2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Leaders and Speed Constraint Multi-Objective Particle Swarm Optimization

Farid Bourennani
Department of Electrical, Computer and Software Engineering
University of Ontario Institute of Technology
Oshawa, Ontario, Canada
farid.bourennani@uoit.ca

Shahryar Rahnamayan
Department of Electrical, Computer and Software Engineering
University of Ontario Institute of Technology
Oshawa, Ontario, Canada

Greg F. Naterer
Faculty of Engineering and Applied Science
Memorial University of Newfoundland
St. John's, Newfoundland, Canada

Abstract— **The particle swarm optimization (PSO) algorithm has been very successful in single objective optimization as well as in multi-objective (MO) optimization. However, the selection of representative leaders in MO space is a challenging task. Most previous MO-based PSOs used exclusively the concept of non-dominance to select leaders which might slow down the search process if the selected leaders are concentrated in a specific region of the objective space. In this paper, a new restriction mechanism is added to non-dominance in order to select leaders in more representative (distributed) way. The proposed algorithm is named leaders and speed constrained multi-objective PSO (LSMPSO) which is an extended version of SMPSO. The convergence speed of LSMPSO is compared to state-of-the-art metaheuristics, namely, NSGA-II, SPEA2, GDE3, SMPSO, AbYSS, MOCell, and MOEA/D. The ZDT and DTLZ family problems are utilized for the comparisons. The proposed LSMPSO algorithm outperformed the other algorithms in terms of convergence speed.**

Keywords — Multi-Objective Optimization, Particle Swarm Optimization, PSO, Metaheuristics, Evolutionary Algorithms.

I. INTRODUCTION

PSO is a nature inspired metaheuristics simulating the social behavior of a flock of birds. Originally, it was proposed by Kennedy and Eberhart [1] in 1995. Since then, over 30 multi-objective variants of PSO have been proposed [2]. However, the most salient variant is SMPSO [2]. SMPSO is an extension of OMOPSO [3]. Its main difference with respect to OMOPSO is the incorporation of a mechanism for velocity limitation and introducing a polynomial mutation operator.

SMPSO has shown a high accuracy for MO optimization problems as in Ref. [2] and a high convergence speed as in Ref. [4]. In PSOs, the movement of candidate solutions, called particles, is influenced by the leaders. However, our observations show that if the leaders are not appropriately selected, the swarms might be attracted by a concentration of leaders in a certain region of the objective space. Consequently, the entire particles movement will prematurely converge towards a specific objective space region where these groups of leaders are located. Additional computational overhead would be required to find the entire Pareto front.

In order to maintain an appropriate distribution of the particles, especially during the early stages of the search process (i.e., exploration), it is proposed to select a more representative group of leaders. Let us call this group deputes. This number of deputes is restricted and they should be more representative of the swarms by being more diverse (well-distributed). The selection of leaders is performed by a non-dominance concept. Then, it is proposed to select deputes using the sum of weighted ratios (SWR).

In this paper, the convergence speed of the proposed algorithm, LSMPSO, is compared to seven state-of-the-art MO metaheuristics including its parent algorithm SMPSO. For comparison purposes, two well-known problem families, the Zitzler-Deb-Thiele (ZDT) [5] and the Deb-Thiele-Laumanns-Zitzler (DTLZ) [6], are used for a total of 12 benchmark functions. The hypervolume (HV) indicator [7] is employed as a stopping criterion. When the HV of a solution set is equal or above 98% of the true Pareto front, then the front can be considered an accurate approximation of the real Pareto front. Consequently, the stopping condition is fixed to HV $\geq$ 98% or when reaching a maximum number of function calls (i.e. 10^6). These termination criteria are commonly used in the literature (e.g. Ref. [8]). Every MO optimization algorithm was independently executed 100 times per problem to minimize the effect of the stochastic nature of the metaheuristics on the results. The metaheuristic requiring the lower number of

This work was supported in part by an Ontario Graduate Scholarship (OGS).

function calls to find an accurate approximation of the Pareto front is considered as the fastest algorithm. Also, the hit (success) rate should be of 100% otherwise the algorithm is disqualified for that specific problem. The hit rate means how many times the algorithm was able to find (approximate with HV $\geq$ 98%) the real Pareto front before meeting the predefined maximum number of function calls.

The reminder of this paper is organized as follows. Section II describes the proposed algorithm LSMPSO. Section III provides details about the experiment settings such as the conducted comparisons with the selected optimization algorithms, the studied problems and others. Section IV presents the results. Finally, Section V concludes the paper.

II. LITTERATURE REVIEW

This section describes the most recent PSO-based multi-objective algorithms which focused on the leaders selection. By recent studies, it is meant after the development of OMOPSO, namely the parent algorithm of SMPSO. Earlier surveys about MO PSO can be found for example in Ref. [9].

In Ref. [10], it is proposed to hybridize PSO with differential evolution (DE). PSO is served to accelerate the convergence speed whereas DE mutation properties are used for diversity maintenance. The selection of leaders is done using the non-dominance scheme. Then, the centroid point of leaders is calculated and a roulette wheel mechanism is used to select leaders. The probability of a leader to be selected is proportional to the distance from the centroid; the further a leader is from the centroid point, the higher are the chances to be selected to encourage diversity. The proposed algorithm was compared with OMOPSO, SMPSO, NSGA-II and DEMO using the ZDT problems. Their proposed algorithm converged faster for two problems out of five, namely, ZDT1 and ZDT2.

In Ref. [11], five leader selection strategies have been incorporated into a multi-objective PSO namely the random strategy, the sigma strategy, the nearest strategy, the grid strategy, and the non-dominated strategy. PSO with different strategies has been tested on the WFG family problem. The results showed that the non-dominance strategy generates the highest accuracy.

Other leader selection methods have been investigated for many-objective optimization (> 3 objectives) problems such as in Ref. [12].

III. DESCRIPTION OF LSMPSO

In order to understand better the purpose of the leaders selection process, consider the relationship among leaders and particles and how leaders influence particles in their movements.

A. Velocity

In PSO algorithms, a candidate solution to an optimization problem is called a *particle.* The entire population of particles is called a *swarm.* In single objective PSO, a particle $\vec{x}_i$ is updated at a generation *t* as follows.

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \qquad (1)$$

where the $\vec{v}_i(t)$ is the particle's velocity is calculated as follows,

$$\vec{v}_i(t) = w.\vec{v}_i(t-1) + C_1.r_1.\left(\vec{x}_{p_i} - \vec{x}_i\right) + C_2.r_2.\left(\vec{x}_{G_i} - \vec{x}_i\right) \quad (2)$$

where $\vec{x}_{p_i}$ is the (local) best found candidate solution by the particle $\vec{x}_i$, $\vec{x}_{G_i}$ is the (global) best found particle in the entire swarm called leader, w is the particle inertia weight, which represents a trade-off between the global and local experiences, r_1 and r_2 are random variables in the range [0,1], and C_1 and C_2 are learning factors towards respectively the particle's personal success and its neighbor's success.

B. Description of the proposed approach

The main challenge of extending the PSO algorithm to MO space resides in the generalization of the concept of leader [3]. The most common approach consists in considering the non-dominated solutions as leaders. If the number of leaders exceeds the maximum size of an archive, then the crowding factor is used as a second discriminant as in the NSGA-II archive [13]. As shown in Figure 1, the non-dominance discriminant can affect the distribution of the particles at early exploration stages. Sometimes, it is possible to have a concentration of leaders in a certain region of the function space; therefore, the entire swarm movement will predominantly affected by these leaders. The particles will be covering a certain portion of the Pareto front rather than the entire Pareto front (PF). Consequently, additional computational overhead would be required to find the remaining portion of the PF or worst the PF won't be found because no leaders are located in the missing part of the PF.

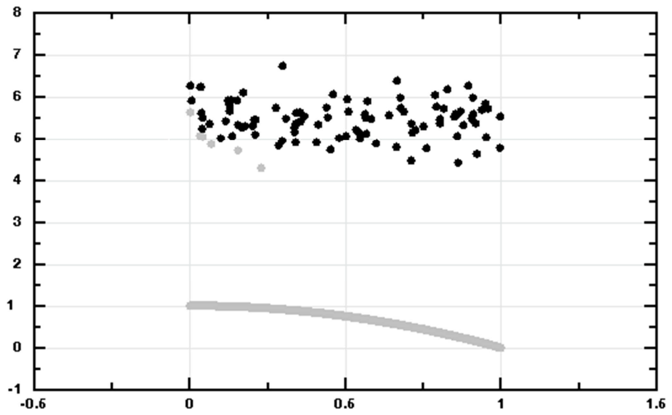

Figure 1: First Iteration for the ZDT 2 Problem - grey particles are leaders

In order to maintain an appropriate distribution of the swarms, this paper proposes to use only p percent of the most representative leaders during the entire search process. Restricting the number of leaders discourages leaders concentrated in the same function space area to influence the swarm movement. In order to select a "good" leaders representation, it is proposed to use the sum of weighted ratios (SWR) explained in the next sub-section.

C. Representative Non-Dominance Concept

The SMPSO archive is the same archive used by NSGA-II which is also commonly used by other MO metaheuristics. However, the proposed archive selects a more representative number of leaders using SWR for the calculation of particle velocity. In SMPSO, at every iteration, two leaders are selected randomly. Then, the less crowded between the two leaders is selected for the calculation of the particle velocity. For the proposed LSMPSO algorithm, only the speed computation function will be modified by restricting the leaders that can influence the swarm movement. Only p% of the most representative leaders, let us call them deputy leaders, are selected first as described in the next sub-section. Then at every iteration, a deputy leader is selected randomly for the calculation of the swarm velocity.

For better clarity, the pseudocode of SMPSO is given bellow.

SMPSO PSEUDOCODE

```
1: initializeSwarm()
2: initializeLeadersArchive()
3: generation = 0
4: while generation < maxGenerations do
5:    computeSpeed() // MODIFIED
6:    updatePosition() // Eq. 1
7:    mutation() // Turbulence
8:    evaluation()
9:    updateLeadersArchive()
10:   updateParticlesMemory()
11:   generation ++
12: end while
13: returnLeadersArchive()
```

D. Selection Scheme of Deputy Leaders

The selection of deputy leaders is based on the sum of weighted ratios (SWR). The fitness value of every objective (f_i) is converted into a ratio (r_i) between [0,1] as follows:

$$r_i = \frac{f_i - min_i}{max_i - min_i}$$

where min_i and max_i are respectively the minimum and maximum values of the objective *i*.

Then, the scaled fitness F_x of the particle *x* is calculated by summing all $r_i s$:

$$F_x = \sum_1^N r_{i_x}, i = 1,2,\dots,N$$

where N is the number of objectives, and *x* is the current particle for which F_x is calculated.

Usually, for a lower F_x, the closer is the point to the Pareto front. The lower values are selected rather than the larger among the leaders to promote a good distribution of solutions. The Fitness value for the extreme points of the front is set usually to 1.0. Therefore, they are always kept among deputy leaders.

At every iteration before the calculation of the particles' velocity, the SWR of the leaders is calculated. Then, the deputy leaders are determined as shown in the following deputy selection pseudocode.

DEPUTES SELECTION PSEUDOCODE

```
Begin
maxDeputies=%p * MaxArchiveSize;
i=0;
If (ArchiveSize == 1) // There only one leader
    Deputies=Leaders
Else if (ArchiveSize ≤ maxDeputies)
    For each Leader
        If (Leader.F < 1.0 Or Leader.Crowding =
        Infinity and Leader.F ≤ 1.0 )
            Deputies.add(Leader)
        EndIf
    endFor
Else // ArchiveSize > maxDeputies
    Sort (Leaders) // ascending sorting based on F
    While Deputies not full and still Leaders
        If (Leader.F < 1.0 Or Leader.Crowding =
        Infinity and Leader.F ≤ 1.0 )
            Deputies.add(Leader)
        EndIf
    EndWhile
endIF
End
```

To clarify the proposed approach, assume there are eight leaders as described in TABLE 1.

TABLE 1: EXAMPLE OF EIGHT LEADERS IN TWO-OBJECTIVE SPACE

Objective 1	Objective 2	F
2.2109	15.989	**1.0**
3.7248	13.997	0.9681
5.5071	9.3777	**0.7640**
9.0128	6.2587	0.7997
10.749	4.4470	**0.7978**
12.0744	3.7935	0.8498
13.6567	3.2865	0.9319
15.4774	2.3427	**1.0**

The same leaders are plotted in FIGURE 2. Let us assume that only four deputies can be selected from the leaders. See TABLE 1. To clarify the proposed approach, assume there are eight leaders as described in TABLE 1.

TABLE 1 values in bold show two leaders having the lowest fitness value and two extreme leaders are selected. These deputies are represented in the graph with plain points whereas the other leaders are represented with empty points. It can be seen that the selected deputies represent well the full spectrum of the leaders. In addition, it can be seen that there is no concentration of leaders in a specific region of the objective space which might influence the particles movement.

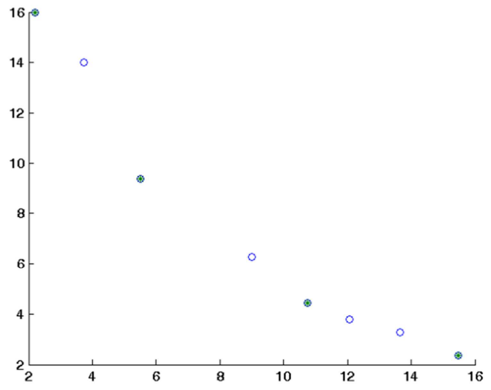

FIGURE 2: EXAMPLE OF FOUR SELECTED DEPUTIES BASED ON SWR

IV. EXPERIMENT SETTINGS

A. Benchmark Problems

As shown in Table 2, two well-known families of problems are used for comparison purposes: the ZDT [5] and the DTLZ [7] family problems. These two families are the most commonly used families. They are composed of different PF geometries, namely, convex, concave, disconnected, linear, and non-uniformly spaced.

B. Comparison with MO Optimization Algorithms

The proposed LSMPSO is compared with seven state-of-the-art MOO metaheuristics which are described in this subsection. The implmentation of thse state-of-the-art algorithms is available in the jMetal [14] multi-objective optimization framework which has been used for conducting all the experiments in this paper.

The Non-dominated Sorting Genetic Algorithm (NSGA-II) was proposed by Deb et al. [15] in 2002. This genetic algorithm consists of generating new populations from the original population by the use of classical genetic operators such as selection, crossover, and mutation. The individuals of the two populations are sorted according to their ranking. Then, the best solutions are recombined for the generation of the next population. In the case of having solutions with the same rank, a density estimation (crowding distance) is calculated with regards to the surrounding solutions for the selection of the most promising solutions.

The Strength Pareto Evolutionary Algorithm (SPEA2) was proposed by Zitzler et al. [16] in 2002. In this MOEA, every candidate solution has a fitness value which equals the sum of its strength raw fitness (solutions that dominates it) plus a density estimation. SPEA2 uses the selection, crossover, and mutation operators for generating an archive of individuals. The non-dominated solutions of both the original population and the archive are copied into a new population. In case the number of non-dominated solutions is superior to the population size, a truncation operator is used by calculating the distances among solutions. The most similar solutions are removed.

Table 2: Utilized Bi-Objective Problems in comparison study

Problem	Number Variables	Geometries
ZDT1	30	Convex
ZDT2	30	Concave
ZDT3	30	Convex, disconnected
ZDT4	10	Convex
ZDT6	10	Concave, non-uniformly spaced
DTLZ1	7	Linear
DTLZ2	12	Concave
DTLZ3	12	Concave
DTLZ4	12	Concave
DTLZ5	12	Concave
DTLZ6	12	Concave
DTLZ7	22	Disconnected

The Speed Constrained Particle Swarm Optimization (SMPSO) algorithm, which is the parent algorithm of LSMPSO, was proposed by Nebro et al. [17] in 2009. It is a particle swarm optimization algorithm for solving MOO problems. This approach is based on OMOPSO [3], whose main features are the use of the crowding distance concept adopted by NSGA-II for filtering leader solutions that are stored in an archive, the use of mutation operators for swarm speed convergence acceleration, and the use of ϵ-Dominace when generating new candidate solutions. Its main difference with respect to OMOPSO is that SMPSO incorporates a mechanism for velocity limitation and introduces a polynomial mutation operator.

The third version of the Generalized Differential Evolution algorithm (GDE3) was proposed by Kukkonen and Lampinen [18]. GDE3 is an improved version of the GDE algorithm [19], which was originally proposed in 2005. It starts with a random solution population. In every iteration, a new offspring population is generated using the differential evolution operator. Both populations are combined; then, the size of the population is reduced using non-dominated sorting and a pruning algorithm for diversity preservation as in NSGA-II. However, the GDE3 pruning algorithm modifies the NSGA-II crowding distance in order to solve some GDE3 drawbacks when dealing with problems with more than two objectives.

The cellular genetic algorithm (MOCell) was introduced by Nebro et al. [20] in 2006. Being a genetic algorithm, it uses selection, crossover, and mutation operators. Similar to many multi-objective metaheuristics, it includes an external archive for storing the non-dominated solutions discovered so far. This archive is bounded by using NSGA-II's crowding distance in order to maintain diversity in the Pareto front. The selection is achieved by selecting a solution from the neighborhood of the current solution (called cell in cGAs) and another solution selected randomly from the archive. Then, the genetic crossover and mutation operators are applied for generating a new offspring which is compared to the current offspring. If the offspring is better, it replaces the current one. Otherwise, if both solutions are non-dominated, then the worst solution in the neighborhood is replaced by the current one and inserted into the archive.

AbYSS was introduced by Nebro et al. [21] in 2008; it is a multi-objective version of a *scatter search.* It has an external archive similar to MoCell. AbYSS uses evolutionary operators such as polynomial mutation, binary crossover and solution combination.

The Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D) [22] was proposed in 2007 and it consists of decomposing a MOO problem into scalar sub-problems which are optimized in parallel. Each sub-problem is transformed into a scalar aggregation problem and optimized using only neighborhood information. These neighborhood relations are determined by the calculation of distances among coefficient vectors.

C. Parameters Settings

The parameter settings are the same for every MO metaheuristic. These parameter settings were taken from Ref. [9].

TABLE 3: PARAMETERIZATION

NSGA-II	
Population size	100 Individuals
Selection of parents	Binary tournament + binary tournament
Recombination	Simulated binary, pc = 0.9
Mutation	Polynomial, pm = 1.0/L
SPEA2	
Population size	100 Individuals
Selection of parents	Binary tournament + binary tournament
Recombination	Simulated binary, pc = 0.9
Mutation	Polynomial, pm = 1.0/L
MOCell	
Population size	100 individuals (10 × 10)
Neighborhood	1-hop neighbors (8 surrounding solutions)
Selection of parents	Binary tournament + binary tournament
Recombination	Simulated binary, pc = 0.9
Mutation	Polynomial, p_m = 1.0/L
Archive size	100 individuals
LSMPSO / SMPSO	
Particles	100 particles
Mutation	Polynomial
Leaders size	100 individuals
GDE3	
Population size	100 individuals
Recombination	Differential evolution, CR = 0.1, F = 0.5
MOEA/D	
Population size	100 individuals
Recombination	Differential evolution,
Mutation	Polynomial
AbYSS	
Population size	100 individuals
Reference set size	10 + 10
Recombination	Simulated binary, pc = 1.0
Mutation	Polynomial, p_m = 1.0/L
Archive size	100 individuals

NSGA-II, SPEA2, MOCell, AbYSS and GDE3 and MOEA/D have a population size of 100. In the same manner, LSMPSO and SMPSO have a configuration of 100 particles. The metaheuristics having an archive such as NSGA-II, SPEA2 and others have also a maximum size of 100.

In regards, to the number of deputes selected for LSMPSO, it is fixed to 10. In other words, only 10% of the maximum archive size is used as deputes leaders.

The configuration parameters of the algorithms are shown in TABLE 3.

D. Performance Measure

A high quality set of solutions, in a multi-objective optimization context, should be accurate and diverse. Accuracy means the solutions should be as close as possible to the Pareto Front. Diversity means the solution should be well-distributed to cover all of the Pareto Front. A popular quality indicator that takes into consideration both the accuracy of a solution set and its diversity is the hypervolume (HV) indicator [8]. The HV is obtained by computing the volume of the non-dominated set of solutions Q for MOO minimization problems. For every solution $\boldsymbol{i} \in Q$, a hypercube v_i is generated with a reference point W and the solution i as its diagonal corner. The reference point W can be generated by building a vector of worst possible objective function values. Then, the HV is computed as a union of all in the hypercube as follows:

$$HV = volume\left(\bigcup_{i=1}^{|Q|} v_i\right)$$

For a lower HV value, the correspondent solutions are better because they are more precise and diverse. Given that the Pareto fronts of the problems used in this study are known beforehand, the algorithms are executed until sufficient approximation of the real Pareto fronts (HV ≥ 98%) as shown in Figure 3.

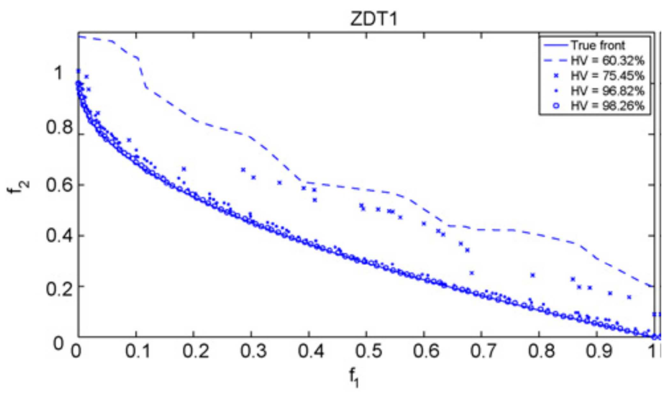

Figure 3: PARETO FRONTS WITH DIFFERENT HYPERVOLUME VALUES OBTAINED FOR THE ZDT1 PROBLEM.

Finding the Pareto front might not always be possible for some algorithms depending on the problem's complexity. In other words, some algorithms might perform well with some specific problems, but not able to find the Pareto front for other types of problems. Or it might take too long to produce the Pareto front. Consequently, another stopping condition is added by allowing every metaheuristic to perform at most 10^6 function calls.

V. Results

Due the stochastic nature of metaheuristics, every algorithm was run 100 times independently. The results are reported in Table 4. The dark grey area shows the fastest algorithm, while the lighter grey shows the second fastest algorithm.

The Wilcoxon statistical procedure is conducted based on Ref. [23] to present results at a 0.05 significance level. However, whenever the statistical test did not pass between the two fastest algorithms, both of them were ranked first. For example, for the problems ZDT3, ZDT4, ZDT6 and DTLZ1, there was not a significant statistical difference between the two fastest algorithms. So the two fastest MO metaheuristics were ranked first for these specific cases.

The hit rates of the compared multi-objective optimization algorithms are shown in Table 5. A hit of 100% means that the multi-objective optimization algorithm was able to find an accurate approximation of the Pareto front (HV ≥ 98%) for every single run (100/100). A lower rate than 100 automatically disqualifies the algorithm from the comparison process for that specific optimization problem.

It can be seen in Table 4 that the proposed algorithm LSMPSO was the fastest algorithm among all the compared algorithms for 10/12 problems. The LSMPSO is the fastest algorithm for all ZDT and DTLZ problems except for the DTLZ4 and DTLZ6 problems. In addition, it achieved a hit rate of 100% for all the problems.

When LSMPSO was compared to its parent algorithm, SMPSO, it is found that LSMPSO improved the convergence speed of SMPSO for 7 problems out of 12 namely for ZDT1, ZDT2, ZDT3, DTLZ2, DTLZ3, DTLZ5 and DTLZ7. LSMPSO had the same performance as SMPSO for the three problems ZDT4, ZDT6 and DTLZ1. LSMPSO presents a lower performance than SMPSO for only two problems DTLZ4 and DTLZ6. However, SMPSO was not the fastest algorithm for both DTLZ4 and DTLZ6 problems. Overall, LSMPSO improved SMPSO 58.66% of the time, had the same fastest performance for 25%, and it had a lower performance that SMPSO for 16.33% of the time. It can be concluded that LSMPSO improved the SMPSO algorithm to become the fastest one while keeping a high accuracy in a consistent manner.

The second performance was achieved by SMPSO. It was the fastest algorithm for three problems, namely ZDT4, ZDT6 and DTLZ1 and the second fastest algorithm for six problems, namely ZDT1, ZDT2, DTLZ3, DTLZ4, DTLZ6 and DTLZ7. SMPSO had a hit rate of 100% for all problems. So, overall SMPSO offered good performance.

The third performance was achieved by GDE3. It was the fastest algorithm only for two problems, ZDT3 and DTLZ6. And GDE3 had a hite rate of 100% for all the problems. So, overall it had good performance except for the problems ZDT2, ZDT4 and DTLZ2 where GDE3 had serious difficulties as compared to the other fastest algorithms.

Table 4: Median and interquartile range (IQR) of the number of evaluations for reaching the Pareto front (HV ≥ 98%). The dark grey area shows the fastest algorithm, while the lighter grey area shows the second fastest algorithm.

	LSMPSO	SPEA2	NSGAII	GDE3	SMPSO	AbYSS	MOCell	MOEAD
ZDT1	$7.40e+03_{2.3e+03}$	$1.61e+04_{9.0e+02}$	$1.41e+04_{8.0e+02}$	$9.60e+03_{5.0e+02}$	$9.55e+03_{2.9e+03}$	$1.31e+04_{1.4e+03}$	$1.30e+04_{1.0e+03}$	$5.24e+04_{6.1e+03}$
ZDT2	$8.05e+03_{2.8e+03}$	$2.46e+04_{1.7e+03}$	$2.22e+04_{1.2e+03}$	$1.12e+04_{6.0e+02}$	$8.90e+03_{2.8e+03}$	$1.71e+04_{2.7e+03}$	$1.53e+04_{6.0e+03}$	$7.03e+04_{9.1e+03}$
ZDT3	$1.05e+04_{3.0e+03}$	$1.54e+04_{1.2e+03}$	$1.27e+04_{1.2e+03}$	$1.02e+04_{5.0e+02}$	$1.33e+04_{4.4e+03}$	$1.24e+04_{2.5e+03}$	$1.28e+04_{1.2e+03}$	$5.60e+04_{6.4e+03}$
ZDT4	$4.95e+03_{1.6e+03}$	$2.60e+04_{4.8e+03}$	$2.19e+04_{5.1e+03}$	$1.63e+04_{9.0e+02}$	$4.95e+03_{1.3e+03}$	$2.10e+04_{9.7e+03}$	$1.67e+04_{5.2e+03}$	$7.06e+04_{1.3e+04}$
ZDT6	$3.95e+03_{1.8e+03}$	$3.33e+04_{1.2e+03}$	$2.84e+04_{1.3e+03}$	$4.50e+03_{7.0e+02}$	$4.40e+03_{2.2e+03}$	$1.54e+04_{1.2e+03}$	$2.16e+04_{1.2e+03}$	$2.00e+04_{8.1e+03}$
DTLZ1	$5.70e+03_{2.0e+03}$	$2.53e+04_{7.1e+03}$	$2.56e+04_{8.4e+03}$	$1.03e+04_{6.0e+02}$	$5.90e+03_{2.6e+03}$	$2.64e+04_{1.4e+04}$	$1.99e+04_{6.2e+03}$	$2.71e+04_{1.5e+04}$
DTLZ2	$2.80e+03_{7.5e+02}$	$7.20e+03_{8.0e+02}$	$6.70e+03_{8.5e+02}$	$6.40e+03_{3.5e+02}$	$5.30e+03_{2.0e+03}$	$4.69e+03_{8.0e+02}$	$5.90e+03_{9.0e+02}$	$9.70e+03_{1.0e+03}$
DTLZ3	$8.55e+03_{1.4e+04}$	$1.01e+05_{4.3e+04}$	$1.09e+05_{4.4e+04}$	$2.24e+04_{1.5e+03}$	$1.15e+04_{2.9e+04}$	$1.23e+05_{6.2e+04}$	$6.69e+04_{2.2e+04}$	$1.17e+05_{5.9e+04}$
DTLZ4	$1.01e+04_{3.0e+03}$	$7.80e+03_{2.4e+05}$	$7.30e+03_{1.2e+03}$	$8.05e+03_{1.3e+03}$	$5.75e+03_{1.5e+03}$	$4.80e+03_{8.1e+02}$	$1.00e+06_{9.9e+05}$	$1.46e+04_{2.1e+03}$
DTLZ5	$2.85e+03_{8.5e+02}$	$7.40e+03_{8.0e+02}$	$6.70e+03_{8.5e+02}$	$6.30e+03_{3.0e+02}$	$5.15e+03_{1.9e+03}$	$4.56e+03_{8.6e+02}$	$5.70e+03_{9.0e+02}$	$9.80e+03_{1.0e+03}$
DTLZ6	$2.02e+04_{9.6e+03}$	$2.50e+05_{2.2e+05}$	$2.50e+05_{2.2e+05}$	$3.70e+03_{2.0e+02}$	$9.15e+03_{3.9e+03}$	$1.00e+06_{2.0e+01}$	$1.00e+06_{0.0e+00}$	$9.50e+03_{1.8e+03}$
DTLZ7	$5.75e+03_{1.5e+03}$	$1.69e+04_{1.1e+03}$	$1.38e+04_{1.0e+03}$	$8.50e+03_{5.0e+02}$	$6.80e+03_{2.3e+03}$	$1.07e+04_{1.8e+03}$	$1.17e+04_{1.6e+03}$	$4.58e+04_{2.8e+05}$

Table 5: Average Hit Rate of the Compared MO Optimzation Algorithms

	LSMPSO	SPEA2	NSGAII	GDE3	SMPSO	AbYSS	MOCell	MOEAD
ZDT1	100	100	100	100	100	100	100	100
ZDT2	100	100	100	100	100	100	100	100
ZDT3	100	100	100	100	100	100	100	100
ZDT4	100	100	100	100	100	100	100	100
ZDT6	100	100	100	100	100	100	100	100
DTLZ1	100	100	100	100	100	100	100	100
DTLZ2	100	100	100	100	100	100	100	100
DTLZ3	100	100	100	100	100	100	100	100
DTLZ4	100	72	82	100	100	100	44	100
DTLZ5	100	100	100	100	100	100	100	100
DTLZ6	100	48	34	100	100	1	9	100
DTLZ7	100	100	100	100	100	82	88	76

It is worth to mention that only LSMPSO, SMPSO and GDE3 achieved a hit rate of 100% for all the compared problems. This demonstrates their consistency in the results as well as the robustness of these three algorithms

AbYSS comes in fourth position. AbYSS has been the fastest optimization algorithm for the DTLZ4 problem and the second fastest for two problems, namely, DTLZ2 and DTLZ5. However, AbYSS had difficulties with DTLZ6 and especially with DTLZ7 by having a hit rate equal to only 1%.

In order to rank the remaining multi-objective optimization algorithms, the boxplots [24] are used. The boxplots allow a graphical analysis of the found solutions using five number summaries: the minimum, lower quartile (Q1), median (Q2), upper quartile (Q3), and the maximum. Also, boxplots indicate solutions as outliers, i.e., a solution that is numerically distant from the rest of the data. Also, the boxplots allow one to see the consistency of the results by examining graphically the IQR.

The boxplot results are compiled in Figure 4 for ZDT family problems and in Figure 5 for DTLZ family problems. It can be seen from the boxplots that MOCell achieves usually the fourth performance most of the time, followed by NSGA-II, followed by SPEA2, and followed finally by MOEA/D.

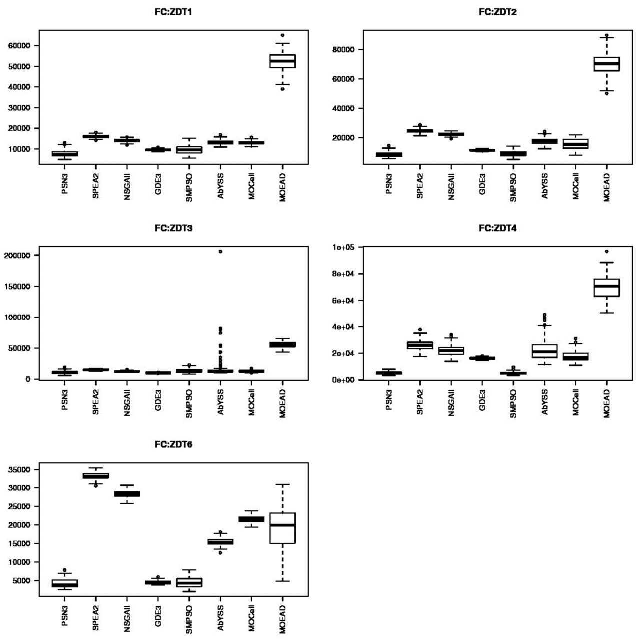

Figure 4: Boxplots for ZDT problems

MOEA/D achieved a low hit rate only for one problem whereas the NSGA-II and SPEA2 algorithms achieved a low rate for two problems and MoCell achieved a low rate for three problems.

By examining the boxplots, another interesting observation is that the proposed algorithm, LSMPSO, has consistent performance most of that time better that its parent algorithm SMPSO. However, GDE3 achieved the highest consistency in the results. It might be interesting to analyze why GDE3 has a higher consistency in comparison with the other algorithms. By this way LSMPSO could be further enhanced not only to be the fastest one but with more consistent results.

VI. Conclusions

This paper proposed a new multi-objective version of particle swarm optimization in order to accelerate its convergence speed. The proposed algorithm called LSMPSO incorporated a restriction mechanism on the leaders that can be used for the particle velocity calculation. Only the p% most representative leaders called deputies are selected, using the SWR algorithm, for the particles velocity calculation. The proposed algorithm was compared to seven state-of-the-art metaheuristics, namely, NSGA-II, SPEA2, GDE3, SMPSO, AbYSS, MOCell and MOEA/D using the ZDT and DTLZ bi-objective family problems. The convergence speed of these algorithms was compared by counting the number of function calls required to find an accurate approximate of the Pareto front (HV≥98%) with a maximum of 10^6 function calls allowed.

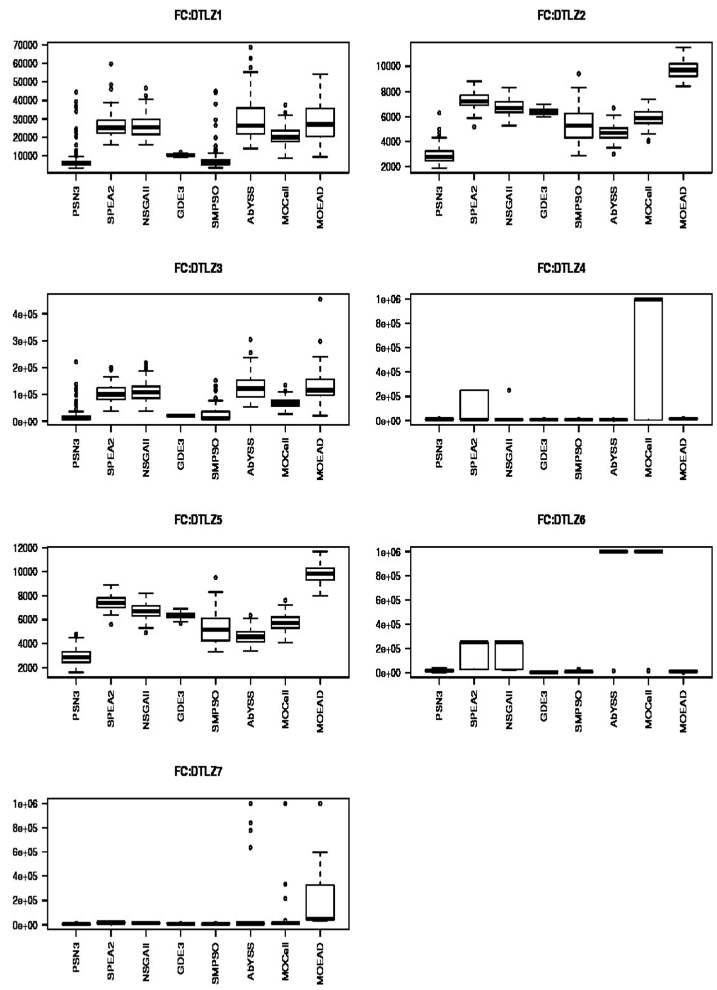

Figure 5: Boxplots for DTLZ Problems

The proposed LSMPSO algorithm was overall the fastest algorithm to find an accurate approximate of the Pareto front for 10 of 12 problems. In addition to its high convergence speed, LSMPSO achieved a hit rate of 100% for all the studied problems.

References

1. Kennedy J., and Eberhart R.C. *Particle Swarm Optimization.* In: Proceedings of the 1995 IEEE International Conference on Neural Networks, Piscataway, New Jersey, pp. 1942–1948, 1995.

2. Nebro A., Durillo J., García-Nieto J., Coello C. C., Luna F., Alba E. *SMPSO: a new PSO-based metaheuristic for multi-objective optimization.* Proceedings of the IEEE Symposium Series on Computational Intelligence, Nashville, TN, U.S.A., pp. 66–73, 2009.

3. Reyes Sierra M., and Coello Coello C. A. *Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and Dominance.* In Evolutionary Multi-Criterion Optimization (EMO 2005), LNCS 3410, Guanajuato, Mexico, pp 505-519, 2005.

4. Durillo J.J., Nebro A.J., Luna F., Coello Coello C.A., and Alba E. *Convergence Speed in Multi-Objective Metaheuristics: Efficiency Criteria and Empirical Study*. International Journal for Numerical Methods in Engineering, Vol. 83, No. 3, 2010.
5. Zitzler E., Deb K., Thieler L. *Comparison of multiobjective evolutionary algorithms: Empirical results.* IEEE Trans. on Evol. Computation, Vol. 8, pp. 173-195, 2000.
6. Deb K., Thiele L., Laumanns M., and Zitzler E. *Scalable Test Problems for Evolutionary Multiobjective Optimization.* In Evolutionary Multiobjective Optimization. Theoretical Advances and Applications, Abraham A, Jain L, Goldberg R (eds), Springer USA, pp. 105–145, 2005.
7. L., Zitzler E. and Thiele. *Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach.* IEEE Trans. Evol. Comput., Vol. 3, No. 4, pp. 257–271, November, 1999.
8. Durillo J.J., Nebro A.J., Coello Coello C.A., Garcıa-Nieto J., Luna F., and Alba E. *A Study of Multiobjective Metaheuristics When Solving Parameter Scalable Problems.* IEEE Transactions On Evolutionary Computation, Vol. 14, No. 4, pp. 618-636, 2010.
9. Reyes-sierra M, and Coello CAC. *Multi-objective particle swarm optimizers: A survey of the state-of-the-art.* International Journal of Computational Intelligence Research, Vol. 2, No. 3, pp. 287-308, 2006.
10. Hernández-Domínguez JS, Pulido GT, Coello CAC. *A Multi-objective Particle Swarm Optimizer Enhanced with a Differential Evolution Scheme.* Artificial Evolution, pp. 169-180, 2011.
11. Xu H. Wang Y, Xu X. *Dominating Global Best Selection for Multi-objective Particle Swarm Optimization.* Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013), pp. 1503-1505, 2012.
12. Castro Junior OR, de Britto AB., Pozo A. *A Comparison of methods for leader selection in many-objective problems.* IEEE Congress on Evolutionary Computation, Brisbane, Australia, pp. 1-8, 2012.
13. Deb K., Pratap A., Agarwal S., and Meyarivan T. *A fast and elitist multiobjective genetic algorithm: NSGA-II.* IEEE Transactions on Evolutionary Computation, Vol. 6, No. 2, pp. 182–197, 2002.
14. Durillo J.J., Nebro A.J., and Alba E. *The jMetal Framework for Multi-Objective Optimization: Design and Architecture.* IEEE-CEC 2010, pp. 4138-4325, July, 2010.
15. Deb K., Pratap A., Agarwal S., and Meyarivan T. *A fast and elitist multiobjective genetic algorithm: NSGA-II.* IEEE Transactions on Evolutionary Computation, 6(2), pp. 182–197, 2002.
16. E. Zitzler, M. Laumanns, and L. Thiele. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm.* EUROGEN 2001, Vol. 3242, No. 103, pp. 95–100, 2002.
17. Nebro A, Durillo J, García-Nieto J, Coello CC, Luna F, Alba E. *SMPSO: a new PSO-based metaheuristic for multi-objective optimization.* Proceedings of the IEEE Symposium Series on Computational Intelligence, Nashville, TN, U.S.A., 2009; 66–73.
18. Kukkonen S., Lampinen J. *GDE3: the third evolution step of generalized differential evolution.* IEEE Congress on Evolutionary Computation (CEC'2005), Edinbourgh, U.K., pp. 443–450, 2005.
19. Lampinen J. *DE's selection rule for multiobjective optimization.* Technical report, Lappeenranta University of Technology, Department of Information Technology, 2001.
20. Nebro A. J., , Durillo J. J., Luna F., Dorronsoro B., and Alba E. *A cellular genetic algorithm for multiobjective optimization.* In Nature Inspired Cooperative Strategies for Optimization (NICSO 2006), Grenada, Spain, pp. 25–36, 2006.
21. Nebro A. J., Luna F., Alba E., Dorronsoro B., Durillo J. J., and Beham A. *AbYSS: Adapting scatter search to multiobjective optimization.* IEEE Transactions on Evolutionary Computation, Vol. 12, No. 4, pp. 439-457, 2008.
22. Zhang Q. and Li H. *MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition IEEE Transactions on Evolutionary Computation, Vol. 11, No. 6, 2007.*
23. J, Sheskin D. *Handbook of Parametric and Nonparametric Statistical Procedures*. 4th ed. New York: Chapman & Hall/CRC Press, 2007.
24. Tukey J.W. *Exploratory Data Analysis.* Addison-Wesley, 1977.