TÜBİTAK

Turk J Elec Eng & Comp Sci, Vol.18, No.6, 2010, © TÜBİTAK
doi:10.3906/elk-0909-18

# StPSO: Strengthened particle swarm optimization

**Aişe Zülal ŞEVKLİ[1], Fatih Erdoğan SEVİLGEN[2]**
[1] *Fatih University, Faculty of Engineering, Department of Computer Engineering, Büyüçekmece, İstanbul, 34500, TURKEY*
*e-mail: zsevkli@fatih.edu.tr*
[2] *Gebze Institue of Technology, Faculty of Engineering, Department of Computer Engineering, Gebze, Kocaeli-TURKEY*
*e-mail: sevilgen@bilmuh.gyte.edu.tr*

**Abstract**

*In this paper, we present a novel approach to strengthen Particle Swarm Optimization (PSO). PSO is a population-based metaheuristic that takes advantage of individual memory and social cooperation in a swarm. It has been applied to a variety of optimization problems because of its simplicity and fast convergence. However, straightforward application of PSO suffers from premature convergence and lack of intensification around the local best locations. To rectify these problems, we modify update procedure for the best particle in the swarm and propose a simple and random moving strategy. We perform a Reduced Variable Neighborhood Search (RVNS) based local search around the particle, as well. The resulting strengthened PSO (StPSO) algorithm not only has superior exploration and exploitation mechanisms but also provides a dynamical balance between them. Experimental analysis of StPSO is performed on continuous function optimization problems and a discrete problem, Orienteering Problem. Its performance is quite robust and consistent for all problem types; discrete or continuous, unimodal or multimodal. StPSO either reproduces the best known solution or provides a competitive solution for each problem instance. So, it is a valuable tool producing promising solutions for all problem types.*

**Key Words:** *Particle Swarm Optimization, Reduced Variable Neighborhood Search, continuous function optimization, Orienteering Problem, premature convergence, local search.*

## 1. Introduction

Metaheuristics are general frameworks commonly used for solving hard optimization problems. To examine a search space systematically, they employ intelligent and powerful heuristic mechanisms for both exploration and exploitation. Amongst several metaheuristics using various search philosophies, ones providing dynamic balance between exploration and exploitation, usually deliver better performance. If exploration is the governing feature of a metaheuristic, it may perform a vague examination on the search space and generally misses good solutions. On the contrary, if exploitation is dominant, the entire search space may not be explored and premature convergence to a local optimum is possible.

Particle Swarm Optimization (PSO) is a metaheuristic introduced by Kennedy and Eberhart [1]. PSO is inspired of the behaviors of social models like bird flocking or fish schooling and is based on individual improvement and social cooperation. In PSO, a population of particles (swarm) flies over a multidimensional search space representing candidate solutions. During their search journey determined by their current tendency, personal experience and swarm's experience, particles are expected to explore the whole search space and to tend towards the optimal solution.

PSO is initially proposed for continuous nonlinear optimization problems. Later it becomes quite popular and is applied to a wide range of optimization problems due to rapid swarm convergence and simplicity in its conceptual features and implementation [2][3]. Although rapid convergence is a desirable advantage, it may result in a severe premature convergence problem, as well [4][5]. The swarm may converge to a local solution and may not escape from it to explore the search space thoroughly. Moreover, PSO has a shortcoming in finding the best solution around a local neighborhood [6][7].

Several modifications on the standard PSO algorithm are proposed to rectify abovementioned drawbacks. Most of these modifications concern only on either exploration or exploitation mechanism and merely correct the deficiencies partially. They locate good solutions for some of the problem types but fail to produce competitive results for others. For instance, while the methods concentrated on exploration mechanism improve the solution quality for multimodal continuous function optimization problems, they typically generate inferior solutions for unimodal problems.

In this paper, a Strengthened PSO (StPSO) algorithm with improved exploration and exploitation mechanisms is proposed. In StPSO, the general flow of the PSO algorithm is preserved. However, among the PSO iterations, the search strategy is temporarily modified only for the particle (the pioneering-particle) which achieves or enhances the best solution. For the pioneering-particles, Reduced Variable Neighborhood Search (RVNS) is employed. The solutions obtained by RVNS do not affect swarm's experience in order to find promising search areas afterwards and not to hinder exploration. To improve exploration further and to prevent premature convergence, the pioneering-particles continue their search journey with random velocities and their previous experience.

The primary advantage of StPSO is its applicability to many different types of optimization problems with satisfactory performance. To support this claim, we experimented StPSO on both continuous and discrete problems. Results for the continuous problems indicate that our approach improves the search capability of the standard PSO algorithm. StPSO produces comparable solutions for both unimodal and multimodal functions while other similar PSO variants deliver poor solutions for either unimodal or multimodal functions. Moreover, our approach achieves the best known solutions for almost all instances of the Orienteering Problem (OP) which is a discrete NP-hard problem.

Besides proposing a promising and valuable tool for solving different types of problems (continuous or discrete / unimodal or multimodal) satisfactorily, there are two auxiliary contributions of this paper; Firstly, we present the first successful PSO based algorithm for OP (preliminary results are presented in [8]). IS-PSO which is derived from StPSO, reproduces the best known solutions for all benchmark problems. Secondly, we propose that RVNS with dynamic neighborhoods is a good alternative for a subsidiary local search.

The remainder of this paper is organized in seven sections: Sections 2 and 3 explains the standard PSO algorithm and its variants, respectively. Section 4 gives details of StPSO. Experimental problem sets and results are given in Sections 5 and 6, respectively. Finally, Section 7 contains the discussion and concluding remarks.

## 2. Particle swarm optimization

Particle Swarm Optimization (PSO) is a population-based method in which a swarm includes $n$ individuals called particles. Each particle has a $d$-dimensional position vector representing a candidate solution and a $d$-dimensional velocity vector expressing the current tendency of the particle during its search journey. Initial swarm can be constructed randomly or by using some predetermined values. At each step, the velocity of each particle is re-evaluated based on the particle's inertia as well as the social interaction (swarm's experience) and personal experience of the particle. The experience of each particle is usually captured by its local best position (*pbest*). The experience of the swarm is captured by the global best position (*gbest*). In the course of several iterations, particles make use of this experience and are supposed to move towards the optimum position.

```
 1 Procedure PSO
 2  Initialize Parameters
 3  Initialize Population
 4  For each particle
 5   Evaluate
 6   Update local best
 7  Update global best
 8  Do
 9   For each particle
10     Update velocity and position
11     Evaluate
12     Update local best
13   Update global best
14  While (Not Terminated)
15 End Procedure
```

**Figure 1**. The pseudo-code of the standard PSO algorithm.

Pseudo-code of the standard PSO algorithm is illustrated in Figure 1. Optimization is achieved in the course of several iterations of update-evaluate steps. During the update step (line 10) at iteration $t$, the velocity and the position vector of each particle are calculated by using the equations (1) and (2). In these equations, $v_{i,j}^t$ and $p_{i,j}^t$ are the velocity and the position values of the $j^{th}$ dimension ($1 \leq j \leq d$) of the $i^{th}$ particle ($1 \leq i \leq n$) at iteration $t$, respectively. The parameters $c_1$ and $c_2$ are coefficients of learning factors, which are the weights for contributions of personal experience and social interaction. The stochastic behavior of PSO is achieved by random numbers $rand_1$ and $rand_2$ which are positive numbers generally uniformly distributed in [0, 1].

$$v_{i,j}^t = v_{i,j}^{t-1} + c_1 rand_1 (pbest_{i,j}^{t-1} - p_{i,j}^{t-1}) + c_2 rand_2 (gbest_j^{t-1} - p_{i,j}^{t-1}) \tag{1}$$

$$p_{i,j}^t = p_{i,j}^{t-1} + v_{i,j}^{t-1} \tag{2}$$

After the update step, the fitness function value is calculated (line 11) for each particle based on its position (the candidate solution represented by the particle.) The local best position, *pbest* of each particle (line 12) and the global best position, *gbest* of the swarm (line 13) are updated if the candidate solution is better than

*pbest* or *gbest*, respectively. The stopping condition (line 14) of the update-evaluate iterations is usually the attainment of a maximum number of iterations or a maximum number of iterations between two improvements.

# 3. Related works on PSO

Since PSO is introduced, it has been adapted to solve several optimization problems. Meanwhile some shortcomings in the standard PSO algorithm are observed for various problem domains and/or objective functions. For example, for multimodal problems, the PSO algorithm is generally not able to perform exploration satisfactorily because of premature convergence [4][9]. Solutions obtained using the standard PSO algorithm are not satisfactory for unimodal problems where exploitation is important, as well. Since PSO is not guaranteed to converge to local optima [7], solutions are usually improved by using fine tuned local search methods [6][10][11][12]. So, neither exploration nor exploitation mechanism in the standard PSO algorithm is adequate for different problem types.

To improve search efficiency and rectify the deficiencies in the standard algorithm, researches proposed several modifications to PSO. Some of these modifications are classified in Table 1. These modifications typically attack to either exploration or exploitation weakness and applied to PSO internally or externally. Internal modifications are moderate adjustments which do not change the heuristic approach in PSO but strengthen them by introducing new components and/or parameters. We call major modifications which introduce new heuristic methods in PSO as external. The heuristic methods are usually barrowed from other metaheuristics that makes the algorithm hybrid.

**Table 1.** Summary of the literature.

| | **Exploitation** | **Exploration** |
|---|---|---|
| Internal Improvement | Inertia weighted PSO [13] | PSO with neighborhood topologies [5][16][17] [18] |
| | Constriction PSO [14][37] | Unified PSO [19] |
| | Guaranteed Convergence PSO [7] | Fully Informed PSO [20] |
| | Fitness-distance ratio based PSO [15] | Comprehensive Learning PSO [4] |
| External Improvement | PSO with SA [21], PSO-ACO [11] | Mutated PSO [22] |
| | PSO with VNS [12], DEPSO [24], NMPSO [6] | Variable Neighborhood PSO [26] |

# 4. StPSO: Strengthened particle swarm optimization

Pseudo-code of the StPSO algorithm is illustrated in Figure 2. Main focus of our modifications is on pioneering-particles which achieves or enhances the swarm's experience. These particles are either converged or potentially converged. They are processed in two steps: Firstly, an external local search is initiated for each pioneering-particle (line 11). This step strengthens exploitation mechanism in PSO. Secondly, at the same iteration a random velocity is assigned to each pioneering-particle in order to force the particle to continue exploration (line 13). In this way, the exploration mechanism of PSO is improved and premature convergence is avoided.

## 4.1. Local Search: RVNS

Reduce Variable Neighborhood Search (RVNS) is employed in StPSO as a local search method. RVNS is a variation of the Variable Neighborhood Search (VNS) which is initially introduced as an optimization method

for combinatorial optimization problems [27]. In VNS, solution space is searched with a systematic change of neighborhood. It has two main steps named *LocalSearch* and *Shake*. VNS becomes RVNS if *LocalSearch* step is removed. RVNS is usually preferred as a general optimization method for problems where exhaustive local search is expensive [28].

```
 1 Procedure StPSO
 2  Initialize Parameters
 3  Initialize Population
 4  For each particle
 5    Evaluate
 6    Update local best
 7  Update global best
 8  Do
 9   For each particle
10      If (fitness equal to fitness of gbest)
11         position←RVNS_LocalSearch(position)
12         Update local best
13         Create new random velocity
14       Else
15         Update velocity
16       Update position
17       Evaluate
18       Update local best
19   Update global best
20  While (Not Terminated)
21 End Procedure
```

**Figure 2.** The Strengthened PSO algorithm.

The pseudo-code of RVNS is given in Figure 3, where $N_k$ (line 2) represents $k^{th}$ neighborhood structure $N_k(s)$ represents the set of solutions in the $k^{th}$ neighborhood of the solution s. Starting from a solution and the first neighborhood (s and $N_1$, respectively), during each iteration of the inner loop, a random solutions s' is selected from the current neighborhood (line 7). If s' is better than s, it replaces s and the search continues with the first neighborhood (line 8-10). Otherwise, the algorithm switches to the next neighborhood structure (line 12). If all neighborhood structures are exhausted, the inner loop is initiated all over again starting from the first neighborhood. The outer loop is repeated until a stopping condition is met. The maximum CPU time, the maximum number of iterations or the maximum number of iterations between two improvements can be used as a stopping condition.

Both the choice and the order of neighborhood structures are critical for the performance of the RVNS algorithm. If neighbor hoods are large, the algorithm explores almost entire solution space quickly with big jumps. On the other hand, narrow neighborhood sizes lead to small footsteps which in turn results in fast exploitation ability. Generally, neighborhoods are ordered from the smallest to the largest.

In StPSO, RVNS may take advantage of the distance between the pioneering-particle and its closest neighbor. At early stages of PSO, the neighborhoods are large since the neighbors of the pioneering-particle are not very close by. So, RVNS helps in exploration. Later, as many particles in the swarm get attracted towards better solutions, the neighborhood size tapers. At that point, RVNS performs a fine-tuned local search around

the swarm's experience. As a result, RVNS may support both exploration and exploitation mechanisms of PSO by adjusting its behavior dynamically.

```
Procedure RVNS_LocalSearch(position p)
 Define neighborhood structures N_k (k=1,...,k_max)
 Use position of p as initial solution s
 while stopping condition is not met do
    k ← 1
    while k ≤ k_max do
       s' ← Shake(s), s' ∈ N_k(s)
       if (Fitness(s')<Fitness(s))
          s ← s'
          k ← 1
       else
          k ← k+1
    end-while
 end-while
End-Procedure
```

**Figure 3.** The RVNS algorithm.

In many studies, a local search is incorporated in the PSO algorithm. For example, a local search is employed for all particles at each iteration in PSOwithSA and after every four iterations in DEPSO. In NM-PSO, a local search is applied to one third of particles and in PSOwithVNS a local search is employed for only the global best position at each iteration.

In PSOwithSA and DEPSO, Simulated Annealing and Differential Evaluation are the central search method, respectively. PSO is only used to generate promising initial solutions. However, in StPSO, PSO is not passive but actively improve the current best solution, and RVNS is employed for faster local search. Moreover, application of local search to all particle positions at each iteration is a very expensive approach for larger size problems.

In order to decrease local search cost, in PSOwithVNS, the local search (VNS) is employed only for the global best position at each iteration. However, VNS is applied several times to the same position if the global best does not change through iterations. The possibility of PSO to change the global best position is very low since PSO has to produce better solution than a solution generated by a detailed local search method. Thus, VNS is the dominant part in PSOwithVNS, as well. On the other hand, in StPSO, the local search (RVNS) is applied to the pioneering-particles which achieve or enhance the global best. At an iteration, there may be several pioneering-particles or none. So, the local search is employed moderately when it is necessary. Moreover, the result of RVNS affects only the pioneering-particle's own experience and position and does not affect the global best solution. In this way, dissemination of knowledge about a good position within the swarm slows down. For any particle, the possibility of being a pioneering-particle during subsequent PSO iterations is not decreased.

## 4.2. Random moving strategy

In the second part of our modifications, a separate velocity update procedure is applied to the pioneering-particles. A random velocity vector which is uniformly distributed within the interval $[-V_{max}, V_{min}]$ is assigned

to the particle. Note that, the position and the local experience of each pioneering-particle are just updated after the local search. During the same iteration, the position of the particle is recalculated using its new velocity. The experience of the particle and swarm's experience are updated based on the new position, as well, but it is not very likely. In this way, instead of possible convergence, the particle jumps to a completely different solution and continue its search with its past experience.

Various random moving strategies are employed to prevent premature convergence in previous studies. In GCPSO, a random velocity term is appended into the velocity update formula. This new formula is used only for the global best particle. Similarly in DEPSO, random positions in the neighborhood of the global best position are assigned to some particles based on their local experience. In these algorithms, random moving strategies only perturb the particles but do not move them far away. So, random moving strategy enhances the exploitation mechanisms in GCPSO and DEPSO. On the other hand, it is used to improve the exploration mechanism in StPSO as in VNPSO.

In VNPSO, random moving strategy is applied to awake "lazy" particles. Lazy particles are determined by a predefined threshold velocity value and in one iteration all lazy particles are relocated to random places. Although two level thresholds are used, it is still predetermined. However, in StPSO, the pioneering-particles are determined dynamically disregarding the velocity values.

To summarize, external local search (first modification) in StPSO improves exploitation mechanism and velocity re-initialization (second modification) improves exploration mechanism in the standard PSO algorithm. Two modifications together make PSO strengthened. Moreover, StPSO provides a dynamic balance between exploration and exploitation which is essential to obtain enhanced solutions for many optimization problems.

# 5. Test problems

We experimented StPSO on both continuous and discrete problems to demonstrate its applicability to various problem domains.

## 5.1. Continuous function optimization problems

Six continuous benchmark functions which have been used in various PSO studies are experimented. They are categorized into two groups: unimodal ($f_1$ and $f_2$) and multimodal ($f_3$ – $f_6$). The optimum of all problems is located at the origin. All functions with their properties are presented in Table 2.

### 5.1.1. Problem representation

Position vector of a particle directly represents the solution of the function optimization problems. The fitness value of a particle is obtained by applying the function to its position vector.

### 5.1.2. RVNS neighborhood structures

For the function optimization problems, $d$-dimensional balls are used as neighborhood structures. Neighborhoods are determined dynamically by using a method similar to local search area determination in [29] . The distance ($l$) between the position of the pioneering-particle and the nearest particle in the swarm is used in the

calculation of the neighborhood radius. For the first neighborhood structure, the distance is multiplied by a low quotient ($q_1$) in order to search around closed by neighborhood. The radius of the second neighborhood is calculated by multiplying the distance by a high quotient ($q_2$) so, a larger neighborhood is examined. The quotients, $q_1$ and $q_2$ are two additional parameters of our method which are determined through experimental evaluation.

**Table 2.** Benchmark functions for continuous optimization.

| Name | Formula | Number of Dimensions | Search Domain |
|---|---|---|---|
| Sphere ($f_1$) | $f(x) = \sum_{i=1}^{d} x_i^2$ | 30 | [-100,100]$^d$ |
| Rosenbrock ($f_2$) | $f(x) = \sum_{i=1}^{d-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$ | 30 | [-30,30]$^d$ |
| Griewank ($f_3$) | $f(x) = \frac{1}{4000} \sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos(\frac{x_i}{\sqrt{i}}) + 1$ | 30 | [-600,600]$^d$ |
| Rastrigin ($f_4$) | $f(x) = \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i) + 10)$ | 30 | [-5.12,5.12]$^d$ |
| Schwefel ($f_5$) | $f(x) = 418.9829d + \sum_{i=1}^{d} x_i \sin(\sqrt{\lvert x_i \rvert})$ | 30 | [-500,500]$^d$ |
| Ackley ($f_6$) | $f(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{d} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{d}\cos(2\pi x_i)) + 20 + e$ | 30 | [-30,30]$^d$ |

## 5.2. Orienteering problem

The Orienteering Problem (OP) is a subset selection version of the well-known Traveling Salesman Problem with profits. The objective of the Orienteering Problem is to construct a path beginning at a starting point and ending at a destination point that maximizes the total profit without violating prescribed limits. The problem is inspired from and named after an outdoor sport usually played on mountains or forest areas. OP is used to model many practical problems such as inventory routing, postal delivery, customer or vehicle assignment and production scheduling.

The Orienteering Problem can be described in detail as follows: Let $G = (X, E)$ be a graph where $X$ denotes the set of control points and $E$ denotes the set of edges between the nodes in $X$. Each node $x_i$ in $X$ ($0 \leq i \leq d+1$) has a score and the scores of $x_0$ (the starting point) and $x_{d+1}$ (the destination point) are zero. Each edge between $x_i$ and $x_j$ has a cost associated with it. The objective of OP is to find an acyclic path from $x_0$ to $x_{d+1}$ that maximizes the total score of the nodes on the path without violating a cost constraint (e.g., the total cost of the edges on the path should be less than a specified limit, $T_{\max}$) Because of this limitation, the path does not necessarily visit all control points. Golden et al. prove that OP is in NP-hard [30]. The mathematical model of OP can be found in [31].

Several metaheuristic including genetic algorithm [32], ant colony [33], and variable neighborhood search [34] are used to solve OP. A comprehensive survey of methods used for OP is presented in [34]. The algorithms are tested for 63 benchmark problems: Problems with 32 (dataset1 includes 16 problems) 21 (dataset2 includes

11 problems), 33 (dataset3 includes 20 problems) control points are provided by Tsiligirides [35] and problems with 32 (dataset 4 includes 16 problems) control points are provided by Chao [36].

### 5.2.1. Problem Representation

Since PSO is developed for continuous problems, some adaptations are required for discrete problems. There are two popular approaches: One is to incorporate and extra conversion process to transform a continuous position vector to the corresponding permutation vector. In this way, all operators and components of the standard PSO remain unchanged. The algorithm always switches between two different search domains, continuous search space and discrete solution space. The second approach is to convert all operators and components of the PSO in order to perform discrete processing. Although, there is a consistency between solution space and algorithm's search domain, the adaptation is more challenging (several operators and components should be redefined)

We employ the first approach; the position of each particle is a $d$-dimensional vector. The continuous position vector transformed to a permutation vector in order to find OP solution. We use the smallest position value (SPV) rule [12] for this transformation. The SPV rule works as follows: Assume that $p_i$ is the value at index $i$ in the position vector. After sorting the values in the position vector, the index of $p_i$ becomes $j$. In the permutation represented by the position vector, the value at the index $i$ is $j$. An example of the SPV rule is given in Figure 4.

| index | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| position vector | 2.5 | 5.7 | 0.7 | 1.3 |

| index | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| sorted position vector | 0.7 | 1.3 | 2.5 | 5.7 |

| index | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| permutation vector | 3 | 4 | 1 | 2 |

**Figure 4.** An example of the smallest position value (SPV) rule.

By the nature of OP, a candidate solution may not include all the control points. So, the permutation vector is not used as a candidate solution directly. The solution is obtained by using the permutation vector as follows: Starting from the first control point in the permutation vector, the control points are inserted between the starting point and the destination point, one by one, until the prescribed cost limit is exceeded.

### 5.2.2. RVNS neighborhood structures

For the Orienteering Problem, *Insert* and *Exchange* neighborhood structures are used in the RVNS algorithm. These structures can be implemented easily on permutation based solutions. *Insert* neighborhood performs an insertion of a control point chosen randomly from the permutation in front of another randomly chosen control point. *Exchange* is used to explore new solutions in a little further vicinity of a solution. In this neighborhood, two randomly selected control points are swapped.

# 6. Experimental results

To observe effects of the proposed modifications, we perform experimental analysis on StPSO and its two sub-variants. One sub-variant which only assigns a random velocity to the pioneering-particle without performing the local search is called as Diversification Strengthened PSO (DS-PSO). Particles in DS-PSO tend to perform more exploration than exploitation since they are not allowed to stay around good positions. The other sub-variant which conducts only RVNS for pioneering-particles is called as Intensification Strengthened PSO (IS-PSO).

Experiments are performed on an Intel P4 2.8 GHz PC with 1 GB of memory. All PSO and RVNS parameters and their values are listed in Table 3. The GBEST model is used in which all particles are considered to find global best value in the swarm. Number of replication is 10 for OP and 30 for the function optimization problems. For both types of problems, initial positions are generated randomly using uniform distribution in search domain. During the search process, neither position nor velocity vectors are allowed to get value outside of their range. If the calculated value is out of the range, a random value is used.

**Table 3.** Parameter settings for StPSO.

| PSO | Function Optimization Problems | | | | | | | Orienteering Problem |
|---|---|---|---|---|---|---|---|---|
| Swarm Size | 40 | | | | | | | 20 |
| Stopping Condition (function evaluation) | 200,000 | | | | | | | 100,000 |
| $w$ | 0.9-0.4 | | | | | | | 0.9-0.4 |
| $c_1$-$c_2$ | $c_1$=$c_2$ | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | 2.0 |
| | IwPSO | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | |
| | The others | 1.1 | 1.1 | 1.1 | 2.0 | 2.0 | 2.0 | |
| Velocity range | Rastrigin:[-0.5,+0.5]<br>The others:[-4,+4] , | | | | | | | [$-d$,+$d$] |
| Position range | refer to Table 2 | | | | | | | [$-d$,+$d$] |
| **RVNS** | **Function Optimization** | | | | | | | **Orienteering Problem** |
| # of Neighborhoods | 2 | | | | | | | 2 |
| Neighborhood Structures | $d$-dim. balls with ( $q_1 \times l$ ) where $q_1$=0.2 | | | | | | | Insert |
| | $d$-dim. balls with ( $q_2 \times l$ ) where $q_2$=0.5 | | | | | | | Exchange |
| Stopping Condition | $d \times 5$ iteration or 30 times back-to-back no improvements | | | | | | | $d \times d$ iteration |

We report our experimental analysis for OP and the continuous optimization problems in two parts. In the first part, the aim is to present effects of the proposed modifications over a standard PSO variant, inertia weighted PSO (IwPSO). The analysis is performed based on the number of fitness function evaluations (Eval.), the time spent (CPU) until the best solution is found and the quality of the best solution (Fitness). The results are presented for the repetition yielding the best solution (Best), for the repetition yielding the worst solution (Worst), for the repetition yielding the median solution (Median) and averaged over all repetitions (Mean). In the second part, we compare our results with the results in several previous studies.

## 6.1. Results for continuous function optimization problems

The performance comparison of our algorithms with IwPSO is presented in Tables 4, 5 and Figures 5, 6. The graphs in the figures illustrate the convergence behavior of Median. The results for unimodal function optimization problems (see Table 4) indicate the impact of exploitation mechanism in our algorithm. Since the algorithms (IwPSO, IS-PSO, and StPSO) except DS-PSO have inherited or imported exploitation mechanisms, their results are better than DS-PSO for Spherical. IS-PSO with its strengthened exploitation mechanism achieves faster improvement (check the CPU times) and enhances average results of IwPSO almost 200 order of magnitude for Spherical. It takes less time to terminate since a local search is substantially faster in search operation than a PSO algorithm. For Rosenbrock, IS-PSO produces the best fitness in Best category; however StPSO outperforms IS-PSO based on other performance measures. The dynamic balance between exploration and exploitation pays off for Rosenbrock. Effects of the premature convergence problem are not observed (see Figure 5).

**Table 4.** Performance of IwPSO, IS-PSO, DS-PSO and St-PSO for 30-D unimodal problems.

| | IwPSO | | | IS-PSO | | | DS-PSO | | | StPSO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fitness | Eval. | CPU | Fitness | Eval. | CPU | Fitness | Eval. | CPU | Fitness | Eval. | CPU |
| Mean | 7.93E-102 | 199,993.1 | 100.68 | 2.14E-322 | 199,610.2 | 61.83 | 1.68E-41 | 199,954.7 | 100.30 | 5.44E-194 | 199,725.8 | 72.94 |
| Best | 1.24E-107 | 199,960.0 | 98.52 | 0.00E+00 | 197,921.0 | 60.61 | 5.92E-44 | 199,760.0 | 99.63 | 1.87E-217 | 198,846.0 | 70.55 |
| Median | 3.44E-104 | 200,000.0 | 99.75 | 1.01E-323 | 199,736.0 | 61.51 | 4.80E-42 | 199,960.0 | 99.72 | 1.19E-206 | 199,792.0 | 72.64 |
| Worst | 1.55E-100 | 200,000.0 | 101.75 | 5.42E-321 | 200,000.0 | 64.19 | 1.65E-40 | 200,000.0 | 103.77 | 1.41E-192 | 200,000.0 | 81.56 |
| Std dv. | 2.91E-101 | 15.38 | 2.22 | ~ 0.00E+00 | 520.8 | 0.95 | 3.57E-41 | 61.0 | 1.18 | ~ 0.00E+00 | 297.8 | 2.10 |

a) Spherical Function

| | IwPSO | | | IS-PSO | | | DS-PSO | | | StPSO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fitness | Eval. | CPU | Fitness | Eval. | CPU | Fitness | Eval. | CPU | Fitness | Eval. | CPU |
| Mean | 2.75E+01 | 164,038.7 | 80.31 | 6.69E+00 | 199,999.5 | 59.40 | 1.12E+01 | 200,000.0 | 102.59 | 7.90E-01 | 200,000.0 | 26.41 |
| Best | 3.52E-03 | 99,000.0 | 48.36 | 2.79E-13 | 199,985.0 | 54.19 | 1.19E-02 | 200,000.0 | 99.60 | 7.54E-05 | 200,000.0 | 23.28 |
| Median | 2.13E+01 | 165,100.0 | 80.66 | 8.30E+00 | 200,000.0 | 59.66 | 1.08E+01 | 200,000.0 | 101.04 | 1.86E-01 | 200,000.0 | 26.42 |
| Worst | 7.69E+01 | 200,000.0 | 98.35 | 1.32E+01 | 200,000.0 | 64.88 | 5.74E+01 | 200,000.0 | 103.88 | 4.12E+00 | 200,000.0 | 33.31 |
| Std dv. | 2.41E+01 | 31,955.5 | 15.62 | 4.50E+00 | 2.7 | 2.84 | 1.06E+01 | 0.0 | 3.97 | 1.49E+00 | 0.0 | 2.04 |

b) Rosenbrock Function

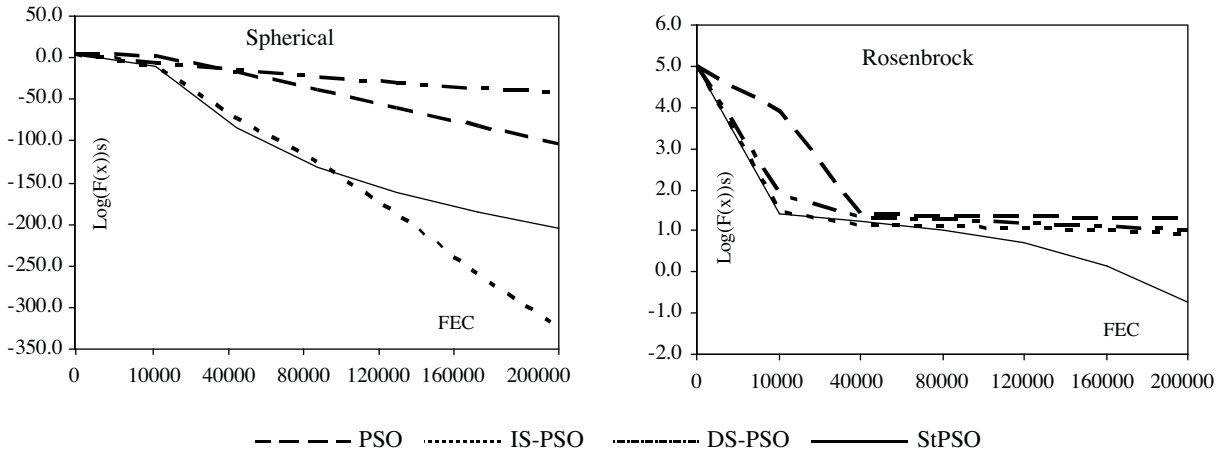


**Figure 5.** Change in median fitness value (logarithmic scale) of the IwPSO, IS-PSO, DS-PSO and StPSO for unimodal function optimization problems over 200.000 function evaluations (FEC: Function Evaluation Count).

Results for the multimodal functions are illustrated in Table 5 and Figure 6. The results support the hypothesis that IwPSO has a premature convergence problem for multimodal functions. For most of the repetitions, the IwPSO converges and provides almost no improvement after the first 50,000-60,000 evaluations. We attack to this problem by our modifications. Both DS-PSO and StPSO do not have such a problem, they

continue to produce better solutions until the termination at 200,000 function evaluations. Moreover, they perform better than IwPSO for almost all performance measures.

**Table 5** Performance of IwPSO, IS-PSO, DS-PSO and St-PSO for 30-D multimodal problems.

| | IwPSO | | | IS-PSO | | | DS-PSO | | | StPSO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fitness | Eval. | CPU | Fitness | Eval. | CPU | Fitness | Eval. | CPU | Fitness | Eval. | CPU |
| Mean | 5.63E-02 | 64,754.7 | 32.69 | 9.27E-03 | 26,161.2 | 5.58 | 1.24E-02 | 88,153.3 | 44.65 | 1.15E-02 | 82,993.4 | 15.54 |
| Best | 0.00E+00 | 38,880.0 | 19.27 | 1.11E-16 | 13,842.0 | 3.99 | 0.00E+00 | 59,240.0 | 30.02 | 0.00E+00 | 27,421.0 | 4.94 |
| Median | 2.22E-02 | 48,840.0 | 24.39 | 7.40E-03 | 14,687.0 | 4.54 | 7.40E-03 | 72,200.0 | 37.33 | 8.63E-03 | 65,877.0 | 8.86 |
| Worst | 6.52E-01 | 177,760.0 | 88.19 | 4.18E-02 | 183,273.0 | 20.17 | 7.11E-02 | 184,680.0 | 93.08 | 5.86E-02 | 189,979.0 | 62.59 |
| Std dv. | 1.18E-01 | 37,873.7 | 18.94 | 1.11E-02 | 40,010.7 | 3.75 | 1.55E-02 | 34,628.7 | 17.46 | 1.29E-02 | 48,604.1 | 13.87 |

a) Griewank Function

| | IwPSO | | | IS-PSO | | | DS-PSO | | | StPSO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fitness | Eval. | CPU | Fitness | Eval. | CPU | Fitness | Eval. | CPU | Fitness | Eval. | CPU |
| Mean | 1.28E+00 | 77,018.7 | 40.79 | 1.69E+00 | 62,837.8 | 19.06 | 1.88E-14 | 180,545.3 | 96.10 | 7.55E-15 | 99,263.7 | 31.34 |
| Best | 7.55E-15 | 37,200.0 | 19.86 | 7.55E-15 | 30,989.0 | 11.26 | 1.47E-14 | 150,880.0 | 78.02 | 7.55E-15 | 62,917.0 | 19.94 |
| Median | 1.50E+00 | 68,120.0 | 36.87 | 1.78E+00 | 51,269.0 | 17.39 | 2.00E-14 | 180,560.0 | 96.58 | 7.55E-15 | 92,020.5 | 28.46 |
| Worst | 2.74E+00 | 200,000.0 | 106.76 | 2.74E+00 | 194,276.0 | 52.82 | 2.89E-14 | 199,600.0 | 105.46 | 7.55E-15 | 152,282.0 | 47.56 |
| Std dv. | 8.71E-01 | 39,601.9 | 20.98 | 7.08E-01 | 38,469.2 | 9.29 | 4.08E-15 | 12,417.3 | 6.56 | 3.21E-30 | 25,199.0 | 7.14 |

b) Ackley Function

| | IwPSO | | | IS-PSO | | | DS-PSO | | | StPSO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fitness | Eval. | CPU | Fitness | Eval. | CPU | Fitness | Eval. | CPU | Fitness | Eval. | CPU |
| Mean | 1.65E+03 | 89,937.1 | 47.52 | 2.15E+03 | 78,818.3 | 22.68 | 5.23E+02 | 199,458.7 | 106.41 | 3.40E+02 | 187,467.6 | 59.14 |
| Best | 8.29E+02 | 47,320.0 | 24.83 | 1.36E+03 | 37,822.0 | 12.00 | 1.18E+02 | 192,200.0 | 102.16 | 1.18E+02 | 151,980.0 | 49.50 |
| Median | 1.72E+03 | 81,060.0 | 42.71 | 2.00E+03 | 66,025.0 | 19.91 | 4.76E+02 | 199,820.0 | 106.15 | 3.55E+02 | 190,289.5 | 58.69 |
| Worst | 2.65E+03 | 175,680.0 | 93.14 | 3.36E+03 | 155,819.0 | 47.83 | 1.07E+03 | 200,000.0 | 115.62 | 5.92E+02 | 200,000.0 | 75.54 |
| Std dv. | 4.09E+02 | 35,195.9 | 18.53 | 5.56E+02 | 38,003.2 | 9.39 | 2.82E+02 | 1,404.0 | 2.97 | 1.42E+02 | 13,718.8 | 6.02 |

c) Schwefel Function

| | IwPSO | | | IS-PSO | | | DS-PSO | | | StPSO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fitness | Eval. | CPU | Fitness | Eval. | CPU | Fitness | Eval. | CPU | Fitness | Eval. | CPU |
| Mean | 4.92E+01 | 82,636.0 | 48.94 | 4.28E+01 | 110,841.9 | 37.15 | 2.48E+00 | 199,762.7 | 101.38 | 2.35E+00 | 199,863.5 | 99.10 |
| Best | 2.79E+01 | 34,920.0 | 20.47 | 1.69E+01 | 35,540.0 | 13.44 | 1.26E-10 | 196,760.0 | 99.50 | 1.26E-10 | 198,960.0 | 96.61 |
| Median | 4.88E+01 | 71,060.0 | 42.13 | 4.08E+01 | 107,651.5 | 34.99 | 1.50E+00 | 199,960.0 | 100.86 | 1.00E+00 | 199,960.0 | 98.92 |
| Worst | 8.36E+01 | 170,040.0 | 100.73 | 7.26E+01 | 199,893.0 | 67.74 | 1.49E+01 | 200,000.0 | 103.47 | 1.49E+01 | 200,000.0 | 101.17 |
| Std dv. | 1.27E+01 | 36,111.3 | 21.40 | 1.39E+01 | 47,810.4 | 15.43 | 2.98E+00 | 611.7 | 1.43 | 2.95E+00 | 229.7 | 0.80 |

d) Rastrigin Function

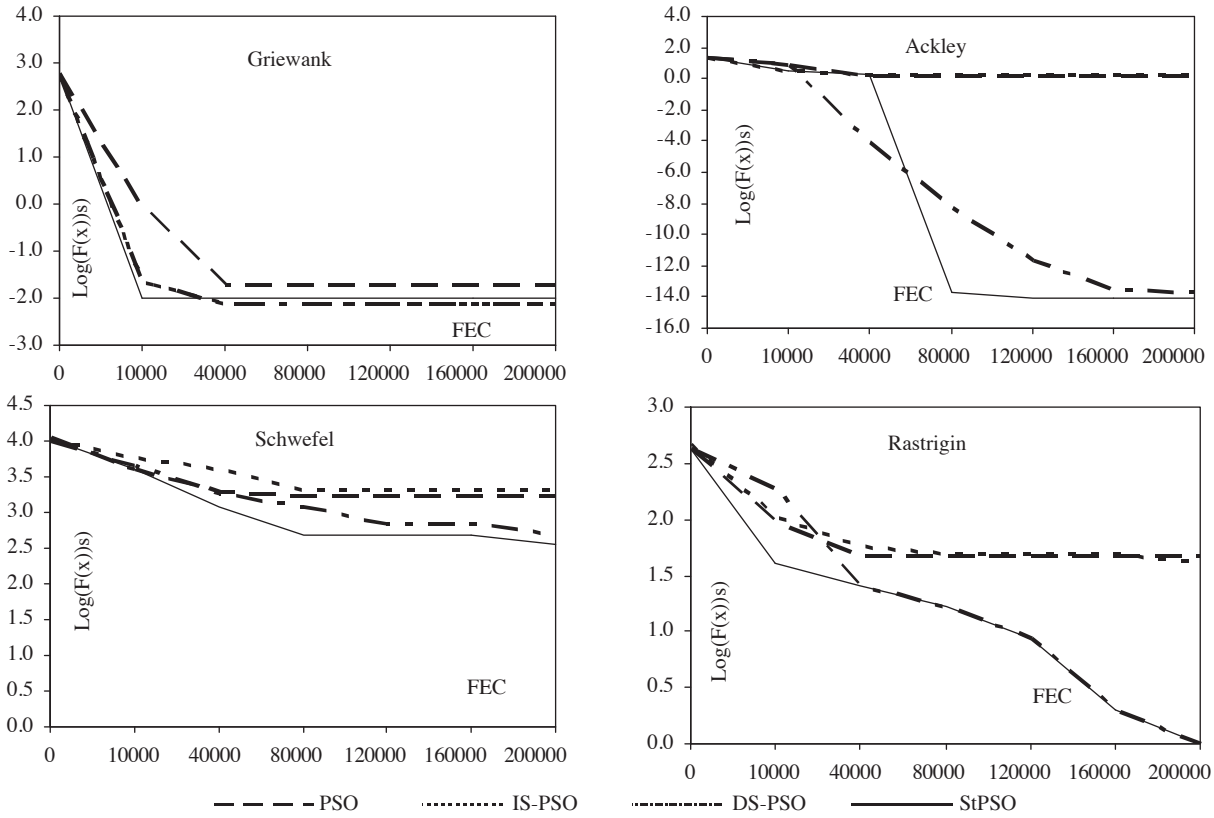


**Figure 6.** Change in median fitness value (logarithmic scale) of the IwPSO, IS-PSO, DS-PSO and StPSO for multimodal function optimization problems over 200.000 function evaluations (FEC: Function Evaluation Count).

The optimal solution is found for Griewank in some repetitions. While the optimal solution is obtained in 5 repetitions using IwPSO, this number is more than 10 for DS-PSO and StPSO. From the corresponding graph for Griewank it is possible to conclude incorrectly that after a number of evaluations, StPSO can not further improve the best solution found so far. However, this is mainly because the optimal solution has already been found for most of the repetitions.

For the remaining three multimodal problems (Ackley, Schwefel and Rastrigin) behavior of the algorithms are quite similar. DS-PSO starts to provide better solutions than IwPSO after approximately 10,000 function evaluations. During early iterations in the search, with the effect of re-initiation which hinders exploitation mechanism, DS-PSO is not much successful. However, the premature convergence problem of IwPSO becomes significant later and DS-PSO outperforms IwPSO. On the other hand, StPSO is almost always more successful than others in finding better solutions in a shorter time.

In the second part of experimental analysis for the function optimization problems, we compare mean and standard deviation of solutions obtained by StPSO with the results obtained by several other algorithms in Table 6. The termination criteria for all algorithms are the same (200,000 evaluations) and results for the other algorithms are taken from the references that are given in Table 6. In this table, "# of success" fields present the comparisons with the other algorithms. The value "$a+b$-" means that, among the algorithms listed in the table, the number of algorithms better than our algorithm is $b$ and the number of algorithms worse than our algorithm is $a$.

Although our algorithms do not always produce the best solutions, the results are comparable and close to the best solutions for all problems either unimodal or multimodal. On the other hand, while GCPSO, conPSO, and FDRPSO are more suitable for the unimodal problems, MPSO, UPSO, FIPSO, and CLPSO are better for the multimodal problems. The success rates of the algorithms in Table 7 presents a clear evidence for this claim.

## 6.2. Results for the orienteering problem

The performance comparison of our algorithms on the Orienteering Problem with IwPSO is presented in Table 8. The results in the table are average values over 63 instances of OP in four datasets. While IwPSO is the fastest method, it produces the lowest average score. Performance of the DS-PSO is better than the IwPSO with large function evaluation count and execution time. When local search method is embedded, in IS-PSO and StPSO, better solutions are obtained. While IS-PSO finds the best known solutions for all problems, StPSO fails to produce the best known solutions for only three problem instances (for Tmax=60 in dataset1 2.2% gap, for Tmax=80 and Tmax=85 in dataset3 1.4% gap and 1.3% gap, respectively) in 100.000 evaluation.

The best results obtained by IS-PSO are compared with the best results obtained by following algorithms in Tables 9-12:

- CGW: Heuristic algorithm [36] (experimented on datasets 1, 2, 3, 4);
- GA: Genetic algorithm [35] (experimented on datasets 1, 2, 3, 4);
- ACO: Ant colony optimization [33] (experimented on datasets 1, 2, 3, 4);
- VNS: Variable neighborhood search [34] (experimented on datasets 1, 2, 3, 4);
- ARPSO: Attractive and repulsive PSO [37] (experimented on dataset 2);
- GLS: Guided local search [38] (experimented on datasets 1, 2, 3).

**Table 6.** Performance of IS-PSO, DS-PSO and St-PSO against the previous studies for function optimization problems.

| | | Unimodal | | Multimodal | | | |
|---|---|---|---|---|---|---|---|
| | | Spherical | Rosenbrock | Griewank | Ackley | Rastrigin | Schwefel |
| Exploitation improved algorithms | *PSO-w* [4] | 9.78E-30 (2.50E-29) | 2.93E+01 (2.51E+01) | 8.13E-03 (7.16E-03) | 3.94E-14 (1.12E+00) | 2.80E+01 (7.70E+00) | 1.10E+03 (2.56E+02) |
| | *ConPSO* [4] | 5.88E-100 (5.40E-100) | 1.11E+01 (1.81E+00) | 2.06E-02 (1.90E-02) | 1.12E+00 (8.65E-01) | 5.62E+01 (9.76E+00) | 3.78E+03 (6.02E+02) |
| | *GCPSO-g* [5] | 3.00E-161 (1.00E-161) | **1.29E-02** (2.50E-02) | 1.62E-02 (2.20E-02) | 2.02E+00 (1.31E+00) | 7.19E+01 (1.86E+01) | 4.54E+03 (7.06E+02) |
| | *FDRPSO* [4] | 4.88E-102 (1.53E-101) | 5.39E+00 (1.76E+00) | 1.01E-02 (1.23E-02) | 2.84E-14 (4.10E-01) | 2.84E+01 (8.71E+00) | 3.61E+03 (3.06E+02) |
| | *NM-PSO* [6] | - | 2.00E-02 (6.00E-04) | 1.52E-02 (9.90E-03) | 3.15E-06 (8.10E-07) | **7.08E-11** (4.11E-11) | - |
| Exploration improved algorithms | *PSO-w-l* [4] | 5.35E-100 (4.41E-13) | 2.39E+01 (3.07E+00) | 5.91E-03 (6.69E-03) | 9.10E-08 (8.11E-08) | 2.72E+01 (7.58E+00) | 1.53E+03 (3.00E+02) |
| | *ConPSO-l* [4] | 7.7E-54 (1.59E-53) | 1.71E+01 (9.16E-01) | 5.91E-03 (8.70E-03) | 5.33E-15 (1.87E-15) | 4.53E+00 (1.17E+01) | 3.78E+03 (5.37E+02) |
| | *GCPSO-l* [5] | 3.00E-93 (2.00E-92) | 6.50E-01 (5.30E-01) | 3.90E-03 (7.50E-03) | 2.79E-01 (2.80E-01) | 6.12E+01 (1.54E+01) | 4.76E+03 (5.09E+02) |
| | *GCPSO-v* [5] | 6.00E-119 (3.00E-118) | 1.80E-01 (9.70E-02) | 1.04E-02 (1.45E-02) | 6.82E-01 (8.26E-01) | 5.58E+01 (1.45E+01) | 4.50E+03 (7.07E+02) |
| | *MPSO-g* [22] | - | - | 2.53E-02 (2.24E-02) | **0.00E+00** (0.00E+00) | 4.69E+01 (1.15E+01) | 1.50E+03 (3.04E+02) |
| | *MPSO-l* [22] | - | - | 1.40E-03 (4.7E-03) | **0.00E+00** (0.00E+00) | 3.00E+00 (3.82E+00) | 1.83E+03 (1.92E+02) |
| | *UPSO* [4] | 4.17E-87 (3.15E-87) | 1.51E+01 (8.14E-01) | 1.66E-03 (3.07E-03) | 1.22E-15 (3.16E-15) | 6.59E+01 (1.22E+01) | 4.84E+03 (4.76E+02) |
| | *FIPSO* [4] | 2.69E-12 (6.84E-13) | 2.45E+01 (2.19E-01) | 1.16E-06 (1.87E-06) | 4.81E-07 (9.17E-08) | 7.30E+01 (1.24E+01) | 2.05E+03 (9.58E+02) |
| | *CLPSO* [4] | 4.46E-14 (1.73E-14) | 2.10E+01 (2.98E+00) | **3.14E-10** (4.64E-10) | **0.00E+00** (0.00E+00) | 4.85E-10 (3.63E-10) | **1.27E-12** (8.79E-13) |
| Our algorithms | ***DS-PSO*** | 1.68E-41 (4.16E-42) | 1.12E+01 (1.06E+01) | 1.24E-02 (1.55E-02) | 1.88E-14 (4.08E-15) | 2.48E+00 (2.98E+00) | 5.23E+02 (2.82E+02) |
| | ***# of success*** | 2+ 9- | 7+ 5- | 6+ 8- | 9+ 5- | 12+ 2- | 12+ 1- |
| | ***IS-PSO*** | **2.14E-322** (~0.00E+0) | 6.69E+00 (4.50E+00) | 9.27E-03 (1.11E-02) | 1.69E+00 (7.08E-01) | 4.28E+01 (1.39E+01) | 2.15E+03 (5.56E+02) |
| | ***# of success*** | 11+ 0- | 8+ 4- | 7+ 7- | 2+ 12- | 8+ 6- | 8+ 5- |
| | ***StPSO*** | **5.45E-194** (~0.00E+0) | 9.60E+00 (3.31E+00) | 1.15E-02 (1.29E-02) | 7.55E-15 (3.21E-30) | 2.35E+00 (2.95E+00) | 3.40E+02 (1.42E+02) |
| | ***# of success*** | 11+ 0- | 7+ 5- | 6+ 8- | 9 + 5- | 12+ 2- | 12+ 1- |

**Table 7.** Success percentage of the algorithms against the other algorithms for unimodal and multimodal function optimization problems.

| GCPSO-g | | CLPSO | | DS-PSO | | IS-PSO | | StPSO | |
|---|---|---|---|---|---|---|---|---|---|
| Unimodal | Multimodal | Unimodal | Multimodal | Unimodal | Multimodal | Unimodal | Multimodal | Unimodal | Multimodal |
| 21+ 2- | 5+ 50- | 4+ 19- | 52+ 1- | 9+ 14- | 37+ 18- | 19+ 4- | 23+ 30- | 18+ 5- | 37+ 18- |
| 91% | 9% | 17% | 98% | 39% | 70% | 83% | 43% | 78% | 70% |

**Table 8.** Performance of IwPSO, DS-PSO, IS-PSO and StPSO application for Orienteering Problem: The average results for 63 problems based on Fitness, Function Evaluaton Count and CPU.

| | IwPSO | | | IS-PSO | | | DS-PSO | | | StPSO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fitness | Eval | CPU | Fitness | Eval | CPU | Fitness | Eval | CPU | Fitness | Eval | CPU |
| Mean | 195.37 | 2,339.3 | 1.28 | 319.59 | 28,674.0 | 4.2 | 258.73 | 34,364.3 | 16.06 | 319.42 | 30,029.1 | 4.39 |
| Best | 234.68 | 160.9 | 0.11 | 322.14 | 5,173.7 | 0.77 | 286.06 | 4,102.3 | 2.01 | 321.75 | 4,855.2 | 0.72 |
| Worst | 99.04 | 15,311.5 | 7.78 | 314.52 | 65,837.4 | 9.58 | 231.91 | 83,142.9 | 38.56 | 313.25 | 68,566.5 | 9.94 |
| Std Dv. | 40.94 | 5,085.1 | 2.57 | 2.84 | 21,009.4 | 3.06 | 17.43 | 27,657.3 | 12.79 | 3.10 | 22,074.5 | 3.19 |

**Table 9.** Performance of IS-PSO for dataset 1 (32 nodes and 16 instances ).

| | | GLS Fitness | IS-PSO Fitness | | | | IS-PSO Eval | IS-PSO CPU |
|---|---|---|---|---|---|---|---|---|
| Tmax | Optimum | Best | Best | Worst | Mean | StDev | Mean | Mean |
| 15 | 45 | 45 | 45 | 45 | 45.0 | 0.00 | 7087.5 | 0.97 |
| 20 | 65 | 55 | 65 | 65 | 65.0 | 0.00 | 21080.8 | 2.90 |
| 25 | 90 | 90 | 90 | 90 | 90.0 | 0.00 | 39325.2 | 5.46 |
| 30 | 110 | 80 | 110 | 110 | 110.0 | 0.00 | 12348 | 1.73 |
| 35 | 135 | 135 | 135 | 130 | 134.5 | 1.58 | 32941.9 | 4.67 |
| 40 | 155 | 145 | 155 | 150 | 152.5 | 2.64 | 35581.5 | 5.11 |
| 46 | 175 | 175 | 175 | 175 | 175.0 | 0.00 | 16838.2 | 2.43 |
| 50 | 190 | 180 | 190 | 185 | 189.0 | 2.11 | 26626.8 | 3.93 |
| 55 | 205 | 200 | 205 | 200 | 200.5 | 1.58 | 32946.2 | 4.88 |
| 60 | 225 | 220 | 225 | 215 | 219.0 | 3.16 | 40177.4 | 6.03 |
| 65 | 240 | 240 | 240 | 230 | 237.5 | 3.54 | 49970.5 | 7.55 |
| 70 | 260 | 260 | 260 | 245 | 256.0 | 4.59 | 41242.1 | 6.24 |
| 73 | 265 | 265 | 265 | 260 | 263.5 | 2.42 | 39257 | 5.98 |
| 75 | 270 | 270 | 270 | 265 | 269.0 | 2.11 | 39253.8 | 6.03 |
| 80 | 280 | 280 | 280 | 275 | 279.0 | 2.11 | 30939.5 | 4.79 |
| 85 | 285 | 285 | 285 | 285 | 285.0 | 0.00 | 48237.9 | 7.54 |
| **Average** | **187.19** | **182.81** | **187.19** | **182.81** | **185.66** | **1.61** | **32115.89** | **4.77** |

Note that, the results obtained by VNS and ACO are not listed on the tables since their best scores (fitness) are completely same with the best known scores obtained by IS-PSO for all problem instances. IS-PSO is superior for 14 problem instances (shaded in Tables 9-12) in datasets 1, 2 and 3 compared to recently published GLS. It produces better results than ARPSO, GA and CGW for several problem instances, as well.

IS-PSO and StPSO are also compared in detail with VNS which is one of the algorithms producing the best known results for all problem instances for OP. Termination criteria of both algorithms are 100,000 evaluations. Average of the best results over all problem instances in datasets 1, 2, 3, and 4 are presented in Table 13. While fitness values are comparable, the fitness evaluation count is twice in VNS. So, our method achieves similar results by evaluating less candidate solutions. Our algorithms are slower than VNS based on the CPU time. This can be attributed to the use of SPV rule for all particles at each iteration in our algorithms and population-based nature of PSO.

**Table 10.** Performance of IS-PSO for dataset 2 (21 nodes and 11 instances ).

| | | ARPSO Fitness | GLS Fitness | IS-PSO Fitness | | | | IS-PSO Eval | IS-PSO CPU |
|---|---|---|---|---|---|---|---|---|---|
| Tmax | Optimum | Best | Best | Best | Worst | Mean | StDev | Mean | Mean |
| 15 | 120 | 120 | 120 | 120 | 120 | 120.0 | 0.00 | 1664 | 0.17 |
| 20 | 200 | 200 | 200 | 200 | 180 | 198.0 | 6.32 | 11838 | 1.25 |
| 23 | 210 | 200 | 210 | 210 | 210 | 210.0 | 0.00 | 15640 | 1.65 |
| 25 | 230 | 210 | 230 | 230 | 230 | 230.0 | 0.00 | 5032.3 | 0.53 |
| 27 | 230 | 230 | 220 | 230 | 220 | 227.0 | 4.83 | 3333.2 | 0.35 |
| 30 | 265 | 230 | 260 | 265 | 260 | 263.0 | 2.58 | 19161.3 | 2.02 |
| 32 | 300 | 260 | 300 | 300 | 290 | 298.0 | 4.22 | 20170.8 | 2.15 |
| 35 | 320 | 295 | 305 | 320 | 310 | 318.0 | 4.22 | 21387.9 | 2.29 |
| 38 | 360 | 340 | 360 | 360 | 350 | 356.0 | 4.59 | 40787.9 | 4.39 |
| 40 | 395 | 365 | 380 | 395 | 370 | 390.5 | 8.64 | 24208.2 | 2.62 |
| 45 | 450 | 440 | 450 | 450 | 440 | 449.0 | 3.16 | 11596.8 | 1.27 |
| **Average** | **280.00** | **262.73** | **275.91** | **280.00** | **270.91** | **278.14** | **3.51** | **15892.76** | **1.70** |

**Table 11**. Performance of IS-PSO for dataset 3 (33 nodes and 20 instances ).

| | | GLS Fitness | IS-PSO Fitness | | | | Eval | CPU |
|---|---|---|---|---|---|---|---|---|
| Tmax | Optimum | Best | Best | Worst | Mean | StDev | Mean | Mean |
| 15 | 170 | 170 | 170 | 170 | 170 | 0.00 | 7689.9 | 1.09 |
| 20 | 200 | 200 | 200 | 190 | 198 | 4.22 | 29934.2 | 4.30 |
| 25 | 260 | 250 | 260 | 250 | 254 | 5.16 | 10835.8 | 1.56 |
| 30 | 320 | 310 | 320 | 320 | 320 | 0.00 | 19080.9 | 2.77 |
| 35 | 390 | 390 | 390 | 390 | 390 | 0.00 | 41231.5 | 6.10 |
| 40 | 430 | 430 | 430 | 420 | 428 | 4.22 | 41256.8 | 6.18 |
| 45 | 470 | 470 | 470 | 460 | 467 | 4.83 | 28839.1 | 4.35 |
| 50 | 520 | 520 | 520 | 500 | 511 | 8.76 | 43167.8 | 6.56 |
| 55 | 550 | 540 | 550 | 540 | 549 | 3.16 | 29064.7 | 4.46 |
| 60 | 580 | 570 | 580 | 570 | 579 | 3.16 | 38885.3 | 6.00 |
| 65 | 610 | 610 | 610 | 600 | 609 | 3.16 | 32128.8 | 5.02 |
| 70 | 640 | 630 | 640 | 620 | 634 | 6.99 | 35642.3 | 5.57 |
| 75 | 670 | 670 | 670 | 650 | 665 | 8.50 | 36011.7 | 5.71 |
| 80 | 710 | 710 | 710 | 680 | 696 | 6.99 | 60554.3 | 9.60 |
| 85 | 740 | 740 | 740 | 710 | 722 | 7.89 | 28893.2 | 4.62 |
| 90 | 770 | 770 | 770 | 750 | 757 | 8.23 | 36602.9 | 5.89 |
| 95 | 790 | 790 | 790 | 780 | 789 | 3.16 | 44714.4 | 7.26 |
| 100 | 800 | 800 | 800 | 790 | 797 | 4.83 | 35628.8 | 5.84 |
| 105 | 800 | 800 | 800 | 800 | 800 | 0.00 | 15964.1 | 2.63 |
| 110 | 800 | 800 | 800 | 800 | 800 | 0.00 | 5829.8 | 0.95 |
| **Average** | **561.00** | **558.50** | **561.00** | **549.5** | **556.75** | **4.16** | **31097.82** | **4.82** |

**Table 12.** Performance of IS-PSO for dataset 4 (32 nodes and 16 instances, corrected ).

| | CGW Fitness | GA Fitness | IS-PSO Fitness | | | | Eval | CPU |
|---|---|---|---|---|---|---|---|---|
| Tmax | Best | Best | Best | Worst | Mean | StDev | Mean | Mean |
| 15 | 45 | 45 | 45 | 45 | 45 | 0.00 | 4955.7 | 0.68 |
| 20 | 65 | 65 | 65 | 65 | 65 | 0.00 | 18602.8 | 2.55 |
| 25 | 90 | 90 | 90 | 85 | 89.5 | 1.58 | 24112.7 | 3.35 |
| 30 | 110 | 110 | 110 | 110 | 110 | 0.00 | 14090 | 1.96 |
| 35 | 135 | 135 | 135 | 130 | 133 | 2.58 | 33114.4 | 4.67 |
| 40 | 155 | 155 | 155 | 150 | 154 | 2.11 | 26450.4 | 3.80 |
| 46 | 175 | 175 | 175 | 175 | 175 | 0.00 | 22738.7 | 3.29 |
| 50 | 190 | 190 | 190 | 190 | 190 | 0.00 | 47918.5 | 7.05 |
| 55 | 205 | 205 | 205 | 200 | 201 | 2.11 | 20949.3 | 3.11 |
| 60 | 220 | 225 | 225 | 215 | 219 | 3.16 | 42010.4 | 6.33 |
| 65 | 240 | 240 | 240 | 230 | 238 | 3.50 | 46330.2 | 6.96 |
| 70 | 260 | 260 | 260 | 255 | 257 | 2.58 | 49678.8 | 7.54 |
| 73 | 265 | 265 | 265 | 255 | 262 | 3.50 | 45513.8 | 6.89 |
| 75 | 275 | 270 | 275 | 265 | 270 | 4.08 | 36691.3 | 5.61 |
| 80 | 280 | 280 | 280 | 270 | 277 | 3.50 | 36597.2 | 5.65 |
| 85 | 285 | 285 | 285 | 280 | 283.5 | 2.42 | 26077 | 4.09 |
| **Average** | **187.19** | **187.19** | **187.5** | **182.5** | **185.56** | **1.94** | **30989.45** | **4.6** |

**Table 13.** Comparison of StPSO, IS-PSO and VNS results averaged over problems in datasets 1, 2, 3 and 4.

| | StPSO | | | ISPSO | | | VNS | | |
|---|---|---|---|---|---|---|---|---|---|
| | Fitness Best (Std.Dev.) | CPU Mean | Eval Mean | Fitness Best (Std.Dev.) | CPU Mean | Eval Mean | Fitness Best (Std.Dev.) | CPU Mean | Eval Mean |
| Dataset1 | 186.88 (2.13 ) | 4.72 | 31,812.16 | 187.19 (1.61 ) | 4.77 | 32,115.89 | 187.19 (2.73) | 0.92 | 62,624.36 |
| Dataset2 | 280.00 (3.78) | 2.12 | 19,839.67 | 280.00 (3.51) | 1.70 | 15,892.76 | 280.00 (2.49) | 1.09 | 38,681.41 |
| Dataset3 | 560.00 (4.43) | 5.22 | 33,438.54 | 561.00 (4.16) | 4.82 | 31,097.82 | 561.00 (6.04) | 0.88 | 52,096.07 |
| Dataset4 | 187.50 (1.94) | 4.60 | 30,989.45 | 187.50 (1.94) | 4.60 | 30,989.45 | 187.50 (2.82) | 0.91 | 60,933.15 |

# 7. Conclusion

In this paper, the StPSO algorithm is presented. The algorithm improves both exploration and exploitation mechanisms of PSO in order to solve well-known drawbacks of PSO such as premature convergence and insufficient intensification around the local optima. StPSO preserves the general flow of the PSO algorithm but the search strategy is temporarily modified for the pioneering-particles. An external local search is performed around each pioneering-particle using RVNS in order to improve the exploitation capability. To improve exploration mechanism and to prevent premature convergence, velocity of the pioneering-particle is updated randomly. So, the pioneering-particle continues its search journey from a different position in the following iterations.

StPSO and its two subversions (IS-PSO and DS-PSO) are experimented on the both continuous and discrete problems. Results for the continuous functions indicate that StPSO improves search ability of the standard PSO for both unimodal and multimodal problems. It achieves either the best or competitive solutions

compared to other similar PSO variants. Moreover, our approach achieves the best known solutions for several instances of the Orienteering Problem by evaluating less candidate solutions in the search space. All these results support our claim that StPSO is a promising general tool producing satisfactory results for both discrete and continuous problems.

# References

[1] J. Kennedy, R. Eberhart, "Particle swarm optimization", Proceedings of IEEE international conference on neural networks, Vol. 4 pp.1942—1948, 1995.

[2] A. Banks, J. Vincent, C. Anyakoha, "A review of particle swarm optimization. Part II: hybridization, combinatorial, multicriteria and constrained optimization, and indicative applications", International Journal of Natural Computing, Vol. 7(1), pp. 109-124, 2008.

[3] K.E. Parsopoulos, M.N. Vrahatis, "Recent approaches to global optimization problems through Particle Swarm Optimization", International Journal of Natural Computing, Vol. 1(2-3), pp. 235-306, 2002.

[4] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions" , IEEE Transaction on Evolutionary Computation, Vol. 10(3), pp. 281-295, 2006.

[5] E.S. Peer, F. Van den Bergh, A.P. Engelbrecht, "Using neighborhoods with the guaranteed convergence PSO", Swarm Intelligence Symposium. Indiana, Indianapolis, pp.235-242, 2003.

[6] S.K. Fan, E. Zahara, "A hybrid simplex search and particle swarm optimization for unconstrained optimization", European Journal of Operational Research, Vol. 181, pp. 527-548, 2007.

[7] F. Van den Berg, A. Engelbrecht, "A new locally convergent particle swarm optimizer", Proceeding of IEEE Conference on System, Man and Cybernetics, pp. 96-101, 2002.

[8] Z. Sevkli, E. Sevilgen, "Particle Swarm Optimization for the Orienteering Problem", Proceedings of International Symposium on Innovation in Intelligent Systems and Applications (INISTA'07). Istanbul, Turkiye, pp.185-190, 2007.

[9] Z. Sevkli, E. Sevilgen, "A Hybrid Particle Swarm Optimization Algorithm for Function Optimization", EvoWorkshop'08. LNCS Vol. 4974, pp. 585-595, 2008.

[10] B. Liu, L. Wang, Y-H. Jin, F. Tang, D.X. Huang, "Improved particle swarm optimization combined with chaos", Chaos Solutions & Fractals Vol.25, pp. 1261-1271, 2005.

[11] P.S. Shelokar, P. Siarry, V.K. Jayaraman, B.D. Kulkarni, "Particle swarm and ant colony algorithms hybridized for improved continuous optimization", Applied Mathematics and Computation, Vol.188, pp.129-142, 2007.

[12] M.F. Tasgetiren, Y.C. Liang, M. Sevkli, G. Gencyilmaz, "Particle Swarm Optimization Algorithm for Makespan and Total Flowtime Minimization in Permutation Flowshop Sequencing Problem", European Journal of Operational Research, Vol. 177(3), pp.1930-1947, 2007.

[13] Y. Shi, R.C. Eberhart, "A modified particle swarm optimizer", Proceedings of IEEE Congress on Evolutionary Computation, pp. 69-73,1998.

[14] M. Clerc, J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space", IEEE Transaction on Evolutionary Computation, Vol. 6(1), pp.58-73, 2002.

[15] T. Peram, K. Veeramachaneni, C.K. Mohan, "Fitness-distance-ratio based particle swarm optimization", Proceedings of Swarm Intelligence Symposium, pp.174-181, 2003.

[16] J. Kennedy, R. Mendes, "Population structure and particle swarm performance", Proceeding of IEEE Congress on Evolutionary Computation, Vol. 2, pp.1671-1676, 2002.

[17] J. Kennedy, "Small worlds and mega-minds: Effect of neighborhood topology on particle swarm performance", Proceeding of IEEE Congress on Evolutionary Computation, pp.1931-1938, 1999.

[18] P.N. Suganthan, "Particle swarm optimizer with neighborhood operator", Proceedings of IEEE Congress on Evolutionary Computation, Washington, DC, pp.1958-1962, 1999.

[19] K.E. Parsopoulos, M.N. Vrahatis, "UPSO-A unified particle swarm optimization scheme", Lecture Series on Computational Sciences, pp. 868-873, 2004.

[20] R. Mendes, J. Kennedy, J. Neves, "The fully informed particle swarm: Simpler, maybe better", IEEE Transaction on Evolutionary Computation, Vol. 8, pp. 204-210, 2004.

[21] C. Ai-ling, Y. Gen-ke, W. Zhi-ming, "Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem", Journal of Zhejiang University Science A, Vol. 7(4), pp. 607-614, 2006.

[22] S.C. Esquivel, C.A. Coello Coello, "On the Use of Particle Swarm Optimization with Multimodal Functions", Proceeding of IEEE Congress on Evolutionary Computation (CEC'2003), Vol. 2, pp. 1130-1136, 2003.

[23] Y.T. Kaoç, E.T Zahara, "A hybrid genetic algorithm and particle swarm optimization for multimodal functions", Applied Soft Computing, Vol.8 (2), pp. 849-857, 2008.

[24] C. Zhan, J. Ning, S. Lu, D. Ouyang, T. Ding, "A novel hybrid differential evaluation and particle swarm optimization algorithm for unconstrained optimization", Operations Research Letters, Vol.37, pp. 117-122, 2009.

[25] H. Liu, A. Abraham, "Fuzzy adaptive turbulent particle swarm optimization", Proceedings of fifth International Conference on Hybrid Intelligent Systems (HIS'05), Rio de Janeiro, Brazil, pp. 445-450, 2005.

[26] H. Liu, A. Abraham, "An Hybrid Fuzzy Variable Neighborhood Particle Swarm Optimization Algorithm for Solving Quadratic Assignment Problems", Journal Of Universal Computer Science, Vol.13, pp. 1032-1054, 2007.

[27] N. Mladenovic, P. Hansen, "Variable Neighborhood Search", Computers & Operations Research, Vol.24(11), pp.1097-1100, 1997.

[28] P. Hansen, N. Mladenovic, "Variable neighborhood search: principles and applications", European Journal of Operation. Research, Vol.130, pp. 449-467, 2001.

[29] M. Clerc, "Swissknife PSO", http://clerc.maurice.free.fr/pso/e.

[30] B. Golden, L. Levy, R. Vohra, "The Orienteering Problem", Naval Res. Logist, Vol.34(3), pp. 307—318, 1987.

[31] A.C. Leifer, M.B. Rosenwein, "Strong Linear Programming relaxations for orienteering problem", European Journal of Operation. Research, Vol.73, pp.517-523, 1994.

[32] F.M. Tasgetiren, A.E. Smith, "A genetic algorithm for the orienteering problem", Proceeding of the Congress on Evolutionary Computation, San Diego, CA, Vol.2, pp. 910- 915 ,2000.

[33] Y-C. Liang, S. Kulturel-Konak, A.E. Smith, "Meta Heuristic for the Orienteering Problem", Proceeding of the Congress on Evolutionary Computation (CEC'02). , Hawaii, pp. 384-389, 2002.

[34] Z. Sevkli, E. Sevilgen, "Variable Neighborhood Search for the Orienteering Problem", LNCS, Vol. 4263, pp.134-143, 2006.

[35] T. Tsiligirides, "Heuristic methods applied to orienteering", Journal of Operational Research Society, Vol.35 (9), pp. 797-809, 1984.

[36] I-M. Chao, B.L. Golden, EA. Wasil, "A fast and effective heuristic for the orienteering problem", European Journal of Operation Research, Vol. 88(3), pp.475-489, 1996.

[37] H. Dallard, S.S. Lam, S. Kulturel-Konak, "Solving the Orienteering Problem Using Attractive and Repulsive Particle Swarm Optimization", IEEE International Conference on Information Reuse and Integration, pp. 12 – 17, 2007.

[38] P. Vansteenwegen, W. Souffriau, G.V. Berghe, D.V. Oudheusden, "A guided local search metaheuristic for team orienteering problem", European Journal of Operational Research, Vol. 196, pp 118-127, 2009.