Particle Swarm Optimization Based on Linear Assignment Problem Transformations

Luis Miguel Antonio
CINVESTAV-IPN,
Departamento de Computación
Av. IPN 2508. San Pedro Zacatenco
México D.F. 07300, MÉXICO
lmiguel@computacion.cs.cinvestav.mx

Carlos A. Coello Coello
CINVESTAV-IPN,
Departamento de Computación
Av. IPN 2508. San Pedro Zacatenco
México D.F. 07300, MÉXICO
ccoello@cs.cinvestav.mx

ABSTRACT

Particle swarm optimization (PSO) algorithms have been widely used to solve a variety of optimization problems. Their success has motivated researchers to extend the use of these techniques to the multi-objective optimization field. However, most of these extensions have been used to solve multi-objective optimization problems (MOPs) with no more than three objective functions. Here, we propose a novel multi-objective PSO (MOPSO) algorithm characterized by the use of a recent approach that transforms a MOP into a linear assignment problem (LAP), with the aim of being able to solve many-objective optimization problems. Our proposed approach, called LAP based PSO (LAPSO), adopts the Munkres assignment algorithm to solve the generated LAPs and has no need of an external archive. LAPSO is compared with respect to three MOPSOs which are representative of the state-of-the-art in the area: the Optimized Multi-Objective Particle Swarm Optimizer (OMOPSO) the Speed-constrained Multiobjective Particle Swarm Optimizer (SMPSO) and a variant of the latter that uses the hypervolume indicator for its leader selection scheme (SMPSOhv). Our results indicate that LAPSO is able to outperform the MOPSOs with respect to which it was compared in most of the test problems adopted, specially when solving instances with more than three objectives.

1. INTRODUCTION

Particle swarm optimization (PSO) is a metaheuristic inspired on the social behavior of birds within a flock [14]. Due to its simplicity and good performance, PSO has become a very popular approach to solve optimization problems and many different extentions to Multi-Objective Optimization have been reported in the specialized literature [28]. Most Multi-Objective Particle Swarm Optimizers (MOPSOs) adopt a selection process of the local and global best solutions based on Pareto dominance and the use of a crowding factor. However, it is now well-known that Pareto-based multi-objective evolutionary algorithms (MOEAs) do not perform properly when dealing with problems having more than three objectives (the so-called *many-objective optimization* problems) [11]. In such problems, the number of non-dominated solutions significantly increases, making it difficult to generate new particles that dominate the previous ones, giving rise to the so-called *dominance resistance* [15, 26]. In this paper, we propose a novel MOPSO designed to solve many-objective optimization problems. We adopt an approach that transforms the original MOP into a linear assignment problem (LAP), recently presented in [19], in order to avoid the scalability problems of Pareto-based approaches. Our preliminary results indicate that our proposed approach is a very good alternative for solving many-objective optimization problems.

The remainder of this paper is organized as follows. Section 2 states the problem of our interest. The previous related work is discussed in Section 3. Section 4 describes our proposed approach and the experiments carried out to validate it. Finally, our conclusions and some possible paths for future work are drawn in Section 5.

2. THE MULTI-OBJECTIVE OPTIMIZATION PROBLEM

Formally, a multi-objective optimization problem (MOP) is defined as follows:

$$\text{minimize}\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \quad (1)$$

subject to:

$$g_i(\vec{x}) \leq 0 \qquad i = 1, 2, \ldots, m \quad (2)$$

$$h_i(\vec{x}) = 0 \qquad i = 1, 2, \ldots, p \quad (3)$$

where k is the number of objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $g_i, h_j : \mathbb{R}^n \to \mathbb{R}, i = 1, ..., m, j = 1, ..., p$ are the constraint functions of the problem and $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ the vector of decision variables. We thus wish to determine from the set Ω (where Ω is the feasible region) of all the vectors that satisfy (2) and (3) to the vector $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ of solutions that are *Pareto optimal* (those solutions in which one objective cannot be improved without worsening another).

3. PARTICLE SWARM OPTIMIZATION

In PSO, a group of candidate solutions is represented as a set of *particles* in a *swarm* [32]. Each particle has to adjust its flight trajectory into the search space according to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'15, July 11-15, 2015, Madrid, Spain.
Copyright 2015 ACM TBA ...$15.00.

a certain direction and keeping a certain *velocity*, given by its own previous flight experience and those of their neighboring particles in the swarm. PSO starts with a random initialization of each particle's position and velocity (particle's displacement) in decision variable space. Then, the velocity v and the position x of the each particle are updated at each step t according to the following equations:

$$\vec{v}_i(t) = w\vec{v}_i(t-1) + C_1 r_1(\vec{x}_{p_i} - \vec{x}_i) + C_2 r_2(\vec{x}_{g_i} - \vec{x}_i) \quad (4)$$

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \quad (5)$$

where x_{p_i} is the best solution that x_i has viewed so far, x_{g_i} is the best particle (also known as the leader) that the entire swarm or a certain pre-defined neighborhood has experienced, W is the inertia weight of the particle, which controls the impact of the previous velocity $v_i(t-1)$ on the movement of the particle, aimed to prevent swarm explosion (i.e., an uncontrolled increase of a particle's velocity), r_1 and r_2 are two uniformly distributed random numbers in the range $[0, 1]$ and C_1 and C_2 are the learning factors which control the effect of the personal and social influence. Particles in the swarm interact by defining a common set of links, which controls the exchange of information between particles, and is known as *swarm topology*. The set of particles informing the ith particle is defined as the particle's neighborhood, which in most cases includes the particle itself as a member. Traditional topologies for PSO include the gbest (star), lbest (circle), and von Neumann [13] topologies. Also, an adaptive random topology has been proposed. Here, each particle randomly informs K neighbors and itself (the same particle may be chosen several times), with K usually set to 3. In this topology, the connections between particles randomly change when the social optimum shows no improvement [3].

3.1 Multi-objective PSO algorithms

The successful application of PSO in a wide variety of single-objective optimization problems has made it very popular in recent years. However, in order to handle multiple objectives, PSO must be obviously modified. In most Multiple-Objective Particle Swarm Optimizers (MOPSOs), the major modifications to the basic PSO algorithm are the selection process of *pbest* and *gbest*. Among them, the differences lie on the number of selected global leaders and in the way in which they are selected [5]. Most of these approaches share the use of an external archive where the nondominated solutions found during the search are stored [27, 23, 28, 22]. The most commonly employed strategy is to use only one global leader randomly selected from the archive. Other works consider the use of a density estimator for selecting leaders from the archive, aiming to favor the spread of the computed approximation to the Pareto front. Next, we briefly describe some MOPSOs representative of the state-of-the-art in the area.

The first implementation of a MOPSO was proposed by Moore et al. [20]. Here, they used Pareto dominance for leader and personal best selection. Reddy et al. [12] proposed a MOPSO where the leader was randomly selected from an external archive where the best solutions found during the process were stored. The main target of this work was the use on an elitist-mutation-mechanism in combination with PSO. In [29] Santana et al. proposed a MOPSO which incorporates a *turbulence* (mutation) operator in addition to the crowding distance mechanism and a roulette wheel to select the social leader and to prevent an excessive number of nondominated solutions in the external archive. Another approach is the Optimized Multi-Objective Particle Swarm Optimizer (OMOPSO) [27]. This MOPSO selects a leader by means of a binary tournament based on the crowding value of the leaders. The maximum size of the set of leaders is fixed equal to the size of the swarm. After each generation, the set of leaders is updated, and so are the corresponding crowding values. If the size of the set of leaders is greater than the maximum allowable size, only the best leaders are retained based on their crowding value. The rest of the leaders are eliminated. OMOPSO adopts two mutation operators: uniform mutation and non-uniform mutation. The swarm is subdivided in three parts (of equal size). Each sub-part of the swarm adopts a different mutation scheme: the first sub-part uses no mutation at all, the second sub-part uses uniform mutation and the third sub-part adopts non-uniform mutation. With the use of these different operators, OMOPSO has the ability to explore and exploit the search space. It also adopts the concept of ϵ-dominance in order to fix the size of the external archive that contains the (nondominated) solutions that will be reported by the algorithm, so the size of the final external archive depends on the ϵ-value, which needs to be obtained from a pre-sampling process.

In [23] the Speed-constrained Multi-objective PSO (SMPSO) is presented. The authors of this approach adopt a control mechanism for particle's velocity that, instead of using upper and lower parameter values to limit the step size of the velocity, adopts a constriction coefficient obtained from a constriction factor χ, originally developed by Clerc and Kennedy in [4]. The following equations are adopted for this sake:

$$\chi = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \quad (6)$$

where

$$\varphi = \begin{cases} C_1 + C_2 & \text{if} \quad C_1 + C_2 > 4 \\ 0 & \text{if} \quad C_1 + C_2 \leq 4 \end{cases} \quad (7)$$

SMPSO also introduces a mechanism where the accumulated velocity of each variable (in each particle) is further bounded by means of the following velocity constriction equation:

$$v_i(t) = \begin{cases} \delta_i & \text{if} \quad v_i(t) > \delta_i \\ -\delta_i & \text{if} \quad v_i(t) \leq -\delta_i \\ v_i(t) & \text{otherwise} \end{cases} \quad (8)$$

where

$$\delta_i = \frac{upper_limit_i - lower_limit_i}{2} \quad (9)$$

So, the velocity of the particles is calculated according to eq. (4). The resulting velocity is then multiplied by the constriction factor and the resulting value is constrained using eq. (8). SMPSO uses a limited size external archive to store the non-dominated solutions found during the search. When the archive becomes full, the solutions with the smallest crowding distance are discarded. It also uses a turbulence

operator by means of polynomial mutation, but it applies it to only 15% of the whole swarm. Later, a variant of SMPSO that uses the hypervolume indicator [36] to guide leader selection was presented in [22]. In this variant, called SMPSOhv, when the velocity of a particle has to be updated, two solutions are randomly selected from the archive, and the one contributing the most to the archive's hypervolume is selected as the leader. To apply this scheme, the external archive of non-dominated solutions is changed for an archive managed by the contribution of each solution to the value of this indicator.

There are some MOPSOs that consider the use of more than one particle from the archive as global leaders at the same time. An example is the approach described in [18], called Multi Leaders Multi Objective Optimization algorithm, which is an initial implementation of multiple leaders in guiding the particles' flight to search for optimum solutions. The multiple leaders' method is implemented by summing up all the distances between a particle and all of its leaders during the velocity update. Many other MOPSOs exist [24], but most of them are Pareto-based approaches and solve MOPs with no more than 3 objectives. Here, we propose a novel MOPSO which adopts the transformation of a MOP into a linear assignment problem (LAP), with the specific aim of solving many-objective optimization problems.

4. OUR PROPOSED APPROACH

As mentioned before, studies have shown that Pareto-based multi-objective evolutionary algorithms (MOEAs) do not perform properly when dealing with problems having more than three objectives (the so-called *many-objective optimization* problems) [11] and most MOPSOs' leader selection strategies are based on this same idea. Here, we propose to use an alternative selection mechanism which is not based on Pareto dominance or on any performance indicator. The algorithm presented here transforms the selection process into a linear assignment problem (LAP), which is solved using the *Munkres assignment algorithm* [17]. As evidenced in [19], the solution of this LAP allows convergence towards the true Pareto front and, at the same time, a good distribution of solutions along the Pareto front.

4.1 Linear assignment problem transformation

The matching or assignment problem is a fundamental class of combinatorial optimization problems [9, 21]. In its most general form, an assignment problem is the problem of choosing an optimal assignment of n to m of tasks, assuming that numerical ratings are incurred for each agent performing each task [17]. An optimal assigment is one which makes the sum of the agents' ratings for their tasks a maximum or minimum, according to the context of the problem. The Linear Assignment Problem (LAP) is the simplest of the assignment problems. In the canonical LAP, there are as many agents as tasks, and any agent can be assigned to perform any task. Formally, the LAP can be formulated as follows:

Let $A = \{a_1, ..., a_n\}$ and $T = \{t_1, ..., t_n\}$ be a set of agents and tasks with the same cardinality, given a cost function $C : A \times T \rightarrow \mathbb{R}$ and having $\Phi : A \rightarrow T$ as the set of all possible bijections between A and T, then a LAP can be expressed as follows :

$$\underset{\phi \in \Phi}{\text{minimize}} \sum_{a \in A} C(a, \phi(a)) \tag{10}$$

Usually, the cost function is also viewed as a squared real-valued matrix C with elements $C_{ij} = C(a_i, t_j)$, and the set Φ of all possible bijections between A and T as a set of assignment matrices $\mathcal{X}$. So, the LAP can be expressed as an integer linear program as follows:

$$\begin{aligned} \underset{x \in \mathcal{X}}{\text{minimize}} \quad & \sum_{i,j=1}^{n} C_{ij} x_{ij} \\ \text{subject to:} \quad & \sum_{i=1}^{n} x_{ij} = 1, \ \forall j \in \{1, .., n\}, \\ & \sum_{j=1}^{n} x_{ij} \leq 1, \ \forall i \in \{1, ..., n\}, \\ & x_{ij} \in \{0, 1\}, \ \forall i, j \in \{1, ..., n\} \end{aligned} \tag{11}$$

In 1955, Harold W. Kuhn [17] proposed an algorithm for constructing a maximum weight perfect matching in a bipartite graph. This combinatorial optimization algorithm solves the assignment problem in polynomial time. Later on, James Munkres [21] reviewed Kuhn's work and made several important contributions to the theoretical aspects of the algorithm. He showed that Kuhn's algorithm is (strongly) polynomial and proposed an improved version of $O(n^3)$. The contribution of Munkres to the development of the Hungarian algorithm has led to the algorithm which is referred to as the Kuhn-Munkres or Munkres assignment algorithm. In [2], Bourgeois and Lassalle developed an extension for rectangular matrices which allows the algorithm to operate in assignment problems where the numbers of agents and tasks are unequal. Such extension can be formulated as follows: Given a $n \times m$ matrix (c_{ij}) of real numbers, find a set of k independent elements $[k = min(n, m)]$ so that the sum of these elements is minimum.

A linear assignment problem can be created from a MOP using the k-dimensional objective vectors from the individuals (solutions) in a MOEA. Since uniformly spread weight vectors in objective function space can be created, one can reformulate the MOP as follows: having n individuals and m vectors well-distributed in a $(k-1)$-dimensional unit simplex of the objective space, a cost can be incurred for each individual representing some vector in the Pareto Front approximation. So, the goal is to describe all regions covered by the n vectors using only m individuals in such a way that the total cost of the assignment is minimized. A cost matrix is then created such that it minimizes the total cost involved in retaining the solutions which are a good approximation of the Pareto Front. This procedure is described next.

First, the n vectors of objective values are normalized to reduce the current objective space to a unit hypercube. This is done in order to deal with non-commensurable objective functions. The maximum $\vec{z}^{max}$ and minimal $\vec{z}^{min}$ vectors are calculated for this purpose.

$$\begin{aligned} \vec{z}^{max} &= [z_1^{max}, ..., z_k^{max}]^T, z_i^{max} = \max_{j=1,...,2n} f_i(\vec{x}_j), i = 1, ..., k, \\ \vec{z}^{min} &= [z_1^{min}, ..., z_k^{min}]^T, \ z_i^{min} = \min_{j=1,...,2n} f_i(\vec{x}_j), i = 1, ..., k, \end{aligned} \tag{12}$$

where $f_i(\vec{x}_j)$ is the ith objective value of the jth solution in

Q_g, and its normalized value $f_i(\vec{x}_j)$ is calculated as:

$$\tilde{f}_i(\vec{x}_j) = \frac{f_i(\vec{x}_j) - z_i^{min}}{z_i^{max} - z_i^{min}}, \quad j = 1, ..., n, \; i = 1, ..., k. \quad (13)$$

Let W be a set of m weight vectors uniformly scattered in objective space.

$$W \subset \mathbb{W} = \{\vec{w} \mid \vec{w} \in [0,1]^k, \sum_{i=1}^{k} w_i = 1\}, \; |W| = m, \quad (14)$$

The cost C_{rj} of assigning the individual $\vec{x}_j$ to the weight vector $\vec{w}_r$ is given by:

$$C_{rj} = \max_{i=1,\ldots,k} w_{ri} \times \tilde{f}_i(\vec{x}_j), \; r = 1, ..., n, \; j = 1, ..., n. \quad (15)$$

The matrix C indicates how each solution is suitable to represent each region of the Pareto Front approximation. The solution to our assignment problem is found by identifying the combination of values in C resulting in the smallest sum, subject to the following conditions:

- Exactly one value must be chosen in each row; this ensures that only one solution is assigned to each position on the Pareto Front.
- At most one value can be selected in each column; this ensures that no solution is assigned to more than one position.

The matrix C and the above conditions are formally represented by (11) as a linear programming problem. The solution to this problem is then obtained by the aforementioned Kuhn-Munkres algorithm for rectangular matrices [2]. The matrix that solves (11) represents the solutions assigned to each weight vector such that it minimizes the total cost of the assignment, allowing to retain the best solutions to approximate the Pareto Front.

4.2 Generation of weight vectors by Uniform Design

There exist several MOEAs [35, 25] that require the computation of a set of weight vectors uniformly scattered on a $(k-1)$-unit simplex to obtain solutions along the entire Pareto Front in a k-objective optimization problem. A variety of methods to obtain an evenly distributed subset of weights in a simplex are available in the specialized literature [8]. The simplex-lattice design method [30] is the approach that has been the most commonly adopted in MOEAs. However, at least three problems can be identified in this method [8]. First, the weight vectors are not uniformly distributed. Also, there are too many vectors at the boundary of the domain. Finally, the number of vectors generated increases nonlinearly with the number of objectives. Thus, if H divisions are considered along each objective, the total number of weight vectors (hence the population size) in a k-objective problem is given by: $\binom{H+k-1}{k-1}$. Some MOEAs have resorted to other methods to generate an arbitrary number of scattered weight vectors. In [25] a hypervolume-based weight vector generation is proposed. This method produces well-distributed vectors maximizing the hypervolume covered by them in objective space. Another approach was presented in [31], where the uniform design (UD) [8] and good lattice point (*glp*) [16] methods were combined to set the weight vectors. However, both the hypervolume and the *glp* method have a high computational cost when the number of objectives grows, which prevents their use in problems with many objectives.

Uniform design is a space filling design method that seeks experimental points to be uniformly scattered in the domain [8]. In uniform design, a set of points is considered uniformly spread throughout the entire domain if it has a small *discrepancy*, where discrepancy is a numerical measure of scattering. Fang and Wang [8] presented different methods for generating points that can be applied to the generation of a set of space-filling design points. Among them, the good lattice point (*glp*) method and Hammersley's method [10] excel. In [19] the authors propose to generate weight vectors using uniform design combined with Hammersley's method because this algorithm allows a more uniform distribution of the weight vectors over the space than the simplex-lattice method, and the population size of the MOEAs neither increases nonlinearly with the number of objectives nor considers a formulaic setting. Additionally, Hammersley's method provides a set of design points with low discrepancy similar to the *glp* method, but at a much lower computational cost [8].

Algorithm 1: Generation of weight vectors

```
Input  : number of objectives (k), number of weights (n)
Output: Set of weight vectors W
p ← array with the first k − 2 prime numbers;
U ← ∅;
for i = 1 to n do
    u_i1 ← (2i − 1)/2n;
    for j = 2 to k − 1 do
        u_ij ← 0;
        f ← 1/p_{j−1};
        d ← i;
        while d > 0 do
            u_ij ← u_ij + f × (d mod p_{j−1});
            d ← ⌊d/p_{j−1}⌋;
            f ← f/p_{j−1};
        end
    end
    U ← U ∪ {u};
end
W ← Apply transformation (19) to U;
```

Hammersley's method is based on the p-adic representation of natural numbers: Any positive integer m can be uniquely expressed using a prime base $p \geq 2$ as

$$m = \sum_{i=0}^{r} b_i \times p^i, \quad 0 \leq b_i \leq p-1, \quad i = 0, \ldots, r, \quad (16)$$

where $p^r \leq m < p^{r+1}$. Then, for any integer $m \geq 1$ with representation (16), let

$$y_p(m) = \sum_{i=0}^{r} b_i \times p^{-(i+1)}, \quad (17)$$

where $y_p(m) \in (0,1)$ and is known as the radical inverse of m base p. Let $k \geq 2$ and $p_1, \ldots, p_{k-1}$ be $k-1$ distinct prime numbers; then, the Hammersley set consisting of n points uniformly scattered on $[0,1]^k$ is given by

$$\vec{x}_i = \left[\frac{2i-1}{2n}, y_{p_1}(i), \ldots, y_{p_{k-1}}(i)\right]^T, \; i = 1, \ldots, n. \quad (18)$$

In [33], it was proposed the use of uniform design for experiments with mixture (UDEM). This method seeks points

to be uniformly scattered in the domain $\mathbb{W}$ defined by (14). The authors employed the transformation method for the construction of such uniform design. This method requires a set of vectors $U = \{\vec{u}_i = [u_{i1}, ..., u_{i(k-1)}]^T, i = 1, ..., n\} \subset [0,1]^{k-1}$ with small discrepancy. In our proposal, Hammersley's method is used to obtain U and then to apply the next transformation:

$$\begin{aligned} w_{ti} &= (1 - u_{ti}^{\frac{1}{k-i}}) \prod_{j=1}^{i-1} u_{tj}^{\frac{1}{k-j}}, \quad i = 1, ..., k-1, \\ w_{tk} &= \prod_{j=1}^{k-1} u_{tj}^{\frac{1}{k-j}}, \quad t = 1, ..., n. \end{aligned} \tag{19}$$

Then $\{\vec{w}_t = [w_{ti}, ..., w_{tk}]^T,\ t = 1, ..., n\}$ is a uniform design on $\mathbb{W}$. The pseudocode of the algorithm used to generate weight vectors is presented in Algorithm 1.

4.3 Description of the proposed approach

Making use of the LAP transformation, we propose a MOPSO that adopts an evolutionary model to select solutions based on the LAP transformation as described next. LAPSO starts with an initial swarm $S(0)$ of n particles and performs a random initialization of each particle's position and velocity (particle displacement) within decision variables space. The velocity v of each particle is updated at each step t according to eq. (4), where x_{p_i} is the best solution that x_i has viewed so far (in terms of the LAP transformation already described), and xg_i is the best particle within a defined neighborhood in terms of Pareto dominance. For this purpose, we adopted an adaptive random topology originally proposed in [3], which has been found to produce good results in the area of global optimization [34, 1]. Here, each particle randomly informs K neighbors and itself (the same particle may be chosen several times). In this topology, the connections among particles randomly change when the social optimum shows no improvement (we set $K = 3$). As there is evidence of the problems that an uncontrolled increase of velocity may cause [7], we adopt the velocity constriction mechanism of SMPSO described in Section 3.1. So, the inertia weight W, the two uniformly distributed random numbers r_1 and r_2 and the learning factors C_1 and C_2 (for the personal and social influence) were adopted, same as in SMPSO [23]. After the initialization, at each step t, we create a new swarm $S_2(t)$ of n particles from the actual swarm $S(t)$. Each particle will be created by means of eq. (5), where $v_i(t)$ is the velocity, computed as we just described. After that, a turbulence operation (mutation) will be applied to this new swarm and then each particle is evaluated according to the given MOP. Having both swarms, we form a set $CS_t = S(t) \cup S_2(t)$ of $2n$ posible solutions to the MOP. Then, a linear assignment problem is created using the k-dimensional objective vectors from CS_t (as explained in Section 4.1) and n weight vectors uniformly spread in objective function space are created as described in Algorithm 1. Then, a selection procedure based on the LAP transformation will be performed in order to select n particles from $CS(t)$ which are closer to the weight vectors and to solve the created LAP. These particles will form the next swarm $S(t+1)$. With this, there is no need of an external archive since the actual swarm always keeps the best solutions to the LAP and the members of the neighborhood for each particle are also in the swarm. This process is repeated until a stopping criterion is fulfilled. The whole procedure is summarized in Algorithm 2.

Algorithm 2: Linear assignment problem based PSO (LAPSO)

Input : MOP, swarm size n, maximum number of iterations t_{max}
Output: Solution set S

```
Generate initial swarm population S(0) and their velocity values randomly;
Assign k = 3 neighbors to each particle in S;
Evaluate each particle in S;
W ← Generate n weight vectors using Algorithm 1;
for t = 1 to t_max do
    S_2(t) ← Generate new particles from S_t using (5) ;
    Apply turbulence operator to each individual in S_2(t);
    Evaluate each individual from S_2(t);
    if social optimum shows no improvement then
        recompute neighbors;
    end
    CS_t ← S(t) ∪ S_2(g);
    Compute z^max and z^min by (12) Normalize objectives of
    each solution in CS_t by (13);
    Generate the cost matrix C by (15) using CS_t and W;
    I ← Obtain the best assignment in C using the Hungarian
    Method;
    S(t+1) ← {x_i | i ∈ I , x_i ∈ CS_t};
    update velocity values using (4);
end
```

4.4 Experimental Results

We validated LAPSO comparing its performance with respect to the three aforementioned MOPSOs representative of the state-of-the-art in the area: OMOPSO [27], SMPSO [23] and SMPSOhv [22]. For the purposes of this study, we adopted the Deb-Thiele-Laumanns-Zitzler (DTLZ) test suite [6] with instances from three to ten objectives. In order to assess the performance of each MOPSO, we selected the hypervolume indicator [36], since this measure can differentiate between degrees of complete outperformance of two sets. The hypervolume is defined as the n-dimensional space that is contained by an n-dimensional set of points. When applied to multi-objective optimization, the n-dimensional objective values for solutions are treated as points for the computation of such space. That is, the hypervolume is obtained by computing the volume (in objective function space) of the nondominated set of solutions Q that minimize a MOP. For every solution $i \in Q$, a hypercube v_i is generated with a reference point W and the solution i as its diagonal corner of the hypercube:

$$\mathcal{S} = Vol\left(\bigcup_{i=1}^{|Q|} v_i\right) \tag{20}$$

The aim of this study is to identify which of the MOPSOs being compared is able to get closer to the true Pareto front using the same number of objective function evaluations and how they behave as the dimensionality of the MOP increases.

4.5 Parameterization

The parameters of each MOPSO used in our study were chosen in such a way that we could do a fair comparison. Thus, the same number of iterations and swarm size were adopted. For all MOPSOs, the mutation probability was set to $p_m = 1/l$, where l is the number of decision variables;

the distribution indexes for the polynomial mutation used by SMPSO, SMPSOhv and LAPSO were set as: $\eta_c = 20$ and $\eta_m = 20$. Different population sizes for each problem instance were used. For problems with 3 and 4 objectives the population size was set to 100. For problems having 5 objectives, the population size was set to 120. For problems with 6, 7, 8 and 9 objectives, a swarm size of 200 was used. Finally, for problems with 10 objectives, the swarm size was set to 220. The maximum number of iterations adopted for problems with 3 objectives was 100. In all other problems we used 300 iterations, regardless of their dimensionality. All other parameters, such as the learning factors and inertia weight, were set as the authors suggest for their respective approaches. The reference points $\vec{y}_{ref} = [y_1, \cdots, y_m]$ were $y_i = 1.5$ for DTLZ1to DTLZ6 and $y_i = 6.5$ for DTLZ7.

4.6 Discussion of Results

In our experiments, we obtained the hypervolume value over the 30 independents runs performed. Table 1 shows the average hypervolume of each of the MOPSOs being compared for each test problem adopted, as well as the results of the statistical analysis that we made to validate our experiments, for which we used Wilcoxon's rank sum. OMOPSO presented a poor performance in all problem instances having more than three objectives, since it couldn't produce solutions which dominate the reference points for the hypervolume computation used in our experimentes. It was only able to scale in DTLZ4 with at most five objetives and we can reject the null hypothesis (medians are equal) in all the cases. SMPSO produced competitive results for DTLZ4, DTLZ5 and DTLZ6, although it could not outperform LAPSO except for one case (DTLZ5 with 3 objectives), in which SMPSO presented a marginal improvement with respect to LAPSO, we cannot reject the null hypothesis in only two cases, DTLZ5 with 4 objectives and DTLZ7 with 3, which means that in these cases both algorithms have a similar behavior. Due to its high computational cost, a time limit of 12 hours for SMPSOhv's runs was used. As a result, SMPSOhv was no able to produce results for more than four objetives. In the comparisons using problems with three and four objectives, LAPSO was able to outperform SMPSOhv in DTLZ1, DTLZ3 and DTLZ7. Regarding the other test problems, LAPSO was able to produce very competitive results but at a significantly lower computational cost and has similar behavior, which means the null hypothesis cannot be rejected, in only three cases (DTLZ1, DTLZ3 and DTLZ7 with dimensions of 3, 4 and 3 respectively).

5. CONCLUSIONS AND FUTURE WORK

Here, we developed a novel MOPSO called LAPSO, which adopts a recent selection scheme originally proposed for MOEAs. LAPSO uses a selection based on the transformation of a MOP into an assignment problem. The implicit elitism associated with this scheme makes unnecessary the use of an external archive. Using a set of well-distributed points on a unit simplex, the obtained assignment problem is solved with the Munkres assignment algorithm. Our experimental results indicate that LAPSO outperforms OMOPSO and SMPSO in MOPs having from four up to ten objective functions and showed very competitive results with respect to SMPSOhv but at a significantly lower computational cost. LAPSO was able to deal with all the difficulties presented in the DTLZ test suite, even in high dimensionality. As part of our future work, we intend to study other algorithms for solving the LAP transformation. We are also interested in studying the possible use of other topologies for LAPSO.

Acknowledgments

The first author acknowledges support from CONACyT and CINVESTAV-IPN to pursue graduate studies in Computer Science. The second author gratefully acknowledges support from CONACyT project no. 221551.

6. REFERENCES

[1] M. R. Bonyadi and Z. Michalewicz. Spso 2011: Analysis of stability; local convergence; and rotation sensitivity. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 9–16, New York, NY, USA, 2014. ACM.

[2] F. Bourgeois and J.-C. Lassalle. An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices. *Commun. ACM*, 14(12):802–804, December 1971.

[3] M. Clerc. *Particle Swarm Optimization*. ISTE (International Scientific and Technical Encyclopedia), USA, 2006.

[4] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, Feb 2002.

[5] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.

[6] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Test Problems for Evolutionary Multiobjective Optimization. In A. Abraham, L. Jain, and R. Goldberg, editors, *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145. Springer Berlin Heidelberg, USA, 2005.

[7] J. Durillo, A. Nebro, C. Coello Coello, J. Garcia-Nieto, F. Luna, and E. Alba. A Study of Multiobjective Metaheuristics When Solving Parameter Scalable Problems. *IEEE Transactions on Evolutionary Computation*, 14(4):618–635, August 2010.

[8] K. T. Fang and Y. Wang. *Number-Theoretic Methods in Statistics*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1994.

[9] D. Gale and L. S. Shapley. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[10] J. M. Hammersley. Monte-Carlo methods for solving multivariable problems. *Annals of the New York Academy of Sciences*, 86(3):844–874, 1960.

[11] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *2008 Congress on Evolutionary Computation (CEC'2008)*, pages 2424–2431, Hong Kong, June 2008. IEEE Service Center.

[12] M. Janga Reddy and D. Nagesh Kumar. An efficient multi-objective optimization algorithm based on swarm intelligence for engineering design. *Engineering Optimization*, 39(1):49–68, 2007.

Function	No. Obj	LAPSO HV	OMOPSO HV	LAPSO-OMOPSO P(H)	SMPSO HV	LAPSO-SMPSO P(H)	SMPSOhv HV	LAPSO-SMPSOhv P(H)
DTLZ1	3	3.347	0.000	0.000000 (1)	3.339	0.000000 (1)	3.335	0.386525 (0)
	4	5.053	0.000	0.000000 (1)	5.028	0.002348 (1)	5.037	0.000067 (1)
	5	7.540	0.000	0.000000 (1)	5.563	0.000000 (1)	_	_
	6	11.357	0.000	0.000000 (1)	7.625	0.000000 (1)	_	_
	7	17.054	0.000	0.000000 (1)	7.392	0.000000 (1)	_	_
	8	25.548	0.000	0.000000 (1)	11.892	0.000000 (1)	_	_
	9	38.429	0.000	0.000000 (1)	25.100	0.000000 (1)	_	_
	10	57.652	0.000	0.000000 (1)	48.215	0.000000 (1)	_	_
DTLZ2	3	2.750	1.684	0.000000 (1)	2.697	0.000000 (1)	2.791	0.000000 (1)
	4	4.559	0.000	0.000000 (1)	4.058	0.000000 (1)	4.631	0.000000 (1)
	5	7.127	0.000	0.000000 (1)	5.251	0.000000 (1)	_	_
	6	11.025	0.000	0.000000 (1)	7.925	0.000000 (1)	_	_
	7	16.721	0.000	0.000000 (1)	11.383	0.000000 (1)	_	_
	8	25.056	0.000	0.000000 (1)	16.646	0.000000 (1)	_	_
	9	37.704	0.000	0.000000 (1)	26.432	0.000000 (1)	_	_
	10	57.070	0.000	0.000000 (1)	19.386	0.000000 (1)	_	_
DTLZ3	3	2.768	0.000	0.000000 (1)	1.061	0.000000 (1)	2.740	0.000171 (1)
	4	4.266	0.000	0.000000 (1)	3.761	0.000000 (1)	4.067	0.579073 (0)
	5	5.928	0.000	0.000000 (1)	1.111	0.000000 (1)	_	_
	6	9.175	0.000	0.000000 (1)	0.606	0.000000 (1)	_	_
	7	11.201	0.000	0.000000 (1)	0.099	0.000000 (1)	_	_
	8	18.886	0.000	0.000000 (1)	0.124	0.000000 (1)	_	_
	9	30.319	0.000	0.000000 (1)	0.000	0.000000 (1)	_	_
	10	51.368	0.000	0.000000 (1)	1.333	0.000000 (1)	_	_
DTLZ4	3	2.715	2.570	0.000000 (1)	2.694	0.013268 (1)	2.733	0.010280 (1)
	4	4.540	4.435	0.000000 (1)	4.441	0.000000 (1)	4.577	0.000000 (1)
	5	7.135	5.861	0.000000 (1)	6.815	0.000000 (1)	_	_
	6	11.062	0.000	0.000000 (1)	10.626	0.000000 (1)	_	_
	7	16.755	0.000	0.000000 (1)	15.995	0.000000 (1)	_	_
	8	25.211	0.000	0.000000 (1)	23.776	0.000000 (1)	_	_
	9	37.860	0.000	0.000000 (1)	36.303	0.000000 (1)	_	_
	10	57.114	0.000	0.000000 (1)	55.753	0.000000 (1)	_	_
DTLZ5	3	2.027	2.012	0.000000 (1)	2.034	0.000000 (1)	2.036	0.000000 (1)
	4	2.837	0.000	0.000000 (1)	2.824	0.077272 (0)	2.833	0.000000 (1)
	5	4.146	0.000	0.000000 (1)	3.973	0.000000 (1)	_	_
	6	6.131	0.000	0.000000 (1)	5.862	0.000000 (1)	_	_
	7	9.118	0.000	0.000000 (1)	8.565	0.000000 (1)	_	_
	8	13.496	0.000	0.000000 (1)	12.718	0.000000 (1)	_	_
	9	20.033	0.000	0.000000 (1)	19.089	0.000000 (1)	_	_
	10	30.307	0.000	0.000000 (1)	29.167	0.000000 (1)	_	_
DTLZ6	3	2.019	0.893	0.000000 (1)	1.490	0.007810 (1)	1.898	0.000060 (1)
	4	2.855	0.000	0.000000 (1)	2.835	0.000000 (1)	2.889	0.000076 (1)
	5	4.154	0.000	0.000000 (1)	3.811	0.000000 (1)	_	_
	6	6.214	0.000	0.000000 (1)	5.631	0.000000 (1)	_	_
	7	9.175	0.000	0.000000 (1)	8.210	0.000000 (1)	_	_
	8	13.677	0.000	0.000000 (1)	12.258	0.000000 (1)	_	_
	9	20.187	0.000	0.000000 (1)	18.553	0.000000 (1)	_	_
	10	30.286	0.000	0.000000 (1)	27.517	0.000000 (1)	_	_
DTLZ7	3	148.430	66.120	0.000000 (1)	148.217	0.050009 (0)	148.257	0.128758 (0)
	4	789.868	0.000	0.000000 (1)	765.637	0.000003 (1)	788.868	0.000000 (1)
	5	4014.694	0.000	0.000000 (1)	3686.969	0.000000 (1)	_	_
	6	95892.612	0.000	0.000000 (1)	68725.215	0.000000 (1)	_	_
	7	96965.307	0.000	0.000000 (1)	65720.831	0.000000 (1)	_	_
	8	404852.438	0.000	0.000000 (1)	259361.361	0.000000 (1)	_	_
	9	1207098.816	0.000	0.000000 (1)	728523.513	0.026075 (1)	_	_
	10	6956918.762	0.000	0.000000 (1)	1203500.273	0.000000 (1)	_	_

Table 1: **Average of the hypervolume indicator values of the results obtained for the DTLZ test problems. We show average values over 30 independent runs. The cells containing the best hypervolume value for each problem have a grey colored background. The P(H) columns show the results of the statistical analysis applied to our experiments using Wilcoxon's rank sum. P is the probability of observing the given result (the null hypothesis is true). Small values of P cast doubt on the validity of the null hypothesis. $H = 0$ indicates that the null hypothesis ("medians are equal") cannot be rejected at the 5% level. $H = 1$ indicates that the null hypothesis can be rejected at the 5% level.**

[13] J. Kennedy. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, volume 3, 1999.

[14] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Nov 1995.

[15] I. Kokolo, K. Hajime, and K. Shigenobu. Failure of Pareto-based MOEAs: Does Non-dominated Really Mean Near to Optimal? In *Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001)*, volume 2, pages 957–962, Piscataway, New Jersey, May 2001. IEEE Service Center.

[16] N. Korobov. The approximate computation of multiple integrals. *Doklady Akademii Nauk SSSR*, 124:1207–1210, 1959.

[17] H. W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, Mar. 1955.

[18] K. S. Lim, S. Buyamin, A. Ahmad, and Z. Ibrahim. An improved leader guidance in multi objective particle swarm optimization. In *Modelling Symposium (AMS), 2012 Sixth Asia*, pages 34–39, May 2012.

[19] J. A. Molinet Berenguer and Carlos A. Coello Coello. Evolutionary Many-Objective Optimization Based on Kuhn-Munkres' Algorithm. In A. Gaspar-Cunha, C. H. Antunes, and C. Coello Coello, editors, *Evolutionary Multi-Criterion Optimization, 8th International Conference, EMO 2015*, pages 3–17. Springer. Lecture Notes in Computer Science Vol. 9019, Guimarães, Portugal, March 29 - April 1 2015.

[20] J. Moore, R. Chapman, and G. Dozier. Multiobjective particle swarm optimization. In *Proceedings of the 38th Annual on Southeast Regional Conference*, ACM-SE 38, pages 56–57, New York, NY, USA, 2000. ACM.

[21] J. Munkres. Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, Mar. 1957.

[22] A. J. Nebro, J. J. Durillo, and C. A. Coello Coello. Analysis of Leader Selection Strategies in a Multi-Objective Particle Swarm Optimizer. In *2013 IEEE Congress on Evolutionary Computation (CEC'2013)*, pages 3153–3160, Cancún, México, 20-23 June 2013. IEEE Press. ISBN 978-1-4799-0454-9.

[23] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. A. Coello Coello, F. Luna, and E. Alba. SMPSO: A New PSO-based Metaheuristic for Multi-objective Optimization. In *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM'2009)*, pages 66–73, Nashville, TN, USA, March 30 - April 2 2009. IEEE Press. ISBN 978-1-4244-2764-2.

[24] N. Padhye, J. Branke, and S. Mostaghim. Empirical Comparison of MOPSO Methods - Guide Selection and Diversity Preservation -. In *2009 IEEE Congress on Evolutionary Computation (CEC'2009)*, pages 2516–2523, Trondheim, Norway, May 2009. IEEE Press.

[25] D. H. Phan and J. Suzuki. R2-IBEA: R2 Indicator Based Evolutionary Algorithm for Multiobjective Optimization. In *IEEE Congress on Evolutionary Computation (CEC'2013)*, pages 1836–1845, 2013.

[26] R. C. Purshouse and P. J. Fleming. On the Evolutionary Optimization of Many Conflicting Objectives. *IEEE Transactions on Evolutionary Algorithms*, 11(6):770–784, December 2007.

[27] M. Reyes Sierra and C. A. Coello Coello. Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance. In C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 505–519, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.

[28] M. Reyes-Sierra and C. A. Coello Coello. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.

[29] R. Santana, M. Pontes, and C. Bastos-Filho. A multiple objective particle swarm optimization approach using crowding distance and roulette wheel. In *Intelligent Systems Design and Applications, 2009. ISDA '09. Ninth International Conference on*, pages 237–242, Nov 2009.

[30] H. Scheffé. Experiments with mixtures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 20(2):344–360, 1958.

[31] Y.-Y. Tan, Y.-C. Jiao, H. Li, and X.-K. Wang. MOEA/D + uniform design: A new version of MOEA/D for optimization problems with many objectives. *Computers & Operations Research*, 40(6):1648–1660, June 2013.

[32] P. K. Tripathi, S. Bandyopadhyay, and S. K. Pal. An Adaptive Multi-Objective Particle Swarm Optimization algorithm with Constraint Handling. In B. K. Panigrahi, Y. Shi, and M.-H. Lim, editors, *Handbook of Swarm Intelligence. Concepts, Principles and Applications*, pages 221–239. Springer-Verlag, Berlin, Germany, 2011.

[33] Y. Wang and K. T. Fang. Number-Theoretic Method in Applied statistics (II). *Chinese Annals of Mathematics. Serie B*, 11:859–914, 1990.

[34] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas. Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2337–2344, June 2013.

[35] Q. Zhang and H. Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, December 2007.

[36] E. Zitzler, D. Brockhoff, and L. Thiele. The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicator Via Weighted Integration. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007*, pages 862–876, Matshushima, Japan, March 2007. Springer. Lecture Notes in Computer Science Vol. 4403.