicago: Springer.

Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, *10*, 281–295.

Liang, J. J., Qu, B.-Y., Suganthan, P. N., & Niu, B. (2012). Dynamic multi-swarm particle swarm optimization for multi-objective optimization problems. In *Proceedings of the IEEE Congress on evolutionary computation, CEC 2012* (pp. 1–8). Brisbane: IEEE.

Lim, K. S., Buyamin, S., Ahmad, A., Shapiai, M. I., Naim, F., Mubin, M., et al. (2014). Improving vector evaluated particle swarm optimization using multiple nondominated leaders. *The Scientific World Journal*, *2014*, 1–21.

Lim, K. S., Ibrahim, Z., Buyamin, S., Ahmad, A., Naim, F., Ghazali, K. H., et al. (2013). Improving vector evaluated particle swarm optimization by incorporating nondominated solutions. *The Scientific World Journal*, *2013*, 1–19.

Lim, W. H., & Isa, N. A. M. (2014). Teaching and peer-learning particle swarm optimization. *Applied Soft Computing*, *18*, 39–58.

Lin, Q., Li, J., Du, Z., Chen, J., & Ming, Z. (2015). A novel multi-objective particle swarm optimization with multiple search strategies. *European Journal of Operational Research*, *247*, 732–744.

Liu, D. S., Tan, K. C., Huang, S. Y., Goh, C. K., & Ho, W. K. (2008). On solving multiobjective bin packing problems using evolutionary particle swarm optimization. *European Journal of Operational Research*, *190*, 357–382.

Luo, J., Qi, Y., Xie, J., & Zhang, X. (2015). A hybrid multi-objective PSO-EDA algorithm for reservoir flood control operation. *Applied Soft Computing*, *34*, 526–538.

Meng, A., Chen, Y., Yin, H., & Chen, S. (2014). Crisscross optimization algorithm and its application. *Knowledge-Based Systems*, *67*, 218–229.

Mezura-Montes, E., Reyes-Sierra, M., & Coello-coello, C. A. (2008). Multi-objective optimization using differential evolution: A Survey of the state-of-the-art. In U. K. Chakraborty (Ed.), *Advances in differential evolution* (Vol. 144, pp. 173–196). Berlin: Springer.

Mirjalili, S., & Hashim, S. Z. M. (2010). A new hybrid PSOGSA algorithm for function optimization. In *Proceedings of the international conference on computer and information application, ICCIA 2010* (pp. 374–377). Tianjin: IEEE.

Moore, J., & Chapman, R. (1999). *Application of particle swarm to multiobjective optimization*. Technical Report, Department of Computer Science and Software Engineering, Auburn University.

Neumuller, C., Wagner, S., Kronberger, G., & Affenzeller, M. (2012). Parameter meta-optimization of metaheuristic optimization algorithms. In R. Moreno-Diaz, F. Pichler, & A. Quesada-Arencibia (Eds.), *Computer aided systems theory-EUROCAST 2011* (Vol. 6927, pp. 367–374). Berlin: Springer. Lecture notes in computer science.

Parsopoulos, K. E., Tasoulis, D. K., & Vrahatis, M. N. (2004). Multiobjective optimization using parallel vector evaluated particle swarm optimization. In *Proceedings of the IASTED international conference on artificial intelligence and applications, as part of the 22nd IASTED international multi-conference on applied informatics* (pp. 823–828). Innsbruck: ACTA Press.

Peng, G., Fang, Y.-W., Peng, W.-S., Chai, D., & Xu, Y. (2016). Multi-objective particle optimization algorithm based on sharing-learning and dynamic crowding distance. *Optik: International Journal for Light and Electron Optics*, *127*, 5013–5020.

Ramesh, S., Kannan, S., & Baskar, S. (2012). Application of modified NSGA-II algorithm to multi-objective reactive power planning. *Applied Soft Computing*, *12*, 741–753.

Ramteke, M., Ghune, N., & Trivedi, V. (2015). Simulated binary jumping gene: A step towards enhancing the performance of real-coded genetic algorithm. *Information Sciences*, *325*, 429–454.

Rangaiah, G. P. (2009). *Multi-objective optimization: Techniques and applications in chemical engineering*. Singapore: World Scientific.

Rangaiah, G. P., & Petriciolet, A. B. (2013). *Multi-objective optimization in chemical engineering: Developments and applications*. Oxford: Wiley.

Reyes-sierra, M., & Coello-coello, C. A. (2006). Multi-objective particle swarm optimizers: A survey of the state of the art. *International Journal of Computational Intelligence Research*, *2*, 287–308.

Sadeghbeii, R. (2000). *Fluid Catalytic Cracking Handbook* (2nd ed.). Houston, TX: Gulf Publishing Company.

Sengupta, S., Basak, S., & Peters, R. A, I. I. (2019). Particle swarm optimization: A survey of historical and recent developments with hybridization perspective. *Machine Learning and Knowledge Extraction*, *1*, 157–191.

Tang, K., Peng, F., Chen, G., & Yao, X. (2014). Population-based algorithm portfolios with automated constituent algorithms selection. *Information Sciences*, *279*, 94–104.

Thierens, D., & Goldberg, D. (1994). Elitist recombination: An integrated selection recombination GA. In *Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence* (pp. 508–512). Piscataway: IEEE.

Tripathi, P. K., Bandyopadhyay, S., & Pal, S. K. (2007). Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. *Information Sciences*, *177*, 5033–5049.

Valadi, J., & Siarry, P. (2014). *Applications of metaheuristics in process engineering*. New York: Springer.
Varshney, P., Kunzru, D., & Gupta, S. K. (2015). Modelling of the riser reactor in a resid fluidised-bed catalytic cracking unit using a multigrain model for an active matrix-zeolite catalyst. *Indian Chemical Engineer*, *57*, 115–135.
Wang, Y., & Yang, Y. (2009). Particle swarm optimization with preference order ranking for multi-objective optimization. *Information Sciences*, *179*, 1944–1959.
Xu, G., Liu, B., Song, J., Xiao, S., & Wu, A. (2019). Multiobjective sorting-based learning particle swarm optimization for continuous optimization. *Natural Computing*, *18*, 313–331.
Xu, G., Yang, Y., Liu, B.-B., Xu, Y., & Wu, A. (2015). An efficient hybrid multi-objective particle swarm optimization with a multi-objective dichotomy line search. *Journal of Computational and Applied Mathematics*, *280*, 310–326.
Yang, W.-C. (2003). *Handbook of fluidization and fluid-particle systems*. New York: Marcel Dekker Inc.
Yu, K., Wang, X., & Wang, Z. (2016). Multiple learning particle swarm optimization with space transformation perturbation and its application in ethylene cracking furnace optimization. *Knowledge-Based Systems*, *96*, 156–170.
Zhang, W.-J., & Xie, X.-F. (2003). DEPSO: Hybrid particle swarm with differential evolution operator. *IEEE international conference on systems, man and cybernetics, 2003* (pp. 3816–3821). IEEE: Washington, DC.
Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, *8*, 173–195.