Artif Intell Rev (2016) 45:131–165
DOI 10.1007/s10462-015-9445-7

Particle swarm optimization with crossover: a review and empirical analysis

A. P. Engelbrecht[1]

Published online: 10 October 2015

Abstract Since its inception in 1995, many improvements to the original particle swarm optimization (PSO) algorithm have been developed. This paper reviews one class of such PSO variations, i.e. PSO algorithms that make use of crossover operators. The review is supplemented with a more extensive sensitivity analysis of the crossover PSO algorithms than provided in the original publications. Two adaptations of a parent-centric crossover PSO algorithm are provided, resulting in improvements with respect to solution accuracy compared to the original parent-centric PSO algorithms. The paper then provides an extensive empirical analysis on a large benchmark of minimization problems, with the objective to identify those crossover PSO algorithms that perform best with respect to accuracy, success rate, and efficiency.

Keywords Swarm intelligence · Particle swarm optimization · Crossover · Boundary constrained optimization

1 Introduction

This paper focuses on one class of PSO variations, i.e. those PSO algorithms that make use of some form of crossover. While a number of crossover PSO algorithms exist, no comprehensive review of these algorithms exist. Furthermore, the papers that introduced these crossover PSO algorithms neglected to conduct a sensitivity analysis of the control parameters introduced due to the hybirdization with the respective crosover operators, and evaluated performance of these algorithms on a very small number of benchmark functions. Comparisons with other PSO algorithms were also not fairly done, due to the control parameters not being optimized for each of the algorithms on the used benchmark functions, and due to differences in swarm sizes and other experimental conditions. This paper addresses these issues by

✉ A. P. Engelbrecht
engel@cs.up.ac.za

[1] Department of Computer Science, University of Pretoria, Pretoria, South Africa

- Providing an extensive review of existing crossover PSO algorithms,
- Conducting a sensitivity analysis of the control parameters introduced by the crossover process, and
- An elaborate empirical comparison of these crossover PSO algorithms in order to determine which crossover PSO algorithm(s) performs best with respect to accuracy, success rate, and efficiency.

In addition, two adaptations of a parent-centric crossover (PCX) PSO algorithm are proposed and shown to improve on the performance of the original PCX PSO algorithm.

The remainder of the paper is organized as follows: Sect. 2 provides a short review of PSO. Section 3 follows with the review of crossover PSO algorithms. The two adaptations of the PCX PSO are given in Sect. 3.4. Section 4 presents and discusses the results of the control parameter sensitivity analysis and the comparison of the crossover PSO algorithms.

2 Particle swarm optimization

PSO is a stochastic, population-based optimization algorithm developed by Kennedy and Eberhart (1995). This section provides a short review of the standard PSO, mainly to introduce the basic elements of PSO and the notation used throughout the remainder of this review.

PSO algorithms maintain a swarm of particles, where each particle represents a candidate solution to the optimization problem. Each particle position is adapted based on two attractors: the best position found by the particle and the best position found by the particle's neighborhood. The standard PSO as introduced by Kennedy and Eberhart (1995) implemented one of two neighborhood topologies: either a star topology where a particle's neigborhood is the entire swarm, or a ring neighborhood topology where a particle's neighborhood is defined by its immediate neighbors. While a number of different neighborhood topologies have been developed (Kennedy 1999; Kennedy and Mendes 2002; Abdelbar and Abdelshahid 2003), this study uses the star neighborhood and the resulting global best PSO (gbest PSO), due to the finding in Engelbrecht (2013a) that there is no statistically significant difference in the performance between the two neighborhood topologies, considering a large benchmark set of 60 functions.

For the gbest PSO, particle positions $\mathbf{x}_i$ are updated using

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{1}$$

and the velocities are updated using the inertia weight model (Shi and Eberhart 1998) as follows

$$v_{ij}(t+1) = w v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \tag{2}$$

where w is the inertia weight, $v_{ij}(t)$ is the velocity of particle i in dimension $j = 1, \ldots, n_x$ at time step t, $x_{ij}(t)$ is the position of particle i in dimension j at time step t, $y_{ij}(t)$ is particle i's personal best position in dimension j, $\hat{y}_j(t)$ is the global best position in dimension j, c_1 and c_2 are positive acceleration constants used to scale the contribution of the cognitive and social components respectively, n_x is the dimension of the search space, and $r_{1j}(t), r_{2j}(t) \sim U(0, 1)$ are uniform random values in the range [0, 1], sampled from a uniform distribution.

For the purposes of this study, the global best position is selected as the personal best position with the best solution quality, and a personal best position is only updated if the new particle position is better than the previous particle position. In addition, all PSO algorithms used in this study implements a boundary constraint on personal best positions where a

personal best position is only updated if no boundary constraints are violated. This ensures that all personal best positions, and therefore also the global best position, remain within the boundaries of the search space (Engelbrecht 2013c). Particle positions are initialized within the domain of the optimization problem and velocities are initialized to zero (Engelbrecht 2012).

3 Particle swarm optimization with crossover

Many improvements have been made to the standard PSO (Engelbrecht 2005; Poli et al. 2007; Sedighizadeh and Masehian 2009). Hybrid PSO algorithms have been developed that incorporate operators from evolutionary algorithms into PSO. Many PSO hybrids with evolutionary algorithms make use of some form of mutation, applied to either velocity vectors or position vectors (Miranda and Fonseca 2002; van den Bergh and Engelbrecht 2002; Wei et al. 2002; Higashi and Iba 2003; Stacey et al. 2003). Only a few studies can be found where crossover operators were used within PSO algorithms. This section reviews these PSO hybrids with crossover operators.

3.1 Arithmetic crossover

The very first application of arithmetic crossover within PSO was by Løvberg et al. (2001). A breeding process is done for each iteration after the velocity and position updates have been done. Two particles are randomly selected for breeding through application of a crossover operator at a user-specified breeding probability. Assume that particles a and b are selected for crossover. The corresponding positions, $\mathbf{x}_a(t)$ and $\mathbf{x}_b(t)$ are then replaced by the offspring,

$$\mathbf{x}_{i_1}(t+1) = \mathbf{r}(t)\mathbf{x}_{i_1}(t) + (\mathbf{1} - \mathbf{r}(t))\mathbf{x}_{i_2}(t)$$
$$\mathbf{x}_{i_2}(t+1) = \mathbf{r}(t)\mathbf{x}_{i_2}(t) + (\mathbf{1} - \mathbf{r}(t))\mathbf{x}_{i_1}(t)$$

with the corresponding velocities,

$$\mathbf{v}_{i_1}(t+1) = \frac{\mathbf{v}_{i_1}(t) + \mathbf{v}_{i_2}(t)}{||\mathbf{v}_{i_1}(t) + \mathbf{v}_{i_2}(t)||}||\mathbf{v}_{i_1}(t)||$$
$$\mathbf{v}_{i_2}(t+1) = \frac{\mathbf{v}_{i_1}(t) + \mathbf{v}_{i_2}(t)}{||\mathbf{v}_{i_1}(t) + \mathbf{v}_{i_2}(t)||}||\mathbf{v}_{i_2}(t)||$$

where $\mathbf{r}_1(t) \sim U(0,1)^{n_x}$. The personal best position of an offspring is initialized to its current position. The resulting algorithm is referred to in this paper as PSO with arithmetic crossover (PSO-AX).

3.2 Novel multi-parent crossover

Wang et al. (2008) proposed the novel multi-parent crossover operator (NMPCO), where an offspring is produced by recombining three randomly selected particles at a user-specified crossover probability. For each particle i, if indices a, b and c with $a \neq b \neq c$ represent the selected particles, then the offspring is calculated as

$$\tilde{\mathbf{x}}_i(t) = k_1\mathbf{x}_i(t) + k_2\mathbf{x}_a(t) + k_3\mathbf{x}_b(t) + k_4\mathbf{x}_c(t)$$

where $k_1, k_2, k_3, k_4 \sim U(0,1)$, and further calculated as $k_l = 5(k_l / \sum_{c=1}^{4} k_c) - 1$ for $l = 1, 2, 3, 4$. If the quality of the offspring $\tilde{\mathbf{x}}_i(t)$ is better than that of parent particle $\mathbf{x}_i(t)$,

then the offspring replaces the parent particle. Personal best and global best positions are updated after application of the NMPCO. As an optional last step, the global best position is mutated as follows:

$$\hat{y}_j(t+1) = \hat{y}_j(t) + \overline{v}_j(t) C_j(x_{j,min}, x_{j,max})$$

for each dimension $j = 1, \ldots, n_x$, where C is a Cauchy distributed function with scale of one, $C_j(x_{j,min}, x_{j,max})$ is a random number within $[\mathbf{x}_{min}, \mathbf{x}_{max}]$, and

$$\overline{v}_j(t) = \frac{\sum_{i=1}^{n_x} v_{ij}(t)}{n_s}$$

is the average velocity of the swarm, with n_s the number of particles in the swarm. The average velocity is restricted to be within $[-V_{max}, V_{max}]$.

3.3 Blend crossover

Duong et al. (2010) developed the hybrid evolutionary algorithm as a hybrid between PSO and the blend crossover (BLX-α) operator (Eshelman and Schaffer 1993). Using BLX-α, offspring is generated as

$$\tilde{x}_{ij}(t) = (1 - \gamma_j) x_{1j}(t) + \gamma_j x_{2j}(t)$$

with $\gamma_j = (1 + 2\alpha) U(0, 1) - \alpha$. The BLX-$\alpha$ operator randomly picks, for each dimension, a random value in the range

$$[x_{1j}(t) - \alpha(x_{2j}(t) - x_{1j}(t)), x_{2j}(t) + \alpha(x_{2j}(t) - x_{1j}(t))]$$

BLX-α assumes that $x_{1j}(t) < x_{2j}(t)$. Parents are selected based on solution quality using roulette wheel selection, and an offspring is generated at a user-specified crossover probability. The personal best position of the offspring particle is set to the offspring's position. This process is performed after all particle positions and velocities have been updated. Once all offspring have been generated, particles and offspring are ranked based on solution quality, and only the n_s best positions survive to the next iteration. The resulting algorithm is referred to as the PSO-BLX-α.

3.4 Parent-centric crossover

Deb et al. (2002) developed the parent-centric crossover (PCX) operator as a variation of the unimodal distributed (UNDX) operator (Ono and Kobayashi 1997). The PCX operator implements a multi-parent crossover which blends the candidate solutions represented by the selected parents to produce a single offspring. Where the UNDX operator samples offspring around the center of mass of the selected parents, PCX samples offspring around the selected parents as illustrated in Fig. 1. PCX has a higher probability of creating an offspring closer to a parent than UNDX.

PCX selects $n_\mu \geq 3$ parents and computes their center of mass, $\overline{\mathbf{x}}(t)$. For each offspring to be generated, one parent, $\mathbf{x}_i$, is selected from the n_μ parents; usually, the best parent is selected. This parent is then mutated to produce an offspring as follows: A direction vector is calculated as

$$\mathbf{d}_i(t) = \mathbf{x}_i(t) - \overline{\mathbf{x}}(t)$$

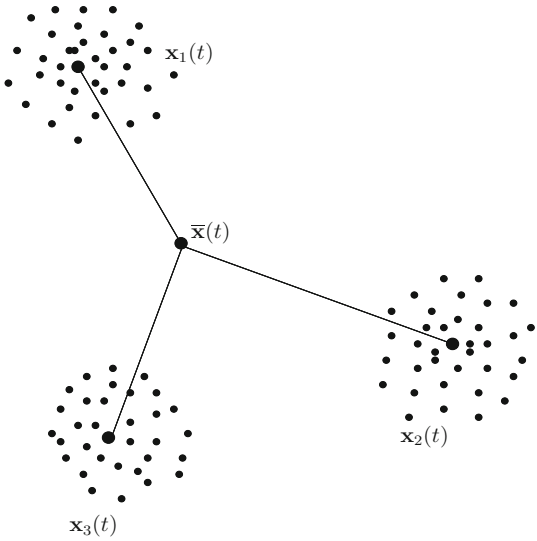

Fig. 1 Illustration of parent-centric crossover

where $\overline{\mathbf{x}}(t)$ is the center of mass,

$$\overline{\mathbf{x}}(t) = \sum_{l=1}^{n_\mu} \mathbf{x}_l(t)$$

of the selected parents. From the other $n_\mu - 1$ parents perpendicular distances, δ_l, for $i \neq l = 1, \ldots, n_\mu$, are calculated to the line $\mathbf{d}_i(t)$. The average over these distances is calculated, i.e.,

$$\overline{\delta} = \frac{\sum_{l=1, l\neq i}^{n_\mu - 1} \delta_l}{n_\mu - 1}$$

An offspring is then generated using

$$\tilde{\mathbf{x}}_i(t) = \mathbf{x}_i(t) + N(0, \sigma_1^2)|\mathbf{d}_i(t)| + \sum_{l=1, i\neq l}^{n_\mu - 1} N(0, \sigma_2^2)\overline{\delta}\mathbf{e}_l(t) \tag{3}$$

where $\mathbf{e}_l(t)$ are the $n_\mu - 1$ orthonormal bases that span the subspace perpendicular to $\mathbf{d}_i(t)$, and σ_1 and σ_2 are the deviations of the two Gaussian distributions. Using Eq. (3), offspring are generated by mutating a randomly selected parent, $\mathbf{x}_i$, with a stochastic weighted distance that the parent is from the center of mass of all selected parents and a stochastic weighted distance that each non-mutated parent is from the direction vector, $\mathbf{d}_i(t)$. If σ_1 is small, then the offspring is generated closer to the parent being mutated. For larger values of σ_2, offspring generation is biased towards the non-mutated parents.

Deb and Padhye (2010) and Padhye (2010) proposed two PCX velocity update strategies, both replacing the standard PSO velocity update with a PCX operator applied to the particle's position, its personal best position, and the global best position. For the first strategy, referred to in this work as PSO-PCX$_{\hat{y}}$, the parent to be mutated in Eq. (3), $\mathbf{x}_i(t)$, is always the global best position. Padhye (2010) reported that the PSO-PCX$_{\hat{y}}$ did not perform well, and showed that the bad performance is due to the personal best position of a particle always being similar to the particle's position for unimodal problems, and frequently for multimodal problems.

When this happens the PCX operator produces an offspring along the line joining the particle's position and the global best position, effectively performing a line search. This finding was, however, based on limited results on only three benchmark functions and without optimizing the parameters σ_1 and σ_2.

As a remedy to this problem, a strategy was proposed to ensure that the parents used in the PCX operator are distinct (Deb and Padhye 2010; Padhye 2010). The second strategy, referred to in this study as PSO-PCX$^*_{\hat{y}}$, therefore first checks if the parents are distinct. If so, the PCX operator is applied as for PSO-PCX$_{\hat{y}}$; if not, three distinct parents are selected from the particle's previous position, its current position, its personal best position, and the global best position. If three distinct parents could be found, the PCX operator is applied on these three solutions. If only two distinct solutions could be found, the PCX operator is applied with $\overline{\delta} = 0$ in Eq. (3). If all of the solutions are identical, the PCX operator is not applied, but the standard velocity update as in Eq. (2) is applied instead.

The effect of the PCX velocity update strategies as proposed by Deb and Padhye is an exploitation of the global best position due to use of the global best position as the parent to be mutated. In order to improve exploration, a mutation operator was included (Deb and Padhye 2010; Padhye 2010). Also note that the implementation in Deb and Padhye (2010) and Padhye (2010) used a steady state implementation of the PSO, that is, asynchronous updates of particle positions and best positions.

Worasucheep et al. (2012) proposed an alternative application of the PCX operator in the PSO, using asynchronous updates of particle positions and best positions. Before each particle position update, the PCX operator is applied at a crossover probability, p_c. If a random number sampled from a uniform distribution between 0 and 1 is greater than p_c, then the particle position is updated using the constriction PSO (Clerc and Kennedy 2002); otherwise three parents are selected, two randomly from the current particle positions and one as the global best particle, and the PCX operator is used to generate two offspring. The randomly selected parents are then replaced by the best two positions from the subset created as the union of these two parents and the two offspring. The resulting algorithm is referred to as the PSO-PSPG.

3.5 Discrete crossover

Park et al. (2010) proposed that a discrete crossover operator be used to update personal best positions. A trial position, $\mathbf{x}'(t+1)$ is computed using

$$x'_{ij} = \begin{cases} x_{ij}(t+1) & \text{if } r_{ij} < p_c \\ y_{ij}(t) & \text{otherwise} \end{cases} \tag{4}$$

where $r_{ij} \sim U(0, 1)$ and p_c is the probability of crossover. The current personal best position is replaced with this trial position if the latter represents a better solution.

Dong and Yang (2010) proposed to update particle positions using a crossover between a particle's new position and the global best position:

$$x'_{ij} = \begin{cases} x_{ij}(t+1) & \text{if } r_{ij} < p_c \\ \hat{y}_j(t) & \text{otherwise} \end{cases} \tag{5}$$

The trial position replaces the particle position if the trial position represents a better solution. The resulting algorithm is referred to as PSO-DX$_{\hat{y}}$.

Engelbrecht (2013b) proposed six discrete crossover operators for PSO, assuming synchronous updates, and in Engelbrecht (2014) using asynchronous position updates. After each particle position update, the crossover operator is applied at a specified crossover prob-

ability. The first parent is the new particle position and the second parent is selected using one of the following strategies:

- Borrowing from the proposal of Park et al. (2010), the second parent is selected as the personal best position of the particle. The resulting PSO algorithm is referred to as PSO-DX$_y$.
- Borrowing from the proposal of Dong and Yang (2010), the second parent is the global best position. The resulting PSO algorithm is referred to as PSO-DX$_{\hat{y}}$.
- The second parent is calculated as a weighted average of the personal and global best positions, i.e.

$$r\mathbf{y}_i(t) + (1-r)\hat{\mathbf{y}}(t) \tag{6}$$

with $r \sim U(0,1)$. The resulting algorithm is referred to as PSO-DX$_{y\hat{y}}$.

If the generated offspring is better than the new particle position, then the offspring replaces the new particle position. If the new particle position is replaced with the offspring, the personal best position is set to the position represented by the offspring and the velocity is set to zero.

For each of the three strategies above, recombination of the two parents is either one-point recombination where one crossover point is randomly selected, or uniform where each dimension has an equal probability of being selected as a crossover point. If one-point recombination is used, the superscript 1 is added to the algorithm name, while the superscript u is added for uniform recombination.

3.6 Parent-centric crossover adaptations

The two PSO-PCX algorithms discussed in Sect. 3.4 selects the global best particle to be mutated in Sect. 3. Offspring are then generated as an exploitation of the global best particle, where the level of exploitation depends on the values of σ_1 and σ_2. As an alternative, this paper proposes that a random particle position be selected to be mutated, in an effort to increase the exploration abilities of the swarm. The two resulting algorithms are referred to in this paper as PSO-PCX$_r$ and PSO-PCX$_r^*$.

4 Empirical analysis

This section conducts an in-depth empirical analysis of the crossover PSO algorithms. As the very first step, sensitivity analysis of the crossover specific control parameters is done in Sect. 4.2, followed by a test to see if each of the PSO crossover algorithms actually does improve the performance of the baseline PSO algorithm in Sect. 4.3. The new PCX adaptations are compared with the original ones to see if any gain can be obtained by selecting the particle position to be mutated randomly. The empirical analysis ends in a comparison of all of the PSO crossover algorithms in Sect. 4.3. The empirical procedure followed is described next in Sect. 4.1.

4.1 Empirical procedure

Detail of the experimental procedure followed is provided in this section. The algorithms used in the comparisons are listed in Sect. 4.1.1. Setting of the control parameters of the baseline PSO algorithm is also discussed. The benchmark set used is described in Sect. 4.1.2. The

Table 1 Control parameters of algorithms used in the empirical study

Algorithm	Control parameters
PSO-AX	p_c, probability of crossover
PSO-BLX	p_c, probability of crossover
	α, scale parameter
PSO-NMPCO	p_c, probability of crossover
PSO-PSPG	σ_1, σ_2, width of Gaussians's deviations
	p_c, probability of crossover
PSO-PCX	σ_1, σ_2, width of Gaussians's deviations
PSO-DX	p_c, probability of crossover

performace measures used are described in Sect. 4.1.3 and the statistical analysis procedure is summarized in Sect. 4.1.4.

4.1.1 Algorithms

Section 3 described a number of PSO algorithms that use crossover operators in some form. In addition to the crossover operators and how they are used within the PSO, these algorithms also differ in the baseline algorithm used. Furthermore, algorithms such as the NMPCO and PSO-PCX$^*_{\hat{y}}$ use an additional mutation operator, while PSO-PCX$_{\hat{y}}$, PSO-PCX$^*_{\hat{y}}$, and PSO-PSPG use asynhronous updates. Since the objective of this study is to determine the contribution that the specific crossover operators make to possible performance improvements, the same baseline algorithm was used for all of the different crossover PSO algorithms. The baseline PSO algorithm is a global best PSO (gbest PSO), with synchronous updates. Velocities, $\mathbf{v}_i(0)$, were initialized to zero for all particles $i = 1, \ldots, n_s$, and velocity clamping was not used. To ensure that found solutions do not violate boundary constraints, personal best positions were only updated if the resulting position is better and satisfies the boundary constraints. The gbest PSO control parameters were set as follows: Swarm sizes, n_s, were set to 30 particles, the inertia weight, w, was set to 0.729844, while the values of the acceleration coefficients, c_1 and c_2, were both set to 1.49618. This choice is based on Eberhart and Shi (2000), and results in convergent particle trajectories (van den Bergh and Engelbrecht 2006; Trelea 2003; Cleghorn and Engelbrecht 2014). All algorithms were run for 5000 iterations.

The algorithms used in the sensitivity analysis and comparison, and their control parameters are summarized in Table 1. All algorithms are implemented in CIlib version 1.0, an opensource library of Computational Intelligence algorithms.[1]

4.1.2 Benchmark functions

A total of 61 benchmark functions, as described in Engelbrecht (2013a), have been used. These functions represent a wide range of problem characteristics, including unimodal, multimodal, separable, non-separable, rotated, shifted, noisy, and composition functions. A summary of the functions and their characteristics is provided in "Appendix". Furthermore, as shown in Garden and Engelbrecht (2014), these functions cover a representative range of different landscape characteristics, including rugged landscapes, smooth landscapes, and landscapes with different gradients (from steep to shallow) and large deviations among

[1] http://www.cilib.net.

gradients. Also note that functions f_{26}–f_{37}, listed in the "Appendix", are functions used in real-valued optimization competitions (Suganthan et al. 2005). Due to the large coverage of problem and landscape characteristics, this benchmark set is comprehensive and represents from easy to very difficult optimization problems. All of the functions were used in 30 dimensions, and in the domains as specified in the "Appendix".

4.1.3 Performance measures

The PSO algorithms are compared with respect to three performance measures:

- *Accuracy*, which is the quality of the global best particle after completion of the 5000 iterations.
- *Success rate*, which is the percentage of the total number of independent runs that reached specified error levels. For the purpose of this study, a total of 1000 accuracy levels have been considered, with the accuracy levels starting at the best accuracy obtained by the two algorithms, logarithmically increasing towards the worst accuracy.
- *Efficiency*, computed as the number of iterations to reach the different accuracy levels.
- *Consistency*, computed as the deviation from the average best function value obtained over the 50 independent runs.

4.1.4 Statistical procedure

Fifty independent runs were executed for each PSO algorithm on each of the benchmark functions. For each function, the Mann–Whitney U test was used to indicate difference in performance with respect to a specific performance measure at a significance level of 0.05. For each function class, the total number of wins for each algorithm was calculated as well as the number of functions for which there was no statistically significant difference in performance. These scores are reported in the tables to follow. With reference to the success rate, the samples to which the Mann–Whitney U test was applied consisted of the success rates for each of the accuracy levels. Therefore, success rate scores indicate how successful each algorithm was over the entire range of accuracy levels. With reference to the efficiency performance measure, the samples consisted of the average number of iterations over the 50 independent runs for each of the accuracy levels. A win based on the Mann–Whitney U test therefore indicates that the corresponding algorithm converged faster to most of the accuracy levels.

4.2 Control parameter sensitivity analysis

The studies that have introduced the existing crossover PSO algorithms have not provided an analysis of the sensitivity of the algorithms to the crossover related control parameters. Furthermore, most of these studies did not even attempt to find the best parameter values to use. The purpose of this section is to investigate the effect that different parameter value combinations have on performance (with reference to accuracy), and to determine the best parameter values to use for each of the benchmark functions. The empirical anlyses in subsequent sections use these best parameter values in the comparisons done.

For the parameter sensitivity analysis, the statistical procedures described in Sect. 4.1.4 was used to compute the sum of wins and losses for each parameter value combination with respect to the fitness of the best found solution for each of the 50 independent runs. For each of the benchmark functions, the parameter value combinations were then ranked based on

Table 2 PSO-AX parameter sensitivity analysis

	Crossover probability			
	0.2	0.4	0.6	0.8
Average rank	1.229	1.803	2.754	3.279
Rank deviation	0.643	0.542	0.869	1.142
Best bank frequency	46	3	4	8

Table 3 PSO-NMPCO parameter sensitivity analysis

	Crossover probability			
	0.2	0.4	0.6	0.8
Average rank	2.279	1.820	1.410	1.279
Rank deviation	1.362	0.958	0.528	0.777
Best rank frequency	8	6	9	38

the sum of wins and losses. The average rank for each parameter value combination was computed over all of the functions, as well as the standard deviation and the number of times that each parameter combination resulted in the best rank. Note that a lower rank value indicates a better rank.

For the PSO-AX's crossover probability, values of $0.2, 0.4, 0.6$ and 0.8 were investigated. Table 2 summarizes the ranks for these parameter values, showing clearly that a small crossover probability of 0.2 is preferred. For 75.41 % of the functions, a 0.2 crossover probability resulted in the best rank. Also note that the performance of PSO-AX is very sensitive to the crossover probability: For only three of the functions were there no statistically significant difference in fitness of the best solutions for the different crossover probability values. For the rest of the functions, a significant difference in performance was found between at least two control parameter values.

For the NMPCO, as illustrated in Table 3, larger values for p_c are preferred, with 62.3 % of the functions resulting in the best performance when $p_c = 0.8$. Note that the NMPCO is insensitive to the crossover probability control parameter for 29 of the 61 functions, showing no significant difference in performance for different crossover probabilities.

The PSO-BLX has two crossover related control parameters. For the crossover probability, values of 0.2, 0.4, 0.6 and 0.8 were tried, and for the scale parameter, α, values from 0.2 up to 2 were tried in increments of 0.2. The results summarized in Table 4 show a preference for a large crossover probability and small scale parameters, α. The best rank and best rank frequency was obtained with $p_c = 0.8$ and $\alpha = 0.2$. For $p_c = 0.8$, irrespective of the value of α, 39 of the 61 functions provided the best performance. For $\alpha = 0.2$, irrespecitve of the value of p_c, 47 of the functions provided the best performance. No wins were obtained for $\alpha \geq 0.8$, and none for $p_c < 0.4$. For none of the functions was a significant difference in performance over the different parameter combinations obtained, indicating that though the preference for large p_c and small α exist, PSO-BLX is very sensitive to its control parameters.

Due to the three parameters of PSO-PSPG and space limitations, Table 5 summarizes only the average ranks for all parameter combinations. The performance of PSO-PSPG is shown to be very sensitive to the values of σ_2 and the crossover probability p_c. For specific values of σ_2 and p_c, PSO-PSPG is relatively insensitive to the value of σ_1. The best performances were obtained using a low crossover probability ($p_c \leq 0.3$) and a $\sigma_2 = 0.9$.

The PSO algorithms with discrete crossover each have only the crossover probability as control parameter. For this parameter, values of 0.2, 0.4, 0.6 and 0.8 were investigated. Table 6

Table 4 PSO-BLX parameter sensitivity analysis

α	Crossover probability											
	Average rank				Rank deviation				Best rank frequency			
	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
0.2	5.672	3.246	1.705	1.836	2.731	1.567	0.823	1.319	0	5	17	25
0.4	12.902	7.754	4.049	2.426	4.400	1.206	1.596	1.310	0	0	0	12
0.6	25.148	13.754	8.361	6.328	4.362	2.838	1.484	1.710	0	0	0	2
0.8	32.492	23.197	13.262	9.230	2.761	4.218	1.949	1.116	0	0	0	0
1.0	37.607	31.000	19.459	12.754	2.478	3.683	2.742	1.650	0	0	0	0
1.2	39.131	35.213	23.475	15.377	2.247	3.061	2.599	1.507	0	0	0	0
1.4	39.508	37.836	27.213	16.902	2.481	3.067	2.599	1.507	0	0	0	0
1.6	40.574	40.344	30.180	18.934	2.493	2.421	3.314	2.366	0	0	0	0
1.8	40.639	40.754	32.066	20.230	2.288	2.102	3.669	2.479	0	0	0	0
2.0	41.705	41.639	32.082	21.066	2.116	1.674	3.730	2.414	0	0	0	0

Table 5 PSO-PSPG parameter sensitivity analysis

σ_1	p_c	σ_2				
		0.1	0.3	0.5	0.7	0.9
0.1	0.1	38.164	24.902	16.754	12.525	6.803
	0.3	64.820	42.934	27.148	15.279	9.197
	0.5	80.672	57.066	37.803	24.311	15.738
	0.7	100.770	74.311	54.689	36.557	27.836
	0.9	120.623	99.377	73.885	51.410	46.836
0.3	0.1	37.541	30.623	21.311	14.049	6.918
	0.3	53.443	39.639	27.115	16.066	8.361
	0.5	75.689	52.148	39.246	23.246	12.443
	0.7	97.410	70.984	50.508	32.984	25.672
	0.9	120.721	87.639	65.459	47.770	46.033
0.5	0.1	33.492	29.574	18.902	15.393	6.131
	0.3	51.525	36.377	24.885	16.541	9.049
	0.5	69.754	55.180	35.475	23.361	15.344
	0.7	91.852	70.836	49.574	34.508	23.787
	0.9	109.607	87.066	63.148	45.689	41.148
0.7	0.1	36.705	31.623	19.869	13.328	8.213
	0.3	50.131	34.541	23.787	16.164	8.689
	0.5	71.820	52.967	37.131	21.623	12.328
	0.7	92.066	66.557	47.803	33.115	25.721
	0.9	107.672	83.984	62.967	44.197	38.016
0.9	0.1	41.721	28.754	22.131	15.656	8.230
	0.3	48.541	31.426	26.213	17.508	9.902
	0.5	69.311	49.787	33.672	23.049	17.541
	0.7	82.820	67.115	47.557	30.918	22.967
	0.9	98.426	80.262	57.705	42.361	36.213

Table 6 PSO-DX parameter sensitivity analysis

	Crossover probability			
	0.2	0.4	0.6	0.8
$PSO\text{-}DX^{u}_{\hat{y}}$				
Average rank	2.082	2.049	2.328	2.607
Rank deviation	1.242	0.990	1.221	1.406
Best rank frequency	26	13	12	10
$PSO\text{-}DX^{1}_{y}$				
Average rank	3.098	2.246	1.885	1.869
Rank deviation	1.287	0.994	0.933	1.284
Best rank frequency	10	9	12	30
$PSO\text{-}DX^{u}_{y}$				
Average rank	3.295	2.459	1.705	1.623
Rank deviation	1.243	1.026	0.919	0.952
Best rank frequency	7	5	20	29

summarizes the results for the three PSO-DX algorithms. For the PSO-DX$^{u}_{\hat{y}}$, a preference for low crossover probabilities is seen. For both the PSO-DX$^{1}_{y}$ and PSO-DX$^{u}_{y}$, a preference for a high crossover porbability is indicated. Because all the p_c values have a number of functions for which best performance was obtained, the PSO-DX algorithms are sensitive to the value of p_c.

The PSO-PCX algorithms all have two control parameters, σ_1 and σ_2, which controls the widths of the Gaussians used to create offspring. Both parameters were evaluated for values of 0.1, 0.3, 0.5, 0.7 and 1.0. Table 7 shows a definite preference for large σ_1 and σ_2 values for the PSO-PCX$_{\hat{y}}$. For $\sigma_2 = 1.0$, irrespective of the value of σ_2, the best ranks were obtained. For no $\sigma_2 < 1.0$ was best performance for any of the functions obtained. For $\sigma_2 = 1.0$, all of the σ_1 values have a number of functions for which the best performance was obtained, but most for large σ_1 values. What is seen from these results is that the best value for σ_1 for the PSO-PCX$_{\hat{y}}$ is problem dependent, but not for σ_2.

Table 8 shows that PSO-PCX$^{*}_{\hat{y}}$ also has a preference for larger σ_1 and σ_2. However, this preference is not as strong as for the PSO-PCX$_{\hat{y}}$. Only parameter combination $\sigma_1 = \sigma_2 = 0.1$ had no functions for which the best performance was obtained. This stands in contrast to the value of 0.17 used by Deb and Padhye (2010), Padhye (2010) for both deviations. The results show that PSO-PCX$^{*}_{\hat{y}}$ is very sensitive to its control parameters, and that the best values are problem dependent.

As shown in Table 9, PSO-PCX$_r$ showed a strong preference for large σ_1 and σ_2, and more specifically so for σ_2. This trend is the same as observed for PSO-PCX$_{\hat{y}}$, but to a lesser extent. No value of $\sigma_2 < 0.5$ resulted in any function with best performance, and very few for $\sigma_2 = 0.5$ and $\sigma_2 = 0.7$. When the mutated individual is randomly selected, instead of selected as the global best, high sensitivity was shown for the choice of $\sigma_1 \in \{0.5, 0.7, 1.0\}$ when $\sigma_2 = 1.0$, with these all having a rank less than 3.0. What is thus noted, is that the best values for the control parameters are problem dependent.

For PSO-PCX$^{*}_{r}$, Table 10 shows again a prefernece for larger values for σ_1 and σ_2. However, the preference is not as strong as for PSO-PCX$^{*}_{\hat{y}}$. For only one function did $\sigma_2 = 0.1$ provide the best results, and a few functions for $\sigma_2 = 0.3$. Random selection of the individual to be mutated confirms again that the preference is for $\sigma_2 = 1.0$ and $\sigma_1 \geq 0.3$. Note that, again, the best values for the control parameters are problem dependent.

Table 7 PSO-PCX$_{\hat{y}}$ parameter sensitivity analysis

σ_2	σ_1														
	Average rank					Rank deviation					Best rank frequency				
	0.1	0.3	0.5	0.7	1.0	0.1	0.3	0.5	0.7	1.0	0.1	0.3	0.5	0.7	1.0
0.1	21.787	21.885	21.820	21.738	20.918	0.897	1.082	1.025	0.982	0.666	0	0	0	0	0
0.3	18.787	18.508	17.361	16.475	15.656	1.404	1.206	1.049	1.192	1.778	0	0	0	0	0
0.5	12.131	11.656	11.131	10.049	8.689	2.623	2.575	2.611	2.390	2.487	0	0	0	0	0
0.7	11.180	10.213	8.820	7.705	6.377	2.947	2.497	2.149	1.829	1.319	0	0	0	0	0
1.0	4.295	3.033	2.607	1.705	1.361	1.626	1.426	1.229	0.715	2.106	7	15	17	24	57

Table 8 PSO-PCX$_r$ parameter sensitivity analysis

σ_2	σ_1														
	Average rank					Rank deviation					Best rank frequency				
	0.1	0.3	0.5	0.7	1.0	0.1	0.3	0.5	0.7	1.0	0.1	0.3	0.5	0.7	1.0
0.1	22.803	22.918	21.197	21.148	20.902	1.641	1.754	1.526	1.167	1.739	0	0	0	0	0
0.3	17.885	17.295	16.852	16.590	15.852	2.602	1.944	2.159	1.131	1.062	0	0	0	0	0
0.5	13.607	12.984	12.377	11.377	10.377	2.216	2.109	2.010	2.192	2.346	1	0	0	2	1
0.7	8.574	7.721	6.934	6.279	5.475	2.686	2.517	2.428	2.464	2.767	5	5	4	4	5
1.0	4.492	3.377	2.852	2.721	2.754	3.080	2.788	2.744	3.541	4.170	14	16	17	17	47

Table 9 PSO-PCX$^{*}_{\hat{y}}$ parameter sensitivity analysis

σ_2	σ_1														
	Average rank					Rank deviation					Best rank frequency				
	0.1	0.3	0.5	0.7	1.0	0.1	0.3	0.5	0.7	1.0	0.1	0.3	0.5	0.7	1.0
0.1	23.951	22.311	20.902	18.590	14.082	1.575	4.880	5.002	5.878	7.251	0	2	2	3	5
0.3	20.213	17.754	15.475	12.85	2.016	4.875	4.739	4.938	5.250	5.211	3	3	4	5	9
0.5	17.311	13.672	10.754	8.869	6.328	4.787	4.285	4.361	4.562	3.325	3	3	6	8	7
0.7	12.885	9.197	7.721	5.639	4.639	5.241	4.041	3.158	2.823	2.690	3	6	6	9	6
1.0	7.672	5.590	4.639	2.902	3.279	5.549	3.662	3.430	2.638	4.680	7	7	6	13	42

Table 10 PSO-PCX$^*_r$ parameter sensitivity analysis

σ_2	σ_1														
	Average rank					Rank deviation					Best rank frequency				
	0.1	0.3	0.5	0.7	1.0	0.1	0.3	0.5	0.7	1.0	0.1	0.3	0.5	0.7	1.0
0.1	22.049	21.672	23.033	22.344	20.574	2.878	2.874	1.879	1.632	2.262	0	1	0	0	0
0.3	18.246	18.557	16.787	15.574	13.574	2.528	2.202	4.325	4.291	4.696	0	0	3	4	4
0.5	13.721	12.738	11.836	10.049	7.213	4.013	3.812	3.976	3.580	3.372	4	4	4	7	9
0.7	10.033	8.902	7.000	5.361	4.443	3.173	2.803	2.543	2.589	3.505	4	5	5	8	15
1.0	5.689	3.885	3.082	2.803	3.762	4.044	3.297	2.326	3.203	4.210	10	18	16	19	35

4.3 Comparison of results

The goal of this section is to analyze the performance of the different crossover PSO algorithms, using the best control parameter values per function as obtained as a result of the sensitivity analysis discussed in Sect. 4.2. This was done in three steps:

- The first step is to show if random selection of the mutated parent for the PCX-based PSO algorithms improves on the performance of the PSO-PCX$_{\hat{y}}$ algorithms where the mutated parent is selected as the global best particle.
- Secondly, the hypothesis that each crossover approach resulted in performance better than the base gbest PSO algorithms is evaluated.
- Thirdly, all of the crossover PSO algorithms are compared to try and find those that performed best.

The first two studies were done only with regards to the accuracy of the solutions found. The third study considered performance with respect to solution accuracy, success rate, and efficiency.

Table 11 shows that random selection of the parent to be mutated in the parent-centrix crossover significantly improved the performance of the PSO-PCX$_{\hat{y}}$ for all of the 61 functions. For the PSO-PCX$^*_{\hat{y}}$, random selection provided significantly better results for 29 (47.5 %) of the functions and similar performance for 19 (31.1 %) of the functions. For only 16 (21.4 %) of the functions did random selection performed significantly worse.

Table 12 summarizes the wins, draws and losses when each of the crossover PSO algorithms is compared with the base gbest PSO algorithm. Also indicated are the sum of the wins and losses, and a ranking of the algorithms based on wins, losses and the sum of wins and losses. The first thing to note is the very bad performance of PSO-BLX and PSO-PCX$_{\hat{y}}$. The former did not improve the performance of gbest PSO for any of the functions, while the latter provided better performance for only one of the functions. In addition to these two algorithms, PSO-PCX$_r$, PSO-PCX$^*_{\hat{y}}$, PSO-AX, and PSO-DX$^u_{\hat{y}}$ had more losses than wins.

The best performance improvement over gbest PSO was achieved with the PSO-DX^u_y, with 38 wins and only 6 losses. While PSO-NMPCO had the second most wins, it also had the fewest losses against gbest PSO. Other algorithms that showed more wins than losses are, in order from best to worst, PSO-DX^1_y, PSO-PCX$^*_r$, and PSO-PSG.

What should be noted from these results, compared to results already published for these algorithms, is that a number of the crossover PSO algorithms do not, in fact, perform as good as indicated in the literature despite being used with optimized control parameters. This is mainly due to the fact that these results were based on a very small benchmark suites.

As a last step, all of the algorithms were compared with one another and ranked based on the three performance measures. Table 16 summarizes the wins and losses, as well as the ranks per problem class with respect to solution accuracy. Based on the ranks over all of the functions, and considering the sum of wins and losses, the algorithms are ranked from best to worse in the order: **PSO-DX^u_y**, **PSO-NMPCO**, **PSO-DX^1_u**, **PSO-PSPG**, PSO-PCX$^*_r$, PSO-DX$^u_{\hat{y}}$, PSO-PCX$_r$, PSO-AX, gbest PSO, PSO-PCX$^*_{\hat{y}}$, PSO-PCX$_{\hat{y}}$, PSO-BLX; the bolded

Table 11 Comparison of wins and losses for adapted PSO-PCX strategies

PCX adaptation	Wins	Draws	Losses	Sum
PSO-PCX$_r$	61	0	0	61
PSO-PCX$^*_r$	29	19	13	16

Table 12 Comparison of wins and losses of crossover PSO algorithms against gbest PSO

Algorithm	Wins	Draws	Losses	Sum	Ranks for		
					Wins	Losses	Sum
PSO-AX	19	16	26	−7	8	5	6
PSO-BLX	0	0	61	−61	11	1	10
PSO-PSPG	32	18	11	21	4	8	4
PSO-NMPCO	36	20	5	31	2	11	2
PSO-PCX$_{\hat{y}}$	1	0	60	−59	10	2	9
PSO-PCX$^*_{\hat{y}}$	20	13	28	−8	7	4	7
PSO-PCX$^*_r$	29	15	17	27	5	7	3
PSO-PCX$_r$	10	15	36	−26	9	3	8
PSO-DX$^u_{\hat{y}}$	23	14	24	−1	6	6	5
PSO-DX^1_y	34	20	7	27	3	9	3
PSO-DX^u_y	38	17	6	32	1	10	1

algorithms are the only ones that obtained more wins than losses. The best performaing algorithm is the PSO that uses discrete crossover with a uniform recombindation with the personal best position. The worst performer is the PSO with blending crossover, achieving only 12 wins against the other 11 algorithms over the 61 problems.

Though PSO-DX^u_y ranked best over all of the functions, when considering the problem classes separately, the choice of best algorithm is very problem class dependent, as indicated in Table 13. PSO-DX^u_y performed best over all of the multimodal problems, over all of the separable problems, the noisy unimodal problems, and the separable multimodal problems. It performed second best over all of the non-separable problems and third best over all of the unimodal problems. The best performer over all of the unimodal problems was the PSO-PCX$^*_r$, and the best over all of the non-separable problems, the PSO-NMPCO. PSO-NMPCO also performed best in terms of the rank over the sum of wins and losses for the shifted unimodal, shifted multimodal, noisy multimodal and composition problems. The PSO-PCX$_r$ performed best for the separable unimodal problems, PSO-DX^u_y performed best for the non-separable unimodal and the separable multimodal problems, while PSO-PSPG performed best for the rotated unimodal, non-separable multimodal and rotated multimodal problems. Note that for both the shifted unimodal and the shifted multimodal problems all of the algorithms produced more losses than wins, indicating that the choice of best algorithms for these problem classes are much more problem dependent, and not just problem class dependent.

Considering success rate, Table 17 shows that PSO-DX^1_y ranks the best over all of the functions, and the over all of the unimodal, multimodal, separable and non-separable problem classes. This means that PSO-DX^1_y had most of its runs converging to the specified accuracy levels. The PSO-DX^u_y also ranked best over all of the separable problems, and ranked second best over all of the problems. Both these algorithms therefore provided the most accurate solutions and were the most stable. While the PSO-NMPCO performed well in terms of solution accuracy, it did not rank well in terms of success rate, which is an indication of large deviations on the mean solution accuracy. PSO-PSPG ranked third best over all of the problems, and as indicated above, was one of the only four algorithms with more wins than losses with respect to the accuracy measure.

Table 13 Best performing algorithms per problem class based on sum of wins and losses

Problem class	Solution accuracy	Success rate	Efficiency
Unimodal			
Separable	PCX_r	NMPCO	PCX_r^*
Non-separable	$DX_{\hat{y}}^u$	DX_y^1	BLX
Noisy	DX_y^u	NMPCO	PSPG
Shifted	NMPCO	DX_y^1	DX_y^1
Rotated	PSPG	DX_y^1	DX_y^1
Multimodal			
Separable	DX_y^u	DX_y^1	BLX
Non-separable	PSPG	DX_y^u	DX_y^u
Shifted	NMPCO	DX_y^u	DX_y^u
Rotated	PSPG	PSPG	NMPCO
Noisy	NMPCO	NMPCO	DX_y^1
Composition	NMPCO	NMPCO	BLX
Total			
Unimodal	PCX_r^*	DX_y^1	DX_y^1
Multimodal	DX_y^u	DX_y^1	BLX
Separable	DX_y^u	DX_y^1, DX_y^u	DX_y^u
Non-separable	NMPCO	DX_y^1	BLX
Overall	DX_y^u	DX_y^u	DX_y^u

Note that the PSO-$PCX_{\hat{y}}$ and PSO-$PCX_{\hat{y}}^*$ ranked as the worst with respect to success rate. Since these two algorithms had the most runs not converging to within the specified accuracy levels, they are the most unstable of the crossover PSO algorithms, exhibiting large deviations over the final obtained accuracy.

With reference to the different problem classes, PSO-DX_y^1 ranked best with respect to success rate for most of the problem classes, followed by PSO-NMPCO.

The wins, losses and ranks with respect to the efficiency measure are summarized in Table 18. Overall, and with respect to the sum of wins and losses, PSO-DX_y^u converged the fastest (i.e. on average the fewest iterations) to the specified accuracy levels. Despite the faster convergence compared to the other algorithms, PSO-DX_y^u also ranked best with respect to solution accuracy and second best with resepect to success rate. While PSO-PSPG, which ranked third best with respect to solution accuracy, obtained the most wins over all of the functions, it ranked eighth based on the sum of wins and losses due to a large number of losses. The algorithms that were slowest to converge are the PSO-AX and the gbest PSO. For both, the slower convergence did not provide any benefit in terms of solution accuracy as indicated in Table 16.

With respect to the different problem classes, PSO-DX_y^u converged the fastest over all of the separable problems, and the multimodal non-separable and multimodal shifted problems. The PSO-DX_y^1 was fastest over all of the unimodal problems, and the unimodal shifted, unimodal rotated, and the multimodal noisy problems. PSO-BLX was fastest for all of the non-seprable and all the multimodal problems, as well as the unimodal non-separable, multimodal separable, and composition problems. The PSO-PSPG was fastest for the unimodal

noisy problems, PSO-NMPCO for the multimodal rotated problems, and PSO-PCX$_r^*$ for the unimodal separable problems.

What is clear from Tables 16, 18 and 13, is that slower convergence, i.e. more exploration time, does not necessarily mean that better solutions will be found. With reference to the algorithms studied in this paper, it is very clearly illustrated that the algorithms that converged faster, also resulted in the most accurate solutions.

The preceeding discussions considered each of the performance measures separately. Table 14 provides for each of the algorithms a rank over all three of the performance measures as the average of the ranks obtained per performance measure. Average performance ranks are given over all of the benchmark functions and per the four main problem classes. PSO-DX$_y^u$ is indicated as the best performer ranked over all three performance measures, followed by PSO-DX$_y^1$. Only the PSO-PCX$_{\hat{y}}^*$ ranked worst than gbest PSO over all three performance measures, with PSO-PCX$_{\hat{y}}^*$ and PSO-PCX$_r$ ranking very close to that of gbest PSO.

Considering the different problem classes, PSO-DX$_y^u$ ranked best for the multimodal functions, the separable functions, and the non-separable functions. PSO-NMPCO obtained the same rank as PSO-DX$_y^u$ on the non-separable problems. For the unimodal problems, PSO-DX$_{\hat{y}}^u$ ranked the best, with PSO-DX$_y^u$ ranking third (though the difference in rank is only 0.667).

5 Conclusions

The main objectives of this paper were to provide an extensive review of PSO algorithms that make use of some form of crossover operator and to provide an empirical analysis of the performance of these algorithms. As a consequence of these objectives, it was found that current publications where these crossover PSO algorithms have been introduced lacked in a number of aspects:

- Their empirical performance have been analyzed on a very small set of benchmark functions.
- A sensitivity analysis of the control parameters introduced by the crossover operators were not done.
- Comparisons with other algorithms were not done using optimized control parameters.
- Comparisons did not include other crossover PSO algorithms.

This paper addressed all of these issues by using a large benchmark suite of 61 benchmark functions of different characteristics and complexity, a sensitivity analysis of all of the algorithms have been done and the best control parameter values per function were determined, all of the algorithms were compared with a base PSO algorithm to determine if the crossover PSO algorithms do in fact improve on the performance of the base PSO algorithm, and all of the crossover PSO algorithms were compared to determine which perform best.

In addition, adaptations of existing parent-centric crossover PSO algorithms have been proposed and shown to improve on the performance of the existing parent-centric PSO algorithms.

With reference to solution accuracy as performance measure, it was found that the performance of most of the algorithms is very sensitive to the crossover control parameter values. Using the best found control parameter values, only the PSO-PSPG, PSO-NMPCO, PSO-PCX$_r^*$, PSO-DX$_y^1$ and PSO-DX$_y^u$ obtained more wins than losses when compared to the gbest PSO.

Table 14 Ranks comparison over all performance measures with respect to sum of wins and losses

Problem class	Performance measure	Crossover operator used											
		gbest	AX	BLX-α	PSPG	NMPCO	PCX$_{\hat{y}}$	PCX$_r$	PCX$^*_{\hat{y}}$	PCX$^*_r$	DX$^u_{\hat{y}}$	DX^1_y	DX^u_y
Unimodal	Accuracy	8	10	12	6	5	11	4	9	1	2	7	3
	Success rate	10	7	8	5	2	12	9	11	6	4	1	3
	Efficiency	11	12	10	8	6	3	9	7	5	2	1	4
	Average rank	9.667	9.667	10.000	6.333	4.333	8.667	7.333	9.000	4.000	2.667	3.000	3.333
Multimodal	Accuracy	7	7	12	4	2	11	9	10	6	8	3	1
	Success rate	9	5	4	3	7	10	11	12	8	6	2	1
	Efficiency	9	8	1	10	3	5	12	11	7	6	4	2
	Average rank	8.333	6.667	5.667	5.667	4.000	8.667	10.667	11.000	7.000	6.667	3.000	1.333
Separable	Accuracy	8	9	12	7	6	11	4	10	3	5	2	1
	Success rate	7	7	11	5	6	12	10	9	4	3	2	1
	Efficiency	9	12	8	7	5	10	11	6	4	2	2	1
	Average rank	8.000	9.333	10.333	6.333	5.667	11.000	8.333	8.333	3.667	3.333	2.000	1.000
Non-separable	Accuracy	9	6	12	3	1	11	8	10	5	7	4	2
	Success rate	10	5	4	2	6	9	11	12	8	7	1	3
	Efficiency	12	11	1	9	2	3	8	10	7	6	5	4
	Average rank	10.333	7.333	5.667	4.667	3.000	7.667	9.000	10.667	6.667	6.667	3.333	3.000
Overall	Accuracy	9	8	12	4	2	11	7	10	5	6	2	1
	Success rate	9	5	4	3	6	11	10	12	8	7	1	2
	Efficiency	11	12	2	8	4	5	10	9	7	6	3	1
	Average rank	9.667	8.333	6.000	5.000	4.000	9.000	9.000	10.333	6.667	6.333	2.000	1.333

The crossover PSO algorithms were compared with respect to solution accuracy, success rate and efficiency. It was found that the best crossover PSO algorithm to use is very problem dependent, but that over all of the functions, PSO-DX$_y^u$ was shown to perform best with reference to average rank over all three performance measures.

Appendix: Benchmark functions and performance tables

This appendix provides detail on the functions included in the benchmark set and provides the detailed performance tables.

Table 15 summarizes the characteristics of these functions. A function, f_l, that is shifted, rotated, or shifted and rotated, is respectively referred to as f_l^{Sh}, f_l^R and f_l^{ShR}, where l is the index of the function.

A function, f_l, was shifted using

$$f_l^{Sh}(\mathbf{x}) = f_l(\mathbf{z}) + \beta$$

where $\mathbf{z} = \mathbf{x} - \gamma$; γ and β are constants. Table 15 lists the values by which each function was shifted. Two approaches to rotation were implemented: Where an entry in the rotation column of Table 15 indicates "ortho", the function f_l was rotated by a randomly generated orthonormal rotation matrix. The second approach, indicated by "linear" rotates the function using a linear transformation matrix. For both approaches the condition number is given in the table in parentheses, and rotation was done using Salomon's method Salomon (1996). A new rotation matrix was computed for each of the 50 independent runs of the algorithm. The rotated function, referred to as f_l^R, was computed by multiplying the decision vector $\mathbf{x}$ with the transpose of the rotation matrix.

Some functions were both shifted and rotated, with the resulting function referred to as f_l^{ShR}. Noisy functions were generated by multiplying each decision variable, x_j, by zero-mean noise sampled from a Gaussian distribution with deviation of one. These functions are referred to as f_l^N. Functions that are shifted and noisy are referred to as f_l^{ShN}.

Note that f_{27} is indicated as a non-separable function, even though it is separable near the optimum.

Rotations for functions f_{26}–f_{37} and function f_{15}^{ShRE} were by linear transformation matrix, with condition numbers that differ for each component function. Also the severity of shifts differ for the different component functions. For the detail of these parameters, the reader is referred to Suganthan et al. (2005). For these functions, the entry "yes" simply indicates if a transformation of the expanded or composition function was done. An entry of "CEC05" refers the reader to the definition of the function as in Suganthan et al. (2005).

The definitions of the benchmark functions used in this study is provided below together with detail on the domain of each function. Many of the functions below are equivalent to functions defined in the IEEE Congress on Evolutionary Computation (CEC) 2005 competition benchmark set Suganthan et al. (2005). Such equivalencies are indicated by giving the CEC 2005 benchmark number with the superscript CEC (Tables 16, 17, 18).

f_1, the absolute value function, defined as

$$f_1(\mathbf{x}) = \sum_{j=1}^{n_x} |x_i| \tag{7}$$

with each $x_j \in [-100, 100]$.

Table 15 Characteristics of benchmark functions

F	Eqs	M	S	Sh	R	ShR	N	E	C
f_1	7	Uni	Yes						
f_2	8	Multi	No	$\beta = -140$	Ortho (1)	$\beta = -140$			
						$\gamma = -32$			
				$\gamma = 10$		linear (100)			
f_3	9	Multi	Yes						
f_4	10	Multi	No						
f_5	11	Uni	Yes	$\beta = -450$	Ortho (1)	$\beta = -450$			
				$\gamma = 10$		$\gamma = 10$			
						Ortho (1)			
f_6	12	Multi	No	$\beta = -180$	Ortho (1)	$\beta = -180$			
				$\gamma = 10$		$\gamma = -60$			
						Linear (3)			
f_7	13	Uni	Yes						
f_8	14	Multi	Yes						
f_9	15	Multi	No						
f_{10}	16	Uni	No						
f_{11}	17	Uni	Yes				N(0,1)		
f_{12}	19	Multi	Yes	$\beta = -330$	Ortho (1)	$\beta = -330$			
				$\gamma = 2$		$\gamma = 1$			
						Linear (2)			
f_{13}	20	Multi	No	$\beta = 390$	Ortho (1)				
				$\gamma = 10$					
f_{14}	21	Multi	No						
f_{15}	22	Multi	No			$\beta = -300$		Yes	
						$\gamma = 20$			
						Linear (3)			
f_{16}	23	Uni	No	$\beta = -450$	Ortho (1)		$\lvert N(0, 0.4)$		
				$\gamma = 10$			$\beta = -450$		
							$\gamma = 10$		
f_{17}	25	Uni	No	$\beta = -310$					
f_{18}	26	Multi	No	$\beta = -460$					
f_{19}	27	Uni	Yes						
f_{20}	28	Uni	Yes						
f_{21}	29	Multi	No						
f_{22}	30	Uni	Yes	$\beta = -450$					
				$\gamma = 10$					
f_{23}	31	Multi	Yes						
f_{24}	32	Multi	Yes						
f_{25}	33	Multi	Yes			$\beta = 90$			
						$\gamma = 0.1$			
						Linear (5)			

Table 15 continued

F	Eqs	M	S	Sh	R	ShR	N	E	C
f_{26}	CEC05	Multi	No	$\beta = -130$ $\gamma = 1$				Yes	
f_{27}	CEC05	Multi	No						Yes
f_{28}	CEC05	Multi	No		Yes				Yes
f_{29}	CEC05	Multi	No				Yes		Yes
f_{30}	CEC05	Multi	No		Yes				Yes
f_{31}	CEC05	Multi	No		Yes				Yes
f_{32}	CEC05	Multi	No		Yes				Yes
f_{33}	CEC05	Multi	No		Yes				Yes
f_{34}	CEC05	Multi	No		Yes				Yes
f_{35}	CEC05	Multi	No		Yes				Yes
f_{36}	CEC05	Multi	No		Yes				Yes
f_{37}	CEC05	Multi	No		Yes				Yes

F function number, *Eq* equation number, *M* modality, *S* separability, *Sh* shifted, *R* rotated, *ShR* shifted and rotated, *N* noisy, *E* expanded function, *C* composition function

f_2, the ackley function, defined as

$$f_2(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n_x} x_j^2}} - e^{\frac{1}{n}\sum_{j=1}^{n_x}\cos(2\pi x_j)} + 20 + e \tag{8}$$

with each $x_j \in [-32.768, 32.768]$. Shifted, rotated, and rotated and shifted verions of the ackley function were used, respectively referred to as f_2^{Sh}, f_2^R and f_2^{ShR}. Function f_2^{ShR} is equivalent to function f_8^{CEC}.

f_3, the alpine function, defined as

$$f_3(\mathbf{x}) = \left(\prod_{j=1}^{n_x} \sin(x_j)\right)\sqrt{\prod_{j=1}^{n_x} x_j} \tag{9}$$

with each $x_j \in [-10, 10]$.

f_4, the egg holder function, defined as

$$f_4(\mathbf{x}) = \sum_{j=1}^{n_x-1}\left(-(x_{j+1}+47)\sin\left(\sqrt{|x_{j+1}+x_j/2+47|}\right) + \sin\left(\sqrt{|x_j-(x_{j+1}+47)|}\right)(-x_j)\right) \tag{10}$$

with each $x_j \in [-512, 512]$.

f_5, the elliptic function, defined as

$$f_5(\mathbf{x}) = \sum_{j=1}^{n_x}(10^6)^{\frac{j-1}{n_x-1}} \tag{11}$$

with each $x_j \in [-100, 100]$. Shifted, rotated, and rotated and shifted verions of the elliptic function were used, respectively referred to as f_5^{Sh}, f_5^R and f_5^{ShR}. Function f_5^{ShR} is equivalent to function f_3^{CEC}.

Table 16 Comparison of solution accuracy wins and losses for crossover PSO algorithms (results are given as wins/losses/sum of wins and losses)

Problem class	Crossover operator used											
	gbest	AX	BLX-α	PSPG	NMPCO	PCX$_{\hat{y}}$	PCX$_r$	PCX$^*_{\hat{y}}$	PCX$^*_r$	DX$^u_{\hat{y}}$	DX^1_y	DX^u_y
Unimodal												
Separable	28/49/−21	21/56/−35	11/66/−55	36/41/−5	40/37/3	14/63/−49	73/4/69	24/53/−29	63/14/49	57/20/37	38/39/−1	49/28/21
Non-separable	12/21/−9	13/20/−7	0/33/−33	22/11/11	25/8/17	3/30/−27	22/11/11	6/27/−21	20/13/7	27/6/21	18/15/3	18/15/3
Noisy	10/12/−2	6/16/−10	0/22/−22	11/11/0	16/6/10	2/20/−18	5/17/−12	8/14/−6	14/8/6	11/11/0	20/2/18	21/1/20
Shifted	17/38/−21	17/38/−21	1/54/−53	23/32/−9	26/29/−3	4/51/−47	12/43/−31	19/36/−17	20/35/−15	20/35/−15	23/32/−9	25/30/−5
Rotated	5/17/−12	5/17/−12	0/22/−22	18/4/14	16/6/10	2/20/−18	13/9/4	11/11/0	17/5/12	15/7/8	9/13/−4	13/9/4
Multimodal												
Separable	32/34/−2	21/45/−24	0/66/−66	36/30/6	37/29/8	5/61/−56	15/51/−36	14/52/−38	28/38/−10	21/45/−24	42/24/18	50/16/34
Non-separable	42/57/−15	34/65/−31	0/99/−99	72/27/45	51/48/3	9/90/−81	38/61/−23	28/71/−43	38/61/−23	41/58/−17	60/39/21	68/31/37
Shifted	36/85/−49	59/62/−3	0/121/−121	59/62/−3	60/61/−1	16/105/−89	46/75/−29	37/84/−47	50/71/−21	45/76/−31	45/76/−31	57/64/−7
Rotated	11/33/−22	22/22/0	0/44/−44	40/4/36	26/18/8	2/42/−40	17/27/−10	9/35/−26	17/27/−10	17/27/−10	17/27/−10	16/28/−12
Noisy	5/6/−1	5/6/−1	0/11/−11	3/8/−5	11/0/11	1/10/−9	2/9/−7	5/6/−1	5/6/−1	3/8/−5	9/2/7	9/2/7
Composition	44/88/−44	56/76/−20	0/132/−132	34/98/−64	85/47/38	12/120/−108	28/104/−76	33/99/−66	53/79/−26	33/99/−66	74/58/16	79/53/26
Total												
Unimodal	72/137/−65	62/147/−85	12/197/−185	110/99/11	123/86/37	25/184/−159	125/84/41	68/141/−73	134/75/59	130/79/51	108/101/7	126/83/43
Multimodal	170/303/−133	197/276/−79	0/473/−473	244/229/15	270/203/67	45/428/−383	146/327/−181	126/347/−221	191/282/−91	160/313/−153	247/226/21	279/194/85
Separable	74/113/−39	58/129/−71	12/175/−163	87/100/−13	94/93/1	22/165/−143	98/89/9	49/138/−89	106/81/25	97/90/7	107/80/27	125/62/63
Non-separable	163/321/−158	196/288/−92	0/484/−484	264/220/44	288/196/92	47/437/−390	171/313/−142	140/344/−204	214/270/−56	190/294/−104	239/245/−6	271/213/58
Overall	242/440/−198	259/423/−164	12/670/−658	354/328/26	393/289/104	70/612/−542	271/411/−140	194/488/−294	325/357/−32	290/392/−102	355/327/28	405/277/128

Table 16 continued

Problem class	Crossover operator used											
	gbest	AX	BLX-α	PSPG	NMPCO	PCX$_{\hat{y}}$	PCX$_r$	PCX$^*_{\hat{y}}$	PCX$^*_r$	DX$^u_{\hat{y}}$	DX^1_y	DX^u_y
Rank												
Unimodal	8/5/8	10/3/10	12/1/12	6/7/6	5/8/5	11/2/11	4/9/4	9/4/9	**1/12/1**	2/11/2	7/6/7	3/10/3
Multimodal	7/6/7	7/6/7	12/1/12	4/9/4	2/11/2	11/2/11	9/4/9	10/3/10	6/7/6	8/5/8	3/10/3	**1/12/1**
Separable	8/5/8	9/4/9	12/1/12	7/6/7	6/7/6	11/2/11	4/9/4	10/3/10	3/10/3	5/8/5	2/11/2	**1/12/1**
Non-separable	9/4/9	6/7/6	12/1/12	3/10/3	**1/12/1**	11/2/11	8/5/8	10/3/10	5/8/5	7/6/7	4/9/4	2/11/2
Overall	9/4/9	8/5/8	12/1/12	4/9/4	2/11/2	11/2/11	7/6/7	10/3/10	5/6/5	6/7/6	3/10/3	**1/12/1**

Best performing algorithms are given in bold

Table 17 Comparison of success rate wins and losses for crossover PSO algorithms (results are given as wins/losses/sum of wins and losses)

Problem class	Crossover operator used											
	gbest	AX	BLX-α	PSPG	NMPCO	PCX$_{\hat{y}}$	PCX$_r$	PCX$^*_{\hat{y}}$	PCX$^*_r$	DX$^u_{\hat{y}}$	DX^l_y	DX^u_y
Unimodal												
Separable	33/36/−3	30/41/−11	25/45/−20	37/36/1	51/19/32	17/50/−33	62/52/10	40/29/11	42/26/16	35/25/10	35/25/10	44/17/27
Non-separable	8/20/−12	16/12/4	20/8/12	19/13/6	15/11/4	12/16/−4	4/25/−21	9/17/−8	3/23/−20	20/9/11	25/6/19	18/9/9
Noisy	5/9/−4	8/10/−2	4/13/−9	10/6/4	21/0/21	11/9/2	7/12/−5	7/9/−2	16/4/12	2/15/−13	15/6/9	4/17/−13
Shifted	9/23/−14	14/23/−9	18/24/−6	21/12/9	16/14/2	13/31/−18	11/20/−9	5/28/−23	15/18/−3	20/15/14	44/1/43	29/15/14
Rotated	8/9/−1	14/7/7	6/14/−8	12/9/3	5/11/−6	12/6/6	6/14/−8	0/19/−19	13/5/8	12/9/3	15/1/14	11/9/2
Multimodal												
Separable	27/28/−1	30/21/9	27/29/−2	33/23/10	15/36/−21	9/42/−33	20/32/−12	7/42/−35/	38/19/19	33/20/13	45/16/29	39/15/24
Non-separable	22/40/−18	47/26/21	47/24/23	56/21/35	18/42/−24	30/38/−8	8/52/−44	11/46/−35	20/43/−23	34/31/3	46/28/18	66/14/52
Shifted	43/37/6	43/33/10	30/40/−10	55/29/26	16/55/−39	26/46/−20	32/41/−9	20/52/−32	32/41/−9	48/40/8	51/38/13	70/20/50
Rotated	20/12/8	12/17/−5	25/12/13	36/7/29	10/18/−8	10/25/−15	0/28/−28	7/21/−14	11/22/−11	18/10/8	26/7/19	15/11/4
Noisy	4/2/2	4/2/2	4/2/2	0/8/−8	11/0/11	4/3/1	0/8/−8	5/3/2	0/8/−8	0/8/−8	10/1/9	5/2/3
Composition	23/54/−31	42/35/7	71/29/42	24/47/−13	75/22/53	38/50/−12/	52/40/12	55/42/13	43/41/2	27/66/−39	48/52/−4	27/57/−30
Total												
Unimodal	63/97/−34	82/93/−11	73/104/−31	99/76/23/	108/55/53	65/112/−47	90/123/−33	61/102/−41	89/76/13	98/73/25	134/39/95	106/67/39
Multimodal	139/173/−34	178/134/44	204/136/68	214/135/79	145/173/−28	117/204/−87/	112/201/−89	105/206/−101	144/174/−30	160/175/−15	226/142/84	222/119/103
Separable	75/77/−2	76/78/−2	63/94/−31	90/69/21	79/69/10	32/124/−92	82/110/−28	64/84/−20	91/64/27	89/60/29	113/47/66	113/41/72
Non-separable	123/191/−68	180/147/33	210/144/66	223/134/89	163/159/4	146/189/−43	120/206/−86	97/221/−124	142/178/−36	169/180/−11	237/133/104	210/143/67
Overall	202/270/−68	260/227/33	277/240/37	313/211/102	253/228/25	182/316/−134/	202/324/−122	166/308/−142	233/250/−17	258/248/10	360/181/179	328/186/142

Table 17 continued

Problem class	Crossover operator used											
	gbest	AX	BLX-α	PSPG	NMPCO	PCX$_{\hat{y}}$	PCX$_r$	PCX$^*_{\hat{y}}$	PCX$^*_r$	DX$^u_{\hat{y}}$	DX^l_y	DX^u_y
Rank												
Unimodal	11/5/10	8/6/7	9/3/8	4/7/5	2/11/2	10/2/12	6/1/9	12/4/11	7/7/6	5/9/4	**1/12/1**	3/10/3
Multimodal	9/6/9	5/11/5	4/9/4	3/10/3	7/6/7	10/2/10	11/3/11	12/1/12	8/5/8	6/4/6	**1/8/2**	2/12/1
Separable	9/6/7	8/5/7	11/3/11	4/7/5	7/7/6	12/1/12	6/2/10	10/4/9	3/9/4	5/10/3	**1/11/2**	**1/12/1**
Non-separable	10/3/10	5/8/5	3/9/4	2/11/2	7/7/6	8/4/9	11/2/11	12/1/12	9/6/8	6/5/7	**1/12/1**	3/10/3
Overall	9/4/9	5/9/5	4/7/4	3/10/3	7/8/6	11/2/11	9/1/10	12/3/12	8/5/8	6/6/7	**1/12/1**	2/11/2

Best performing algorithms are given in bold

Table 18 Comparison of efficiency wins and losses for crossover PSO algorithms (results are given as wins/losses/sum of wins and losses)

Problem class	Crossover operator used											
	gbest	AX	BLX-α	PSPG	NMPCO	PCX$_{\hat{y}}$	PCX$_r$	PCX$^*_{\hat{y}}$	PCX$^*_r$	DX$^u_{\hat{y}}$	DX^1_y	DX^u_y
Unimodal												
Separable	22/45/−23	15/54/−39	26/47/−21	39/34/5	38/35/3	28/43/−15	28/43/−15	50/25/25	52/24/28	44/24/20	38/28/10	43/21/22
Non-separable	2/21/−19	7/22/−15	26/7/19	8/18/−10	13/13/0	21/12/9	12/16/−4	12/16/−4	9/16/−7	23/8/15	18/13/5	19/8/11
Noisy	5/9/−4	6/11/−5	0/17/−17	19/2/17	9/6/3	16/5/11	12/10/2	9/7/2	18/4/14	2/15/−13	11/9/2	4/16/−12
Shifted	10/25/−15	0/28/−28	17/24/−7	5/26/−21	20/17/3	35/16/19	21/17/4	5/26/−12	22/18/4	36/13/23	39/13/26	30/17/13
Rotated	9/10/−1	6/13/−7	1/17/−16	8/9/−1	14/5/9	18/4/14	9/10/−1	5/11/−6	3/12/−9	12/9/3	18/3/15	10/10/0
Multimodal												
Separable	27/26/1	21/33/−12	45/20/25	21/30/−9	28/26/2	18/36/−18	17/32/−15	7/36/−29	31/24/7	35/19/16	35/24/11	37/16/21
Non-separable	26/36/−10	35/39/−4	60/25/35	19/40/−21	57/20/37	39/33/6	15/43/−28	9/45/−36	13/40/−27	31/33/−2	42/31/11	60/21/39
Shifted	35/45/−10	30/47/−17	33/43/−10	19/47/−28	59/33/26	49/34/15	31/41/−10	35/38/−3	28/43/−15	44/39/5	43/39/4	65/22/43
Rotated	14/16/−2	7/19/−12	27/10/17	1/23/−22	30/10/20	12/16/−4	5/19/−14	6/19/−13	12/18/−6	19/8/11	25/8/17	17/9/8
Noisy	7/4/3	8/3/5	10/1/9	0/10/−10	1/7/−6	6/5/1	1/7/−6	5/6/−1	1/7/−6	0/7/−7	11/0/11	9/2/7
Composition	34/57/−23	51/49/2	98/26/72	67/36/31	40/57/−17	52/53/−1	44/52/−8	67/52/15	60/39/21	18/64/−46	38/59/−21	31/56/−25
Total												
Unimodal	48/110/−62	34/128/−94	70/112/−42	79/89/−10	94/76/18	118/80/38	82/96/14	81/85/−4	104/74/30	117/69/48	124/66/58	106/72/34
Multimodal	143/184/−41	152/190/−38	273/125/148	127/186/−59	215/153/62	113/194/−81	129/196/−67	145/171/−26	147/170/−23	194/161/33	219/126/93	
Separable	60/89/−29	42/105/−63	79/90/−11	70/80/−10	83/74/9	63/98/−35	50/95/−45	73/76/−3	98/61/37	103/59/44	105/61/44	110/48/62
Non-separable	124/201/−77	136/210/−74	254/146/108	136/185/−49	225/148/77	225/154/71	144/188/−44	132/199/−67	150/177/−27	161/173/−12	202/166/36	206/148/58
Overall	191/294/−103	186/318/−132	343/237/106	206/275/−69	309/229/80	294/257/37	195/290/−95	210/281/−71	249/245/4	264/239/25	318/227/91	325/198/127

Table 18 continued

Problem class	Crossover operator used											
	gbest	AX	BLX-α	PSPG	NMPCO	PCX$_{\hat{y}}$	PCX$_r$	PCX$^*_{\hat{y}}$	PCX$^*_r$	DX$^u_{\hat{y}}$	DX^1_y	DX^u_y
Rank												
Unimodal	11/3/11	12/1/12	10/2/10	9/5/8	6/8/6	2/7/3	7/4/9	8/6/7	5/9/5	3/11/2	**1/12/1**	4/10/4
Multimodal	9/5/9	6/3/8	**1/12/1**	11/4/10	3/10/3	5/6/5	12/2/12	10/1/11	8/7/7	7/8/6	4/9/4	2/11/2
Separable	10/5/9	12/1/12	6/4/8	8/6/7	5/8/5	9/2/10	11/3/11	7/7/6	4/9/4	3/11/2	2/9/2	**1/12/1**
Non-separable	12/2/12	9/1/11	**1/12/1**	1/5/9	2/10/2	2/9/3	8/4/8	11/3/10	7/6/7	6/7/6	5/8/5	4/10/4
Overall	11/2/11	12/1/12	1/9/2	1/5/8	4/10/4	5/6/5	10/3/10	8/4/9	7/7/7	6/8/6	3/11/3	**2/12/1**

Best performing algorithms are given in bold

f_6, the griewank function, defined as

$$f_6(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{j=1}^{n_x} x_j^2 - \prod_{j=1}^{n_x} \cos\left(\frac{x_j}{\sqrt{j}}\right) \tag{12}$$

with each $x_j \in [-600, 600]$. Shifted, rotated, and rotated and shifted verions of the elliptic function were used, respectively referred to as f_6^{Sh}, f_6^{R} and f_6^{ShR}. For function f_6^{ShR}, each $x_j \in [0, 600]$, which means that the global minimum is outside of the bounds. Function f_6^{ShR} is equivalent to function f_7^{CEC}. with bounds as given above.

f_7, the hyperellipsoid function, defined as

$$f_7(\mathbf{x}) = \sum_{j=1}^{n_x} j x_j^2 \tag{13}$$

with each $x_j \in [-5.12, 5.12]$.

f_8, the michalewicz function, defined as

$$f_8(\mathbf{x}) = -\sum_{j=1}^{n_x} \sin(x_j)\left(\sin\left(\frac{j x_j^2}{\pi}\right)\right)^{2m} \tag{14}$$

with each $x_j \in [0, \pi]$ and $m = 10$.

f_9, the norwegian function, defined as

$$f_9(\mathbf{x}) = \prod_{j=1}^{n_x}\left(\cos\left(\pi x_j^3\right)\left(\frac{99 + x_j}{100}\right)\right) \tag{15}$$

with each $x_j \in [-1.1, 1.1]$.

f_{10}, the quadric function, defined as

$$f_{10}(\mathbf{x}) = \sum_{i=1}^{n_x}\left(\sum_{j=1}^{i} x_j\right)^2 \tag{16}$$

with each $x_j \in [-100, 100]$.

f_{11}, the quartic function, defined as

$$f_{11}(\mathbf{x}) = \sum_{j=1}^{n_x} j x_j^4 \tag{17}$$

with each $x_j \in [-1.28, 1.28]$. A noisy version of the quartic function, referred to as de jong's f4 function, was generated as follows:

$$f_{11}^N(\mathbf{x}) = \sum_{j=1}^{n_x}\left(j x_j^4 + N(0, 1)\right) \tag{18}$$

The domain was the same as that of the quartic function.

f_{12}, the rastrigin function, defined as

$$f_{12}(\mathbf{x}) = 10 n_x + \sum_{j=1}^{n_x}\left(x_j^2 - 10\cos(2\pi x_j)\right) \tag{19}$$

with $x_j \in [-5.12, 5.12]$. Shifted, rotated, and rotated and shifted verions of the rastrigin function were used, respectively referred to as f_{12}^{Sh}, f_{12}^{R} and f_{12}^{ShR}. Function f_{12}^{ShR} is equivalent to function f_{10}^{CEC}.

f_{13}, the rosenbrock function, defined as

$$f_{13}(\mathbf{x}) = \sum_{j=1}^{n_x-1} \left(100\left(x_{j+1} - x_j^2\right)^2 + (x_j - 1)^2\right) \tag{20}$$

with $x_j \in [-30, 30]$. Shifted, rotated, and rotated and shifted verions of the rosenbrock function were used, respectively referred to as f_{13}^{Sh}, f_{13}^{R} and f_{13}^{ShR}. Function f_{13}^{Sh}'s domain was $[-100, 100]$ to be equivalent to f_6^{CEC}. Function f_{13}^{R}'s domain was also set to $[-100, 100]$. f_{10}^{CEC}.

f_{14}, the salomon function, defined as

$$f_{14}(\mathbf{x}) = -\cos\left(2\pi \sum_{j=1}^{n_x} x_j^2\right) + 0.1\sqrt{\sum_{j=1}^{n_x} x_j^2} + 1 \tag{21}$$

with $x_j \in [-100, 100]$.

f_{15}, the schaffer 6 function, defined as

$$f_{15}(\mathbf{x}) = \sum_{j=1}^{n_x-1} \left(0.5 + \frac{sin^2\left(x_j^2 + x_{j+1}^2\right) - 0.5}{\left(1 + 0.001\left(x_j^2 + x_{j+1}^2\right)\right)^2}\right) \tag{22}$$

with each $x_j \in [-100, 100]$. A shifted and rotated, expanded version of the schaffer 6 function was used, referred to as f_{15}^{ShRE}, which is equivalent to f_{14}^{CEC}.

f_{16}, the schwefel 1.2 function, defined as

$$f_{16}(\mathbf{x}) = \sum_{j=1}^{n_x} \left(\sum_{k=1}^{j} x_k\right)^2 \tag{23}$$

with $x_j \in [-100, 100]$. Shifted, rotated, and noisy shifted verions of the schwefel 1.2 function were used, respectively referred to as f_{16}^{Sh}, f_{16}^{R} and f_{16}^{ShN}. Function f_{16}^{Sh} is equivalent to f_2^{CEC}, and f_{16}^{ShN} is equivalent to f_4^{CEC}, defined as

$$f_{16}(\mathbf{x}) = \sum_{j=1}^{n_x} \left(\sum_{k=1}^{j} x_k\right)^2 (1 + 0.4|N(0, 1)|) \tag{24}$$

f_{17}, the schwefel 2.6 function, defined as

$$f_{17}(\mathbf{x}) = \max_j\{|\mathbf{A}_j\mathbf{x} - \mathbf{B}_j|\} \tag{25}$$

with $x_j \in [-100, 100]$, $a_{ji} \in \mathbf{A}$ is uniformly sampled from $U(-500, 500)$ such that $\det(\mathbf{A}) \neq 0$, and each $\mathbf{B}_j = \mathbf{A}_j\mathbf{x}$. Function f_{17} is equivalent to f_5^{CEC}. A shifted version of schwefel 2.6 was also implemented, referred to as f_{17}^{Sh}.

f_{18}, the schwefel 2.13 function, defined as

$$f_{18}(\mathbf{x}) = \sum_{j=1}^{n_x} (\mathbf{A}_j - \mathbf{B}_j(\mathbf{x}))^2 \tag{26}$$

with each $x_j \in [-\pi, \pi]$, and

$$\mathbf{A}_j = \sum_{i=1}^{n_x} (a_{ji} \sin \alpha_i + b_{ji} \cos \alpha_i)$$
$$\mathbf{B}_j(\mathbf{x}) = \sum_{i=1}^{n_x} (a_{ji} \sin x_i + b_{ji} \cos x_i$$

where $a_{ji} \in \mathbf{A}$ and $b_{ji} \in \mathbf{B}$ with $a_{ji}, b_{ji} \sim U(-100, 100)$, and $\alpha_i \sim U(-\pi, \pi)$. This function is equivalent to f_{12}^{CEC}. A shifted version of schwefel 2.13 was also implemented, referred to as f_{18}^{Sh}.

f_{19}, the schwefel 2.21 function, defined as

$$f_{19}(\mathbf{x}) = \max_j \{|x_j|, 1 \leq j \leq n_x\} \tag{27}$$

with each $x_j \in [-100, 100]$.

f_{20}, the schwefel 2.22 function, defined as

$$f_{20}(\mathbf{x}) = \sum_{j=1}^{n_x} |x_j| + \prod_{j=1}^{n_x} |x_j| \tag{28}$$

with each $x_j \in [-10, 10]$.

f_{21}, the shubert function, defined as

$$f_{21}(\mathbf{x}) = \prod_{j=1}^{n_x} \left(\sum_{i=1}^{5} (i \cos((i+1)x_j + i) \right) \tag{29}$$

with each $x_j \in [-10, 10]$.

f_{22}, the spherical function, defined as

$$f_{22}(\mathbf{x}) = \sum_{j=1}^{n_x} x_i^2 \tag{30}$$

with each $x_j \in [-5.12, 5.12]$. A shifted version was implemented, referred to as f_{22}^{Sh} and equivalent to f_1^{CEC}.

f_{23}, the step function, defined as

$$f_{23}(\mathbf{x}) = \sum_{j=1}^{n_x} \left(\lfloor x_j + 0.5 \rfloor \right)^2 \tag{31}$$

with each $x_j \in [-100, 100]$.

f_{24}, the vincent function, defined as

$$f_{24}(\mathbf{x}) = -\left(1 + \sum_{j=1}^{n_x} \sin(10\sqrt{x_j}) \right) \tag{32}$$

with each $x_j \in [0.25, 10]$.

f_{25}, the weierstrass function, defined as

$$f_{25}(\mathbf{x}) = \sum_{j=1}^{n_x}\left(\sum_{i}^{20}(a^i \cos(2\pi b^i (x_j + 0.5)))\right) - n_x \sum_{i=1}^{20}(a^i \cos(\pi b^i)) \quad (33)$$

with each $x_j \in [-0.5, 0.5]$, $a = 0.5$ and $b = 3$. A shifted and rotated version of the weierstrass function was implemented, referred to as f_{25}^{ShR}, equivalent to f_{11}^{CEC}.

f_{26}, a shifted expansion of the griewank and rosenbrock functions (f_6 and f_{13} respectively), equivalent to f_{13}^{CEC}. Each $x_j \in [-3, 1]$.

$f_{27} - f_{37}$, which are all composition functions, respectively equivalent to f_{15}^{CEC} to f_{25}^{CEC}. All functions have $x_j \in [-5, 5]$, except f_{37} for which $x_j \in [2, 5]$.

References

Abdelbar A, Abdelshahid S (2003) Swarm optimization with instinct-driven particles. In: Proceedings of the IEEE congress on evolutionary computation. IEEE, Piscataway, pp 777–782

Cleghorn C, Engelbrecht A (2014) A generalized theoretical deterministic particle swarm model. Swarm Intell 8(1):35–59

Clerc M, Kennedy J (2002) The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6(1):58–73

Deb K, Padhye N (2010) Development of efficient particle swarm optimizers by using concepts from evolutionary algorithms. In: Proceedings of the 12th annual conference on genetic and evolutionary computation. ACM, New York, pp 55–62

Deb K, Joshi D, Anand A (2002) Real-coded evolutionary algorithms with parent-centric recombination. In: Proceedings of the IEEE congress on evolutionary computation. IEEE, Piscataway, pp 61–66

Dong Y, Yang H (2010) A new approach for reactive power/voltage optimization control of regional grid. In: Proceedings of the Asia-Pacific power & energy conference, pp 1–5

Duong S, Konjo H, Uezato E, Yamamoto T (2010) Particle swarm optimization with genetic recombination: a hybrid evolutionary algorithm. Artif Life Robot 15(4):444–449

Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the IEEE congress on evolutionary computation, vol 1. IEEE, Piscataway, pp 84–88

Engelbrecht A (2005) Fundamentals of computational swarm intelligence. Wiley, Chichester

Engelbrecht A (2012) Particle swarm optimization: velocity initialization. In: Proceedings of the IEEE congress on evolutionary computation

Engelbrecht A (2013a) Particle swarm optimization: global best or local best?. In: Proceedings of the first BRICS countries congress on computational intelligence

Engelbrecht A (2013b) Particle swarm optimization with discrete crossover. In: Proceedings of the IEEE congress on evolutionary computation

Engelbrecht A (2013c) Roaming behavior of unconstrained particles. In: Proceedings of the first BRICS countries congress on computational intelligence

Engelbrecht A (2014) Asynchronous particle swarm optimization with discrete crossover. In: Proceedings of the IEEE swarm intelligence symposium

Eshelman L, Schaffer J (1993) Real-coded genetic algorithms and interval schemata. In: Whitley D (ed) Foundations of genetic algorithms, vol 2. Morgan Kaufmann, San Mateo, pp 187–202

Garden R, Engelbrecht A (2014) Analysis and classification of function optimisation benchmark function and benchmark suites. In: Proceedings of the IEEE congress on evolutionary computation, IEEE

Higashi H, Iba H (2003) Particle swarm optimization with gaussian mutation. In: Proceedings of the IEEE swarm intelligence symposium. IEEE, Piscataway, pp 72–79

Kennedy J (1999) Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: Proceedings of the IEEE congress on evolutionary computation, vol 3. IEEE, Piscataway, pp 1931–1938

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the IEEE international joint conference on neural networks. IEEE, Piscataway, pp 1942–1948

Kennedy J, Mendes R (2002) Population structure and particle performance. In: Proceedings of the IEEE congress on evolutionary computation. IEEE, Piscataway, pp 1671–1676

Løvberg M, Rasmussen T, Krink T (2001) Hybrid particle swarm optimiser with breeding and subpopulations. In: Proceedings of the genetic and evolutionary computation conference. Morgan Kaufmann, San Fransisco, pp 469–476

Miranda V, Fonseca N (2002) EPSO-best-of-two-worlds meta-heuristic applied to power system problems. In: Proceedings of the IEEE congress on evolutionary computation, vol 2. IEEE, Piscataway, pp 1080–1085

Ono I, Kobayashi S (1997) A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover. In: Proceedings of the seventh international conference on genetic algorithms. Morgan Kaufmann, San Fransisco, pp 246–253

Padhye N (2010) Development of efficient particle swarm optimizers and bound handling methods. Master's thesis, Indian Institute of Technology

Park JB, Jeong YW, Shin JR, Lee K (2010) An improved particle swarm optimization for nonconvex economic dispatch problems. IEEE Trans Power Syst 25(1):156–166

Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. Swarm Intell 1(1):33–57

Salomon R (1996) Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions. BioSystems 39:263–278

Sedighizadeh D, Masehian E (2009) Particle swarm optimization methods, taxonomy and applications. Int J Comput Theory Eng 1(5):486–502

Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: Proceedings of the IEEE congress on evolutionary computation. IEEE, pp 69–73

Stacey A, Jancic M, Grundy I (2003) Particle swarm optimization with mutation. In: Proceedings of the IEEE congress on evolutionary computation, vol 2. IEEE, Piscataway, pp 1425–1430

Suganthan P, Hansen N, Liang J, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report, Nanyang Technological University, Singapore

Trelea I (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf Process Lett 85(6):317–325

van den Bergh F, Engelbrecht A (2002) A new locally convergent particle swarm optimizer. In: Proceedings of the IEEE international conference on systems, man, and cybernetics. IEEE, Piscataway, pp 96–101

van den Bergh F, Engelbrecht A (2006) A study of particle swarm optimization particle trajectories. Inf Sci 176(8):937–971

Wang W, Wu Z, Liu Y, Zeng S (2008) Particle swarm optimization with a novel multi-parent crossover operator. In: Proceedings of the fourth international conference on natural computation. IEEE, Piscataway, pp 664–668

Wei C, He Z, Zheng Y, Pi W (2002) Swarm directions embedded in fast evolutionary programming. In: Proceedings of the IEEE congress on evolutionary computation, vol 2. IEEE, Piscataway, pp 1278–1283

Worasucheep C, Pipopwatthana C, Srimontha S, Phanmak W (2012) Enhanced performance of particle swarm optimization with generalized generation gap model with parent-centric recombination operator. ECTI Trans Electr Eng Electron Commun 6(2):166–175