Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art

Margarita Reyes-Sierra and Carlos A. Coello Coello
CINVESTAV-IPN (Evolutionary Computation Group)
Electrical Engineering Department, Computer Science Section
Av. IPN No. 2508, Col. San Pedro Zacatenco
México D.F. 07300, MÉXICO
`mreyes@computacion.cs.cinvestav.mx`
`ccoello@cs.cinvestav.mx`

Abstract- The success of the Particle Swarm Optimization (PSO) algorithm as a single-objective optimizer (mainly when dealing with continuous search spaces) has motivated researchers to extend the use of this bio-inspired technique to other areas. One of them is multi-objective optimization. Despite the fact that the first proposal of a Multi-Objective Particle Swarm Optimizer (MOPSO) is over six years old, a considerable number of other algorithms have been proposed since then. This paper presents a comprehensive review of the various MOPSOs reported in the specialized literature. As part of this review, we include a classification of the approaches, and we identify the main features of each proposal. In the last part of the paper, we list some of the topics within this field that we consider as promising areas of future research.

1 Introduction

Optimization problems that have more that one objective function are rather common in every field or area of knowledge. In such problems, the objectives to be optimized are normally in conflict with respect to each other, which means that there is no single solution for these problems. Instead, we aim to find good “trade-off” solutions that represent the best possible compromises among the objectives.

Particle Swarm Optimization (PSO) is a heuristic search technique (which is considered as an evolutionary algorithm by its authors [18]) that simulates the movements of a flock of birds which aim to find food. The relative simplicity of PSO and the fact that is a population-based technique have made it a natural candidate to be extended for multi-objective optimization.

Moore and Chapman proposed the first extension of the PSO strategy for solving multi-objective problems in an unpublished manuscript from 1999[1] [41]. After this early attempt, a great interest to extend PSO arose among researchers, but interestingly, the next proposal was not published until 2002. Nevertheless, there are currently over twenty five different proposals of MOPSOs reported in the specialized literature. This paper provides the first survey of this work, attempting to classify these proposals and to delineate some of the potential research paths that could be followed in the future by researchers in this area.

The remainder of this paper is organized as follows. In Section 2, we provide some basic concepts from multi-objective optimization required to make the paper self-contained. Section 3 presents an introduction to the PSO strategy and Section 4 presents a brief discussion about extending the PSO strategy for solving multi-objective problems. A complete review of the MOPSO approaches is provided in Section 5. We provide a brief discussion about the convergence properties of PSO and MOPSO in Section 6. In Section 7, possible paths of future research are discussed and, finally, we present our conclusions in Section 8.

2 Basic Concepts

We are interested in solving problems of the type[2]:

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

subject to:

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

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

where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, ..., k$ are the objective functions and $g_i, h_j : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, ..., m$, $j = 1, ..., p$ are the constraint functions of the problem.

To describe the concept of optimality in which we are interested, we will introduce next a few definitions.

Definition 1. Given two vectors $\vec{x}, \vec{y} \in \mathbb{R}^k$, we say that $\vec{x} \leq \vec{y}$ if $x_i \leq y_i$ for $i = 1, ..., k$, and that $\vec{x}$ **dominates** $\vec{y}$ (denoted by $\vec{x} \prec \vec{y}$) if $\vec{x} \leq \vec{y}$ and $\vec{x} \neq \vec{y}$.

Figure 1 shows a particular case of the **dominance relation** in the presence of two objective functions.

Definition 2. We say that a vector of decision variables $\vec{x} \in \mathcal{X} \subset \mathbb{R}^n$ is **nondominated** with respect to $\mathcal{X}$, if there does not exist another $\vec{x}' \in \mathcal{X}$ such that $\vec{f}(\vec{x}') \prec \vec{f}(\vec{x})$.

Definition 3. We say that a vector of decision variables $\vec{x}^* \in \mathcal{F} \subset \mathbb{R}^n$ ($\mathcal{F}$ is the feasible region) is **Pareto-optimal** if it is nondominated with respect to $\mathcal{F}$.

Definition 4. The **Pareto Optimal Set** $\mathcal{P}^*$ is defined by:

$$\mathcal{P}^* = \{\vec{x} \in \mathcal{F} | \vec{x} \text{ is Pareto-optimal}\}$$

---

[1] This paper may be found in the EMOO repository located at: `http://delta.cs.cinvestav.mx/~ccoello/EMOO/`

[2] Without loss of generality, we will assume only minimization problems.

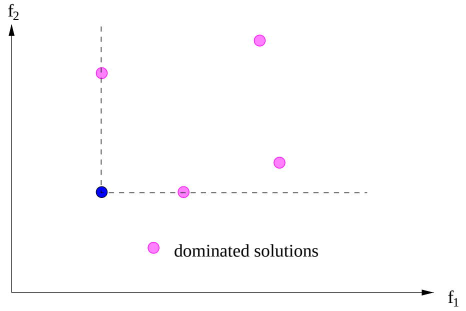

Figure 1: Dominance relation in a bi-objective space.

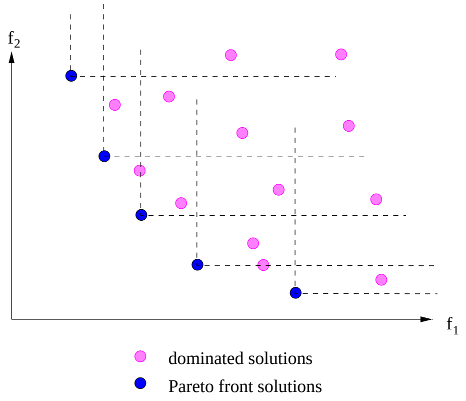

Figure 2: The Pareto front of a set of solutions in a two objective space.

Definition 5. The **Pareto Front** $\mathcal{PF}^*$ is defined by:

$$\mathcal{PF}^* = \{\vec{f}(\vec{x}) \in \mathbb{R}^k | \vec{x} \in \mathcal{P}^*\}$$

Figure 2 shows a particular case of the **Pareto front** in the presence of two objective functions.

We thus wish to determine the Pareto optimal set from the set $\mathcal{F}$ of all the decision variable vectors that satisfy (2) and (3). Note however that in practice, not all the Pareto optimal set is normally desirable (e.g., it may not be desirable to have different solutions that map to the same values in objective function space) or achievable.

3 Particle Swarm Optimization

James Kennedy and Russell C. Eberhart [30] originally proposed the PSO algorithm for optimization. PSO is a population-based search algorithm based on the simulation of the social behavior of birds within a flock. Although originally adopted for balancing weights in neural networks [17], PSO soon became a very popular global optimizer, mainly in problems in which the decision variables are real numbers[3] [32, 19].

According to Angeline [3], we can make two main distinctions between PSO and an evolutionary algorithm:

1. Evolutionary algorithms rely on three mechanisms in their processing: parent representation, selection of individuals and the fine tuning of their parameters. In contrast, PSO only relies on two mechanisms, since PSO does not adopt an explicit selection function. The absence of a selection mechanism in PSO is compensated by the use of leaders to guide the search. However, there is no notion of offspring generation in PSO as with evolutionary algorithms.

2. A second difference between evolutionary algorithms and PSO has to do with the way in which the individuals are manipulated. PSO uses an operator that sets the velocity of a particle to a particular direction. This can be seen as a directional mutation operator in which the direction is defined by both the particle's personal best and the global best (of the swarm). If the direction of the personal best is similar to the direction of the global best, the angle of potential directions will be small, whereas a larger angle will provide a larger range of exploration. In contrast, evolutionary algorithms use an mutation operator that can set an individual in any direction (although the relative probabilities for each direction may be different). In fact, the limitations exhibited by the directional mutation of PSO has led to the use of mutation operators similar to those adopted in evolutionary algorithms.

Two are the key aspects by which we believe that PSO has become so popular:

1. The main algorithm of PSO is relatively simple (since in its original version, it only adopts one operator for creating new solutions, unlike most evolutionary algorithms) and its implementation is, therefore, straightforward. Additionally, there is plenty of source code of PSO available in the public domain (see for example: `http://www.swarmintelligence.org/codes.php`).

2. PSO has been found to be very effective in a wide variety of applications, being able to produce very good results at a very low computational cost [32, 20].

In order to establish a common terminology, in the following we provide some definitions of several technical terms commonly used:

- **Swarm:** Population of the algorithm.

- **Particle:** Member (individual) of the swarm. Each particle represents a potential solution to the problem being solved. The position of a particle is determined by the solution it currently represents.

- ***pbest*** (*personal best*)**:** Personal best position of a given particle, so far. That is, the position of the particle that has provided the greatest success (measured in terms of a scalar value analogous to the fitness adopted in evolutionary algorithms).

---

[3] It is worth noting that there have been proposals to use alternative encodings with PSO (e.g., binary [31] and integer [26]), but none of them has been as popular as the original proposal in which the algorithm operates using vectors of real numbers.

- ***lbest*** (*local best*)**:** Position of the best particle member of the neighborhood of a given particle.
- ***gbest*** (*global best*)**:** Position of the best particle of the entire swarm.
- **Leader:** Particle that is used to guide another particle towards better regions of the search space.
- **Velocity (vector):** This vector drives the optimization process, that is, it determines the direction in which a particle needs to "fly" (move), in order to improve its current position.
- **Inertia weight:** Denoted by W, the inertia weight is employed to control the impact of the previous history of velocities on the current velocity of a given particle.
- **Learning factor:** Represents the attraction that a particle has toward either its own success or that of its neighbors. Two are the learning factors used: C_1 and C_2. C_1 is the *cognitive* learning factor and represents the attraction that a particle has toward its own success. C_2 is the *social* learning factor and represents the attraction that a particle has toward the success of its neighbors. Both, C_1 and C_2, are usually defined as constants.
- **Neighborhood topology:** Determines the set of particles that contribute to the calculation of the *lbest* value of a given particle.

In PSO, particles are "flown" through hyperdimensional search space. Changes to the position of the particles within the search space are based on the social-psychological tendency of individuals to emulate the success of other individuals.

The position of each particle is changed according to its own experience and that of its neighbors. Let $\vec{x}_i(t)$ denote the position of particle p_i, at time step t. The position of p_i is then changed by adding a velocity $\vec{v}_i(t)$ to the current position, i.e.:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{4}$$

The velocity vector reflects the socially exchanged information and, in general, is defined in the following way:

$$\begin{aligned}\vec{v}_i(t) = W\vec{v}_i(t-1) + C_1 r_1(\vec{x}_{pbest_i} - \vec{x}_i(t))\\ +C_2 r_2(\vec{x}_{leader} - \vec{x}_i(t))\end{aligned} \tag{5}$$

where and $r_1, r_2 \in [0,1]$ are random values.

Particles tend to be influenced by the success of anyone they are connected to. These neighbors are not necessarily particles which are close to each other in parameter (decision variable) space, but instead are particles that are close to each other based on a neighborhood topology that defines the social structure of the swarm [32].

Particles can be connected to each other in any kind of neighborhood topology represented as a graph. In the following, list some typical neighborhood graphs used in PSO.

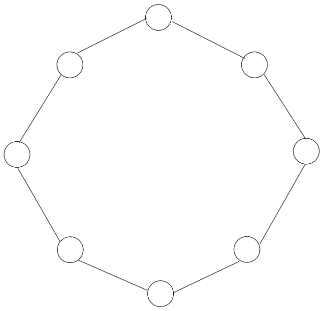

Figure 3: The ring neighborhood topology that represents the *local* best scheme, when $k = 2$. In this common *local* best case, each particle is affected only by its two immediate adjacent neighbors. Each circle represents a particle.

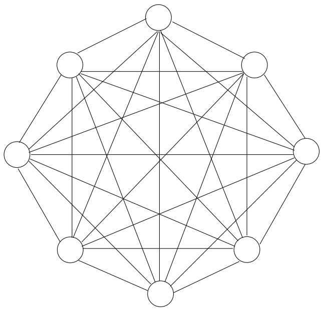

Figure 4: The fully connected graph represents the *fully connected* neighborhood topology (each circle represents a particle). All members of the swarm are connected to one another.

- **Empty graph:** In this topology, particles are isolated. Each particle is connected only with itself, and it compares its current position only to its own best position found so far (*pbest*) [19]. In this case, $C_2 = 0$ in Equation 5.
- **Local best:** In this topology, each particle is affected by the best performance of its k immediate neighbors. Particles are influenced by the best position within their neighborhood (*lbest*), as well as their own past experience (*pbest*) [19]. When $k = 2$, this structure is equivalent to a ring topology such as the one shown in Figure 3. In this case, *leader=lbest* in Equation 5.
- **Fully connected graph:** This topology is the opposite of the *empty graph*. The fully connected topology connects all members of the swarm to one another. Each particle uses its history of experiences in terms of its own best solution so far (*pbest*) but, in addition, the particle uses the position of the best particle from the entire swarm (*gbest*). This structure is also called *star* topology in the PSO community [19]. See Figure 4. In this case, *leader=gbest* in Equation 5.
- **Star network:** In this topology, one particle is connected to all others and they are connected to only that one (called *focal* particle) [19]. See Figure 5.

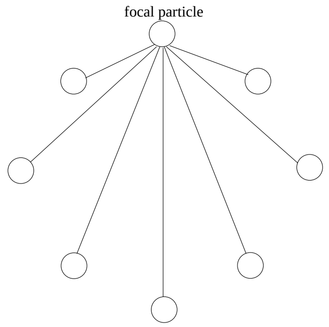

Figure 5: The star network topology (each circle represents a particle). Tne focal particle is connected to all the other particles and they are connected to only that one.

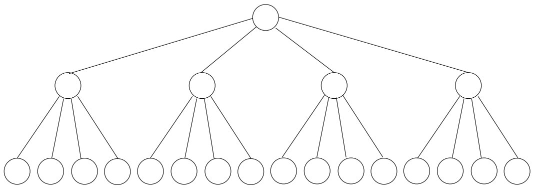

Figure 6: The tree network topology (each circle represents a particle). All particles are arranged in a tree. A particle is influenced by its own best position so far (*pbest*) and by the best position of the particle that is directly above in the tree. Here, we show an example of a topology defined by a regular tree with a height equal to 3, degree equal to 4 and a total of 21 particles.

Particles are isolated from one another, as all information has to be communicated through the *focal* particle. The *focal* particle compares performances of all particles in the swarm and adjusts its trajectory towards the best of them. That performance is eventually communicated to the rest of the swarm. This structure is also called *wheel* topology in the PSO community. In this case, *leader=focal* in Equation 5.

- **Tree network:** In this topology, all particles are arranged in a tree and each node of the tree contains exactly one particle [28]. See Figure 6. A particle is influenced by its own best position so far (*pbest*) and by the best position of the particle that is directly above in the tree (parent). If a particle at a child node has found a solution that is better than the best so far solution of the particle at the parent node, both particles are exchanged. In this way, this topology offers a dynamic neighborhood. This structure is also called *hierarchical* topology in the PSO community. In this case, *leader=* $pbest_{parent}$ in Equation 5.

The neighborhood topology is likely to affect the rate fo convergence as it determines how much time it takes to the particles to find out about the location of good (better) regions of the search space. For example, since in the *fully connected* topology all particles are connected to each other, all particles receive the information of the best solution from the entire swarm at the same time. Thus, when using such topology, the swarm tends to converge more rapidly than when using *local best* topologies, since in this case, the information of the best position of the swarm takes a longer time to be transferred. However, for the same reason, the *fully connected* topology is also more susceptible to suffer premature convergence (i.e., to converge to local optima) [20].

```
Begin
    Initialize swarm
    Locate leader
    $g = 0$
    While $g < gmax$
        For each particle
            Update Position (Flight)
            Evaluation
            Update $pbest$
        EndFor
        Update leader
        $g$++
    EndWhile
End
```

Figure 7: Pseudocode of the general PSO algorithm.

Figure 7 shows the way in which the general (single-optimization) PSO algorithm works. First, the swarm is initialized. This initialization includes both positions and velocities. The corresponding *pbest* of each particle is initialized and the leader is located (usually the *gbest* solution is selected as the leader). Then, for a maximum number of iterations, each particle flies through the search space updating its position (using (4) and (5)) and its *pbest* and, finally, the leader is updated too.

4 Particle Swarm Optimization for Multi-Objective Problems

In order to apply the PSO strategy for solving multi-objective optimization problems, it is obvious that the original scheme has to be modified. As we saw in Section 2, the solution set of a problem with multiple objectives does not consist of a single solution (as in global optimization). Instead, in multi-objective optimization, we aim to find a set of different solutions (the so-called Pareto optimal set). In general, when solving a multi-objective problem, three are the main goals to achieve [73]:

1. Maximize the number of elements of the Pareto optimal set found.

2. Minimize the distance of the Pareto front produced by our algorithm with respect to the true (global) Pareto front (assuming we know its location).

3. Maximize the spread of solutions found, so that we can have a distribution of vectors as smooth and uniform as possible.

Given the population-based nature of PSO, it is desirable to produce several (different) nondominated solutions with a single run. So, as with any other evolutionary algorithm, the three main issues to be considered when extending PSO to multi-objective optimization are [13]:

1. How to select particles (to be used as leaders) in order to give preference to nondominated solutions over those that are dominated?
2. How to retain the nondominated solutions found during the search process in order to report solutions that are nondominated with respect to all the past populations and not only with respect to the current one? Also, it is desirable that these solutions are well spread along the Pareto front.
3. How to maintain diversity in the swarm in order to avoid convergence to a single solution?

As we could see in the previous section, when solving single-objective optimization problems, the leader that each particle uses to update its position is completely determined once a neighborhood topology is stablished. However, in the case of multi-objective optimization problems, each particle might have a set of different leaders from which just one can be selected in order to update its position. Such set of leaders is usually stored in a different place from the swarm, that we will call *external archive* [4]: This is a repository in which the nondominated solutions found so far are stored. The solutions contained in the external archive are used as leaders when the positions of the particles of the swarm have to be updated. Furthermore, the contents of the external archive is also usually reported as the final output of the algorithm.

Figure 8 shows the way in which a general MOPSO algorithm works. We have marked with *italics* the processes that make this algorithm different from the general PSO algorithm for single objective optimization.

First, the swarm is initialized. Then, a set of leaders is also initialized with the nondominated particles from the swarm. As we mentioned before, the set of leaders is usually stored in an external archive. Later on, some sort of quality measure is calculated for all the leaders in order to select (usually) one leader for each particle of the swarm. At each generation, for each particle, a leader is selected and the flight is performed. Most of the existing MOPSOs apply some sort of mutation operator[5] after performing the flight. Then, the particle is evaluated and its corresponding $pbest$ is updated. A new particle replaces its $pbest$ particle usually when this particle is dominated or if both are incomparable (i.e., they are both nondominated with respect to each other). After all the particles have been updated, the set of leaders is updated, too. Finally, the quality measure of the set of leaders is re-calculated. This process is repeated for a certain (usually fixed) number of iterations.

```
Begin
    Initialize swarm
    Initialize leaders in an external archive
    Quality(leaders)
    $g = 0$
    While $g < gmax$
        For each particle
            Select leader
            Update Position (Flight)
            Mutation
            Evaluation
            Update $pbest$
        EndFor
        Update leaders in the external archive
        Quality(leaders)
        $g$++
    EndWhile
    Report results in the external archive
End
```

Figure 8: Pseudocode of a general MOPSO algorithm.

As we can see, and given the characteristics of the PSO algorithm, the issues that arise when dealing with multi-objective problems are related with two main algorithmic design aspects [64]:

1. Selection and updating of leaders:
    - How to select a single leader out of set of nondominated solutions which are all equally good? Should we select this leader in a random way or should we use an additional criterion (to promote diversity, for example)?
    - How to select the particles that should remain in the external archive from one iteration to another?
2. Creation of new solutions:
    - How to promote diversity through the two main mechanisms to create new solutions: updating of positions (Equations 4 and 5) and mutation (turbulence) operator.

These issues are discussed in more detail in the next subsections.

4.1 Leaders in Multi-Objective Optimization

Since the solution of a multi-objective problem consist of a set of equally good solutions, it is evident that the concept of leader traditionally adopted in PSO has to be changed.

A few researches have avoided the problem of defining a new concept of leader for multi-objective problems by adopting aggregating functions (i.e., weighted sums of the objectives) or approaches that optimize each objective separately. We will briefly discuss these approaches in Section 5.

---

[4] This *external archive* is also used by many Multi-Objective Evolutionary Algorihtms (MOEAs).

[5] The mutation operators adopted in the PSO literature have also been called *turbulence* operators.

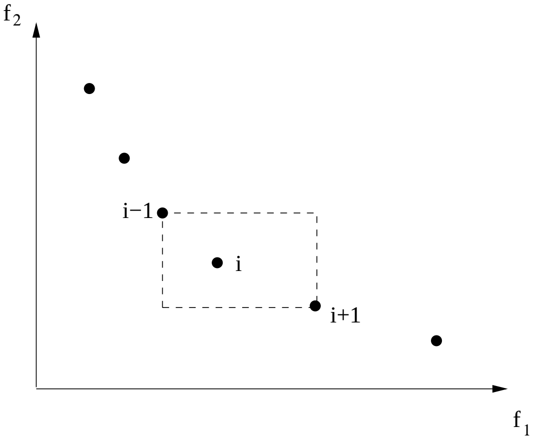

Figure 9: The nearest neighbor density estimator for an example with two objective functions. Particles with a larger value of this estimator are preferred.

However, it is important to indicate that the majority of the currently proposed MOPSO approaches redefine the concept of leader.

As we mentioned before, the selection of a leader is a key component when designing a MOPSO approach. The most straightforward approach is to consider every nondominated solution as a new leader and then, just one leader has to be selected. In this way, a *quality* measure that indicates how good is a leader is very important. Obviously, such feature can be defined in several different ways. As we will see in Section 5, there exist already different proposals to deal with this issue.

One posible way of defining such *quality* measure can be related to density measures. Promoting diversity may be done through this process by means of mechanisms based on some *quality* measures that indicate the closeness of the particles within the swarm.

Several authors have proposed leader selection techniques that are based on density measures. In order to help understanding the specific approaches that are going to be described later on, we present here two of the most important density measures used in the area of multi-objective optimization:

- **Nearest neighbor density estimator** [16]. The nearest neighbor density estimator gives us an idea of how crowded are the closest neighbors of a given particle, in objective function space. This measure estimates the perimeter of the cuboid formed by using the nearest neighbors as the vertices. See Figure 9.

- **Kernel density estimator** [22, 15]: When a particle is sharing resources with others, its fitness is degraded in proportion to the number and closeness to particles that surround it within a certain perimeter. A neighborhood of a particle is defined in terms of a parameter called σ_{share} that indicates the radius of the neighborhood. Such neighborhoods are called *niches*. See Figure 10.

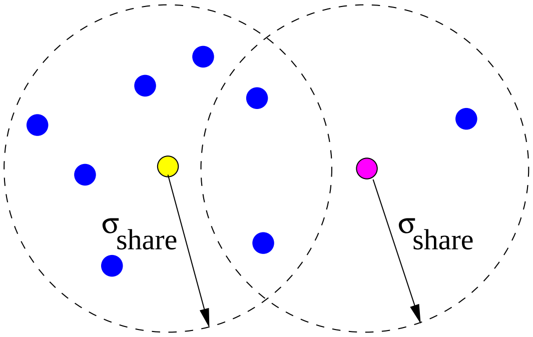

Figure 10: For each particle, a niche is defined. Particles whose niche is less crowded are preferred.

4.2 Retaining and Spreading Nondominated Solutions

As we mentioned before, it is important to retain the nondominated solutions found along *all* the search process so that we can report at the end those solutions that are nondominated with respect to all the previous populations. This is important not only for pragmatic reasons, but also for theoretical ones [54].

The most straightforward way of retaining solutions that are nondominated with respect to all the previous populations (or swarms) is to use an external archive. Such an archive will allow the entrance of a solution only if: (a) it is nondominated with respect to the contents of the archive or (b) it dominates any of the solutions within the archive (in this case, the dominated solutions have to be deleted from the archive).

This approach has, however, the drawback of increasing the size of the archive very quickly. This is an important issue because the archive has to be updated at each generation. Thus, this update may become very expensive, computationally speaking, if the size of the archive grows too much. In the worst case, all members of the swarm may wish to enter into the archive, at each generation. Thus, the corresponding updating process, at each generation, has a complexity of $O(kN^2)$, where N is the size of the swarm and k is the number of objectives. In this way, the complexity of the updating process for the complete run is of $O(kMN^2)$, where M is the total number of iterations.

Thus, mainly due to practical reasons, archives tend to be bounded [13], which makes necessary the use of an additional criterion to decide which nondominated solutions to retain, once the archive is full. In evolutionary multi-objective optimization, researchers have adopted different techniques to prune the archive (e.g., clustering [74] and geographical-based schemes that place the nondominated solutions in cells in order to favor less crowded cells when deleting in-excess nondominated solutions [34]). However, the use of an archive introduces additional issues: for example, do we impose additional criteria to enter the archive instead of just using nondominance (e.g., use the distribution of solutions as an additional criterion)?

Note that, strictly speaking, three archives should be used when extending PSO for multi-objective optimization:

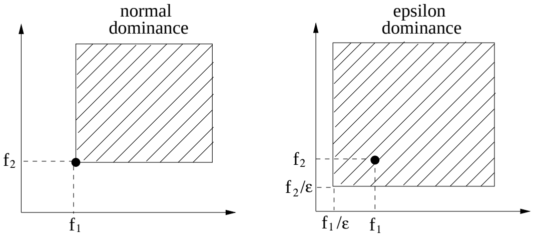

Figure 11: To the left, we can see the area that is dominated by a certain solution. To the right, we graphically depict the ε-dominance concept. In this case, the area being dominated has been extended by a value proportional to the parameter ε (which is defined by the user).

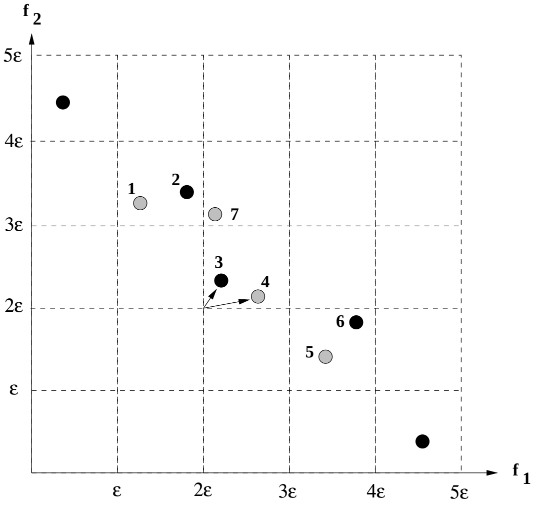

Figure 12: An example of the use of ε-dominance in an external archive. Solution 1 dominates solution 2, therefore solution 1 is preferred. Solutions 3 and 4 are incomparable. However, solution 3 is preferred over solution 4, since solution 4 is the closer to the lower lefthand corner represented by point (2ε,2ε). Solution 5 dominates solution 6, therefore solution 5 is preferred. Solution 7 is not accepted since its box, represented by point (2ε,3ε) is dominated by the box represented by point (2ε,2ε).

one for storing the global best solutions, one for the personal best values and a third one for storing the local best (if applicable). However, in practice, few authors report the use of more than one archive in their MOPSOs.

Besides the use of an external file, it is also possible to use a plus selection in which parents compete with their children and those which are nondominated (and possibly comply with some additional criterion such as providing a better distribution of solutions) are selected for the following generation. In the case of PSO, a plus selection involves selecting from a merge of two consecutive swarms.

More recently, other researchers have proposed the use of relaxed forms of dominance. The main one adopted in PSO has been ε-dominance [36], which is illustrated in Figure 11. The main use of this concept in multi-objective PSO has been to filter solutions in the external archive. By using ε-dominance, we define a set of boxes of size ε and only one nondominated solution is retained for each box (e.g., the one closest to the lower lefthand corner). This is illustrated in Figure 12, for a bi-objective case. The use of ε-dominance, as proposed in [36] and illustrated in Figure 12, guarantees that the retained solutions are nondominated with respect to all solutions generated during the run. It is worth noting, however, that, when using ε-dominance, the size of the final external archive depends on the ε-value, which is normally a user-defined parameter [36]. Mostaghim and Teich [43] have found that when comparing ε-dominance against existing clustering techniques for fixing the archive size, the ε-dominance method can find solutions much faster (computationally speaking) than the clustering technique with a comparable (and even better in some cases) convergence and diversity.

4.3 Promoting Diversity while Creating New Solutions

It is well-known that one of the most important features of the PSO algorithm is its fast convergence. This is a positive feature as long as we don't have premature convergence (i.e., convergence to a local optimum).

Premature convergence is caused by the rapid loss of diversity within the swarm. So, the appropriate promotion of diversity in PSO is a very important issue in order to control its (normally fast) convergence.

As we mentioned in Section 4.1, when adopting PSO for solving multi-objective optimization problems, it is possible to promote diversity through the selection of leaders. However, this can be also done through the two main mechanisms used for creating new solutions:

1. **Updating of positions.** As we mentioned in Section 3, the use of different neighborhood topologies determines how fast is the process of transfering the information through the swarm (since a neighborhood determines who the *leader* particle is in Equation 5). Since in a *fully connected* topology all particles are connected with each other, the information is transferred faster than in the case of a *local best* or a *tree* topology, since in these cases particles have smaller neighborhoods. Under the same argument, a specified neighborhood topology also determines how fast is diversity lost within the swarm. Since in a *fully connected* topology, the tranfer of information is fast, when using this topology, diversity within the swarm is also lost rapidly. In this way, topologies that define neighborhoods smaller than the entire swarm for each particle can also preserve diversity within the swarm a longer time.

   On the other hand, diversity can also be promoted by means of the inertia weight (W in Equation 5). As it was defined in Section 3, the inertia weight is employed to control the impact of the previous history of velocities on the current velocity. Thus, the inertia weight influences the trade-off between global (wide-ranging) and local (nearby) exploration abilities [58]. A large inertia weight facilitates global exploration (searching new areas) while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area. The value of the inertia weight may vary during the optimization process. Shi [59] asserted that by linearly decreasing the inertia weight from a relatively large value to a small one through the course of the PSO run, the PSO tends to have more global search ability at the beginning of the run and have more local search ability near the end of the run. On the other hand, Zheng et al. [72] argue that either global or local search ability associates with a small inertia and that a large inertia weight provides the algorithm more chances to be stabilized. In this way, inspired on the process of the simulated annealing algorithm, the authors proposed to use an increasing inertia weight through the PSO run.

The addition of velocity to the current position to generate the next position is similar to the mutation operator in evolutionary algorithms, except that "mutation" in PSO is guided by the experience of a particle and that of its neighbors. In other words, PSO performs "mutation" with a "conscience" [58].

2. **Through the use of a mutation (or turbulence) operator**.

   As mentioned in the previous section, when a particle updates its position, a mutation with "conscience" occurs. Sometimes, however, some unconciousness or "craziness", as called by Kennedy and Eberhart in the original proposal of PSO [30], is needed. Craziness, also referred as turbulence, reflects the change in a particle's flight which is out of its control [21].

   In general, when a swarm stagnates, that is, when the velocities of the particles are almost zero, it becomes unable to generate new solutions which might lead the swarm out of this state. This behavior can lead to the whole swarm being trapped in a local optimum from which it becomes impossible to escape. Since the global best individual attracts all members of the swarm, it is possible to lead the swarm away from a current location by mutating a single particle if the mutated particle becomes the new global best. This mechanism potentially provides a means both of escaping local optima and of speeding up the search [62].

   In this way, the use of a mutation operator is very important in order to escape from local optima and to improve the exploratory capabilities of PSO. When a solution is chosen to be mutated each component is then mutated (randomly changed) or not with certain probability. Actually, different mutation operators have been proposed that mutate components of either the position or the velocity of a particle.

   In our experience, the choice of a good mutation operator is a difficult task that has a significant impact on performance. On the other hand, once we have selected a specific mutation operator another difficult task is to decide how much mutation to apply: with how much probability, in which moments of the process, in which specific component of a particle, etc.

   Several proposed approaches have used different mutation operators, however, there are also approaches which do not use any kind of mutation operator and that show good performance. So, the use of mutation is an issue that certainly deserves a more careful study.

5 A Taxonomy of Approaches

The taxonomy that we propose to classify the current MOPSOs is the following:

- Aggregating approaches
- Lexicographic ordering
- Sub-Population approaches
- Pareto-based approaches
- Combined approaches
- Other approaches

We will discuss next each of these types of approaches. Also, Table 1 summarize all the different approaches and indicates their most important features.

5.1 Aggregating Approaches

Under this category we consider approaches that combine (or "aggregate") all the objectives of the problem into a single one. In other words, the multi-objective problem is transformed into a single-objective one. This is not a new idea, since aggregating functions can be derived from the well-known Kuhn-Tucker conditions for nondominated solutions [35].

- Parsopoulos and Vrahatis [50]: This algorithm adopts three types of aggregating functions: (1) a conventional linear aggregating function (were weights are fixed during the run), (2) a dynamic aggregating function (were weights are gradually modified during the run) and (3) the bang bang weighted aggregation approach (were weights are abruptly modified during the run)[6] [29]. In all cases, the authors adopt the *fully connected* topology.

- Baumgartner et al. [6]: This approach, based on the *fully connected* topology, uses linear aggregating functions. In this case, the swarm is equally partitioned into n subswarms, each of which uses a different set of weights and evolves into the direction of its own swarm leader. The approach adopts a gradient technique to identify the Pareto optimal solutions.

---

[6] This approach has the peculiarity of being able to generate nonconvex portions of the Pareto front, which is something that traditional linear aggregating functions cannot do [14].

5.2 Lexicographic Ordering

In this method, the user is asked to rank the objectives in order of importance. The optimum solution is then obtained by minimizing the objective functions separately, starting with the most important one and proceeding according to the assigned order of importance of the objectives [40]. Lexicographic ordering tends to be useful only when few objective functions are used (two or three), and it may be sensitive to the ordering of the objectives [10].

- Hu and Eberhart [24]: In this algorithm, only one objective is optimized at a time using a scheme similar to lexicographic ordering [13]. This approach adopts the *ring* (*local best*) topology. No external archive is adopted in this case. However, in a further version of this approach [25], the authors incorporate an external archive (called "extended memory") and introduce some further improvements to their dynamic neighborhood PSO approach.

5.3 Sub-Population Approaches

These approaches involve the use of several subpopulations as single-objective optimizers. Then, the subpopulations somehow exchange information or recombine among themselves aiming to produce trade-offs among the different solutions previously generated for the objectives that were separately optimized.

- Parsopoulos et al. [49] studied a parallel version of the Vector Evaluated Particle Swarm (VEPSO) method for multi-objective problems. VEPSO is a multi-swarm variant of PSO, which is inspired on the Vector Evaluated Genetic Algorithm (VEGA) [56, 57]. In VEPSO, each swarm is evaluated using only one of the objective functions of the problem under consideration, and the information it possesses for this objective function is communicated to the other swarms through the exchange of their best experience (*gbest* particle). The authors argue that this process can lead to Pareto optimal solutions.

- Chow and Tsui [8]: In this paper, the authors use PSO as an autonomous agent response learning algorithm. For that sake, the authors propose to decompose the award function of the autonomous agent into a set of local award functions and, in this way, to model the response extraction process as a multi-objective optimization problem. A modified PSO called "Multi-Species PSO" is introduced by considering each objective function as a species swarm. A communication channel is established between the neighboring swarms for transmitting the information of the best particles, in order to provide guidance for improving their objective values. Also, the authors use the flight formula of the *fully connected* topology, but include a *neighbor swarm reference velocity*. Such velocity is directly related with the best particle within each subswarm (similar to *lbest*).

5.4 Pareto-Based Approaches

These approaches use leader selection techniques based on Pareto dominance. The basic idea of all the approaches considered here is to select as leaders to the particles that are nondominated with respect to the swarm. Note however, that several variations of the leader selection scheme are possible since most authors adopt additional information to select leaders (e.g., information provided by a density estimator) in order to avoid a random selection of a leader from the current set of nondominated solutions.

- Moore and Chapman [41]: This algorithm was presented in an unpublished document and it is based on Pareto dominance. The authors emphasize the importance of performing both an individual and a group search (a cognitive component and a social component). In this approach, the personal best (*pbest*) of a particle is a list of all the nondominated solutions it has found in its trajectory. When selecting a *pbest*, a particle from the list is randomly chosen. Since the *ring* topology is used, when selecting the best particle of the neighborhood, the solutions contained in the *pbest* lists are compared, and a nondominated solution with respect to the neighborhood is chosen. The authors don't indicate how they choose the *lbest* particle when more that one nondominated solution is found in the neigborhood.

- Ray and Liew [53]: This algorithm (based on a *fully connected* topology) uses Pareto dominance and combines concepts of evolutionary techniques with the particle swarm. The approach uses a nearest neighbor density estimator to promote diversity (by means of a roulette selection scheme of leaders based on this value) and a multilevel sieve to handle constraints (for this, the authors adopt the constraint and objective matrices proposed in some of their previous research [52]). The set of leaders maintained by the authors can be considered an external archive.

- Fieldsend and Singh [21]: This approach uses an unconstrained elite external archive (in which a special data structure called "dominated tree" is adopted) to store the nondominated individuals found along the search process. The archive interacts with the primary population in order to define leaders. The selection of the *gbest* for a particle in the swarm is based on the structure defined by the dominated tree. First, a composite point of the tree is located based on dominance relations, and then the closest member (in objective function space) of the composite point is chosen as the leader. On the other hand, a set of personal best particles found (nondominated) is also maintained for each swarm member, and the selection is performed uniformly. This approach also uses a "turbulence" operator that is basically a mutation operator that acts on the velocity value used by the PSO algorithm.

- Coello et al. [11, 12]: This proposal is based on the idea of having an external archive in which every particle will deposit its flight experiences after each flight cycle. The updates to the external archive are performed considering a geographically-based system defined in terms of the objective function values of each particle. The search space explored is divided on hypercubes. Each hypercube receives a fitness value based on the number of particles it contains. Thus, in order to select a leader for each particle of the swarm, a roulette-wheel selection using these fitness values is first applied, to select the hypercube from which the leader will be taken. Once a hypercube has been selected, the leader is randomly chosen. This approach also uses a mutation operator that acts both on the particles of the swarm, and on the range of each design variable of the problem to be solved.

In more recent work, Toscano and Coello [66] use the concept of Pareto dominance to determine the flight direction of a particle. The authors adopt clustering techniques to divide the population of particles into several swarms. This aims to provide a better distribution of solutions in decision variable space. Each sub-swarm has its own set of leaders (nondominated particles). In each sub-swarm, a PSO algorithm is executed (leaders are randomly chosen) and, at some point, the different sub-swarms exchange information: the leaders of each swarm are migrated to a different swarm in order to variate the selection pressure. Also, this approach does not use an external archive since elitism in this case is an emergent process derived from the migration of leaders.

- Srinivasan and Hou [61]: This approach, called Particle Swarm Inspired Evolutionary Algorithm (PS-EA), is a hybrid between PSO and an evolutionary algorithm. The main aim is to use EA operators (mutation, for example) to emulate the workings of PSO mechanisms, based on a *fully connected* topology. Since the authors mention that the final swarm constitutes the final solution (Pareto front), we conclude that a plus selection is performed at each iteration of the algorithm. Also, the authors use a niche count and a Pareto ranking approach in order to assign a fitness value to the particles of the swarm. However, the selection technique used is not described in the paper.

- Mostaghim and Teich [44]: They propose a *sigma* method in which the leader for each particle is selected in order to improve the convergence and diversity of a MOPSO approach. The idea of the sigma method is similar to compromise programming [13]. In order to select a leader for each particle of the swarm, a *sigma* value is assigned to each particle of the swarm and of the external archive. Each particle of the swarm selects as its leader the particle of the external archive with the closest sigma value. The use of the sigma values makes the selection pressure of PSO even higher, which may cause premature convergence in some cases. The authors also use a "turbulence" operator, which is applied on decision variable space. This approach has been successfully applied to the molecular force field parametrization problem [42].

  In further work, Mostaghim and Teich [43] studied the influence of ϵ-dominance [36] on MOPSO methods. ϵ-dominance is compared with existing clustering techniques for fixing the external archive size and the solutions are compared in terms of computational time, convergence and diversity. The results show that the ϵ-dominance method can find solutions much faster than the clustering technique with a comparable (and even better in some cases) convergence and diversity. The authors suggest a new density measure (sigma method) inspired on their previous work [44]. Also, based on the idea that the initial external archive from which the particles have to select a leader has influence on the diversity of solutions, the authors propose the use of successive improvements adopting a previous external archive of solutions. In this way, in more recent work, Mostaghim and Teich [45] propose a new method called *covering*MOPSO (cvMOPSO) which retakes this idea. This method works in two phases. In phase 1, a MOPSO algorithm is run with an external archive with restricted size and the goal is to obtain a good approximation of the Pareto-front. In the phase 2, the non-dominated solutions obtained from the phase 1 are considered as the input external archive of the cvMOPSO. The particles in the swarm of the cvMOPSO are divided into subswarms around each non-dominated solution after the first generation. The task of the subswarms is to cover the gaps between the non-dominated solutions obtained from the phase 1. No restrictions on the archive size are imposed in the phase 2.

- Bartz et al. [5]: This approach starts from the idea of introducing elitism (through the use of an external archive) into PSO. Different methods for selecting and deleting particles (leaders) from the archive are analyzed to generate a satisfactory approximation of the Pareto front. The deletion methods analyzed are based on the contribution of each particle to the diversity of the Pareto front. Selecting methods are either inversely related to the fitness value or based on the previous success of each particle. The authors provide some statistical analysis in order to assess the impact of each of the parameters used by their approach.

- Li [37]: This approach is based on a *fully connected* topology and incorporates the main mechanisms of the NSGA-II [16] to the PSO algorithm. In this approach, once a particle has updated its position, instead of comparing the new position only against the *pbest* position of the particle, all the *pbest* positions of the swarm and all the new positions recently obtained are combined in just one set (given a total of $2N$ solutions, where N is the size of the swarm). Then, the approach selects the best solutions among them to conform the next swarm (by means of a nondominated sorting). The author doesn't specify which values are assigned to the velocity of *pbest* positions, in order to consider them as particles. This approach also selects the leaders randomly from the leaders set (stored in an external archive) among the best of them, based on two different mechanisms: a niche count and a nearest neighbor density estimator. This approach uses a mutation operator that is applied at each iteration step only to the particle with the smallest density estimator value (or the largest niche count).

- Reyes and Coello [60]: This approach is based on Pareto dominance and the use of a nearest neighbor density estimator for the selection of leaders (by means of a binary tournament). This proposal uses two external archives: one for storing the leaders currently used for performing the flight and another for storing the final solutions. The density estimator factor is used to filter out the list of leaders whenever the maximum limit imposed on such list is exceeded. Only the leaders with the best density estimator values are retained. On the other hand, the concept of ε-dominance is used to select the particles that will remain in the archive of final solutions. Additionally, the authors propose a scheme in which they subdivide the population (or swarm) into three different subsets. A different mutation operator is applied to each subset. Note however, that for all other purposes, a single swarm is considered (e.g., for selecting leaders). This approach is based on a *fully connected* topology.

- Alvarez-Benitez et al. [2]: The authors propose methods based exclusively on Pareto dominance for selecting leaders from an unconstrained nondominated (external) archive. Three different selection techniques are presented: One technique that explicitly promotes diversity (called *Rounds* by the authors), one technique that explicitly promotes convergence (called *Random*) and finally one technique that is a weighted probabilistic method (called *Prob*) and forms a compromise between *Random* and *Rounds*. Also, the authors propose and evaluate four mechanisms for confining particles to the feasible region, that is, constraint-handling methods. The authors show that probabilistic selection favoring archival particles that dominate few particles provides good convergence towards the Pareto front while properly covering it at the same time. Also, they conclude that allowing particles to explore regions close to the constraint boundaries is important to ensure convergence to the Pareto front. This approach uses a turbulence factor that is added to the position of the particles with certain probability.

- Ho et al. [23]: The authors propose a novel formula for updating velocity and position particles, based on three main modifications to the known flight formula for the *fully connected* topology. First, since the authors argue that the random factors r_1 and r_2 in Equation 5 are not completely independent, they propose to use: $r_2 = 1 - r_1$. Second, they propose to incorporate the term $(1 - W)$ in the second and third terms of Equation 5, where $W = rnd(0, 1)$. Third (and last), under the argument of allowing a particle to fly sometimes back, the authors propose to allow the first term of Equation 5 being negative with a 50% probability. On the other hand, the authors introduce a "craziness" operator in order to promote diversity within the swarm. This "craziness" operator is applied (with certain probability) to the velocity vector before updating the position of a particle. Finally, the authors introduce one external archive for each particle and one global external archive for the whole swarm. The archive of each particle stores the latest Pareto solutions found by the particle and the global archive stores the current Pareto optimal set. Every time a particle updates its position, it selects its personal best from its own archive and the global best from the global archive. In both cases, the authors use a roulette selection mechanism based on the fitness values of the particles (assigned using the mechanism originally proposed by Zitzler et al. [74], for the SPEA algorithm) and on an "age" variable that the authors introduce and that is increased at each generation.

- Villalobos-Arias et al. [68]: The authors propose a new mechanism to promote diversity in multi-objective optimization problems. Although the approach is independent of the search engine adopted, they incorporate it into the MOPSO proposed in [12]. The new approach is based on the use of stripes that are applied on the objective function space. Based on an analysis for a bi-objective problem, the main idea of the approach is that the Pareto front of the problem is "similar" to the line determined by the minimal points of the objective functions. In this way, several points (that the authors call stripe centers) are distributed uniformly along such line, and the particles of the swarm are assigned to the nearest stripe center. When using this approach for solving multi-objective problems with PSO, one leader is used in each stripe. Such leader is selected minimizing a weighted sum of the minimal points of the objective functions. The authors show that their approach overcomes the drawbacks on other popular mechanisms such as ε-dominance [36] and the sigma method proposed in [44].

- Salazar-Lechuga and Rowe [55]: The main idea of this approach is to use PSO to guide the search with the help of niche counts (applied on objective function space) [22] to spread the particles along the Pareto front. The approach uses an external archive to store the best particles (nondominated particles) found by the algorithm. Since this external archive helps to guide the search, the niche count is calculated for each of the particles in the archive and the leaders are chosen from this set by means of an stochastic sampling method (roulette wheel). Also, the niche count is used as a criterion to update the external archive. Each time the archive is full and a new particle wants to get in, its niche count is compared with the niche count of the worst solution of the archive. If the new particle is better than the worst particle, then the new particle enters into the archive and the worst particle is deleted. Niche counts are updated when inserting or deleting a particle from the archive.

- Raquel and Naval [51]: As in [60], this approach incorporates the concept of nearest neighbor density estimator for selecting the global best particle and also for deleting particles from the external archive of nondominated solutions. When selecting a leader, the archive of nondominated solutions is sorted in descending order with respect to the density estimator, and a particle is randomly chosen from the top part of the list. On the other hand, when the external archive is full, it is again sorted in descending order with respect to the density estimator value and a particle is randomly chosen to be deleted, from the bottom part of the list. This approach uses the mutation operator proposed in [12] in such a way that it is applied only during a certain number of generations at the beginning of the process. Finally, the authors adopt the constraint-handling technique from the NSGA-II [16].

- Zhao and Cao [71]: This approach is very similar to the proposal of Coello and Lechuga [11]. However, the authors indicate that they maintain two external archives, but one of them is actually a list that keeps the *pbest* particle for each member of the swarm. The another external archive stores the nondominated solutions found along the evolutionary process. This truncated archive is similar to the adaptive grid of PAES [34]. The authors apply their approach to solve the economic load dispatch problem. With this aim, they employ a fuzzy-based mechanism to extract the best compromise solution, in which they incorporate the preferences of the decision maker. The approach adopts a linear membership function to represent the goals of each objective function. This membership function is adopted to modify the ranking of the nondominated solutions as to focus the search on the single solution that attains the maximum membership in the fuzzy set.

  Janson and Merkle [27] proposed a hybrid particle swarm optimization algorithm for multi-objective optimization, called ClustMPSO. ClustMPSO combines the PSO algorithm with clustering techniques to divide all particles into several subswarms. For this aim, the authors use the K-means algorithm. Each subswarm has its own nondominated front and the total nondominated front is obtained from the union of the fronts of all the subswarms. Each particle randomly selects its neighborhood best ($lbest$) particle from the nondominated front of the swarm to which it belongs. Also, a particle only selects a new $lbest$ particle when the current is no longer a nondominated solution. On the other hand, the personal best ($pbest$) of each particle is updated based on dominance relations. Finally, the authors define that a subswarm is dominated when none of its particles belongs to the total nondominated front. In this way, when a subwarm is dominated for a certain number of consecutive generations, the subswarm is relocated. The proposed algorithm is tested on an artificial multi-objective optimization function and on a real-world problem from biochemistry, called the molecular docking problem. The authors reformulate the molecular docking problem as a multi-objective optimization problem and, in this case, the updating of the $pbest$ particle is also based on the weighted sum of the objectives of the problem. ClustMPSO outperforms a well-known Lamarckian Genetic Algorithm that had been previously adopted to solve such problem.

5.5 Combined Approaches

- Mahfouf et al. [39]: The authors propose an Adaptive Weighted PSO (AWPSO) algorithm, in which the velocity is modified by including an acceleration term that increases as the number of iterations increases. This aims to enhance the global search ability at the end of run and to help the algorithm to jump out of local optima. Also, a weighted aggregating function is introduced within the algorithm for performance evaluation and to guide the selection of the personal and global bests. The authors use dynamic weights to generate Pareto optimal solutions. When the population is losing diversity, a mutation operator is applied to the positions of certain particles and the best of them are retained. Finally, the authors include a nondominated sorting algorithm to select the particles from one iteration to the next. Since plus selection is adopted, an external archive is not necessary in this case. This approach is applied in the optimal design of heat-treated alloy steels.

- Xiao-hua et al. [69]: The authors propose an Intelligent Particle Swarm Optimization (IPSO) algorithm for multi-objective problems based on an Agent-Environment-Rules (AER) model to provide an appropriate selection pressure to propel the swarm population towards the Pareto optimal front. In this model, the authors modify the *fully connected* flight formula including the *lbest* position of the neighborhood of each particle. The neighborhood of a particle is determined by a lattice-like topology. On the other hand, each particle is taken as an agent particle with the ability of memory, communication, response, cooperation and self-learning. Each particle has its position, velocity and energy, which is related to its fitness. All particles live in a latticelike environment, which is called an agent lattice, and each particle is fixed on a lattice-point. In order to survive in the system, they compete or cooperate with their neighbors so that they can gain more resources (increase energies). Each particle has the ability of cloning itself, and the number of clones produced depends of the energy of the particle. General agent particles and latency agent particles (those who have smaller energy but contain certain features—e.g., favoring diversity—that make them good candidates to be cloned) will be cloned. The aim of the clonal operator (which is modeled in the clonal selection theory also adopted with artificial immune systems [46]) is to increase the competition among particles, maintain diversity of the swarm and improving the convergence of the process. Also, a clonal mutation operator is used. Leaders are selected based on the energy values of the particles. Finally, this approach adopts an external archive in order to store the nondominated solutions found throughout the run and to provide the final solution set.

5.6 Other Approaches

Here, we consider the approaches that could not fit any of the main categories previously described.

- Li [38]: This author proposes the *maximinPSO*, which uses a fitness function derived from the maximin strategy proposed by Balling [4] to determine Pareto-domination. The author shows that one advantage of this approach is that no additional clustering or niching technique is needed, since the maximin fitness of a solution can tell us not only if a solution is dominated or not, but also if it is clustered with other solutions, i.e., the approach also provides diversity information. In this approach, for each particle, a different leader is selected for each of the decision variables to conform a single global best. Leaders (stored in an external archive) are randomly selected based on the maximin fitness.

- Zhang et al. [70]: This approach (based on a *fully connected* topology) attempts to improve the selection of $gbest$ and $pbest$ when the velocity of each particle is updated. For each objective function, there exists both a $gbest$ and a $pbest$ for each particle. In order to update the velocity of a particle, the algorithm defines the $gbest$ of a particle as the average of the complete set of $gbest$ particles. Analogously, the $pbest$ is computed using either a random choice or the average from the complete set of $pbest$ values. This choice depends on the dispersion degree between the $gbest$ and $pbest$ values of each particle.

6 Convergence Properties of PSO and MOPSO

Recently, some theoretical studies about the convergence properties of PSO have been published. As in the case of many evolutionary algorithms, these studies have concluded that the performance of the PSO is sensitive to control parameter choices [20].

Most of the theoretical studies are based on simplified PSO models, in which a swarm consisting of one particle of one dimension is studied. The *pbest* and *gbest* particles are assumed to be constant throughout the process. Also, the terms $\phi_1 = c_1 r_1$, $\phi_2 = c_2 r_2$ (used in Equation 5) are assumed to be constant. Under these conditions, particle trajectories and convergence of the swarm have been analyzed.

In the theoretical studies developed about PSO, convergence has been defined as follows:

Definition 6. Considering the sequence of global best solutions $\{gbest_t\}_{t=0}^{\infty}$, we say that the swarm converges iff

$$\texttt{lim}_{t \to \infty} gbest_t = p$$

where p is an arbitrary position in the search space.

Since p refers to an arbitrary solution, Definition 6 does not mean convergence to a local or global optimum.

The first studies on the convergence properties of PSO were developed by Ozcan and Mohan [47, 48]. Ozcan and Mohan studied a PSO under the conditions previously described but, in addition, their model did not consider the inertia weight. They concluded that, when $0 < \phi < 4$, where $\phi = \phi_1 + \phi_2$, the trajectory of a particle is a sinusoidal wave where the initial conditions and parameter choices determine the amplitude and frequency of the wave. Also, they concluded that the periodic nature of the trajectory may cause a particle to repeatedly search regions of the search space already visited, unless another particle in its neighborhood finds a better solution.

In [67], van den Bergh developed a model of PSO under the same conditions, but considering the inertia weight. Van den Berg proved that, when $w > \frac{1}{2}(c_1 + c_2) - 1$, the particle converges to the point

$$\frac{\phi_1 pbest + \phi_2 gbest}{\phi_1 + \phi_2}.$$

In this way, if $c_1 = c_2$, the particle converges to the point

$$\frac{pbest + gbest}{2}.$$

Since these conclusions were obtained under the assumption of ϕ_1 and ϕ_2 being constants, van den Bergh generalized his model considering the stochastic nature of ϕ_1 and ϕ_2. In this case, he concluded (assuming uniform distributions) that the particle then converges to the position:

$$(1-a)pbest + agbest$$

where $a = \frac{c_2}{c_1 + c_2}$. In this way, van den Bergh showed that a particle converges to a weighted average between its personal best and its neighborhood best position.

	neighborhood topology	leaders selection based on	external archive	dynamic W	mutation operator
Aggregating approaches					
Parsopolous and Vrahatis [50]	*fully connected*	single-objective	no	yes $(1.0 \to 0.4)$	no
Baumgartner et al. [6]	*fully connected*	single-objective	no	no	no
Lexicographic ordering					
Hu and Eberhart [24]	*ring*	single-objective	no	yes $rnd(0.5, 1.0)$	no
Hu et al. [25]	*ring*	single-objective	yes	yes $rnd(0.5, 1.0)$	no
Sub-Population approaches					
Parsopoulos et al. [49]	*fully connected*	single-objective	yes	no	no
Chow and Tsui [8]	*fully connected*	single-objective	no	no	no
Pareto-Based approaches					
Moore and Chapman [41]	*ring*	dominance	no	no	no
Ray and Liew [53]	*fully connected*	density estimator	yes	no	no
Fieldsend and Singh [21]	*fully connected*	dominance & closeness	yes	no	yes
Coello et al. [11, 12]	*fully connected*	density of solutions	yes	no	yes
Toscano and Coello [66]	*fully connected*	randomly	no	no	no
Srinivasan and Hou [61]	*fully connected*	niche count & dominance	no	no	yes
Mostaghim and Teich [44]	*fully connected*	*sigma* value	yes	no	yes
Mostaghim and Teich [43]	*fully connected*	*sigma* value	yes	no	yes
Mostaghim and Teich [45]	*fully connected*	*sigma* value	yes	no	yes
Bartz et al. [5]	*fully connected*	density of solutions; success	yes	no	no
Li [37]	*fully connected*	niche count; density estimator	yes	yes $(1.0 \to 0.4)$	yes
Reyes and Coello [60]	*fully connected*	density estimator	yes	yes $rnd(0.1, 0.5)$	yes
Alvarez-Benitez et al. [2]	*fully connected*	dominance	yes	no	yes
Ho et al. [23]	*fully connected*	fitness & *age*	yes	yes *proposed*	yes
Villalobos-Arias et al. [68]	*fully connected*	*stripes*	yes	no	yes
Salazar-Lechuga and Rowe [55]	*fully connected*	niche count	yes	no	no
Raquel and Naval [51]	*fully connected*	density estimator	yes	no	yes
Zhao and Cao [71]	*fully connected*	fuzzy membership	yes	no	no
Janson and Merkle [27]	*fully connected*	random	yes	no	no
Combined approaches					
Mahfouf et al. [39]	*fully connected*	single-objective	no	yes $rnd(0.15, 1.0)$	yes
Xiao-hua et al. [69]	*fully connected & lattice*	*energy* value	yes	yes $(0.6 \to 0.2)$	yes
Other approaches					
Li [38]	*fully connected*	*maximin* fitness	yes	yes $(1.0 \to 0.4)$	no
Zhang et al. [70]	*fully connected*	composite leader	no	yes $(0.8 \to 0.4)$	no

Table 1: Complete list of the MOPSO proposals reviewed. For each proposal, we indicate the corresponding neighborhood topology adopted, leader selection scheme used and whether the approach incorporates some dynamic scheme for the inertia weight (W), an external archive and a mutation operator.

As we said before, in order to ensure convergence, the condition $w > \frac{1}{2}(c_1 + c_2) - 1$ must hold. However, it is possible to choose values of c_1, c_2 and w such that the condition is violated, and the swarm still converges [67]: if

$$\phi_{ratio} = \frac{\phi_{crit}}{c_1 + c_2}$$

is close to 1.0, where $\phi_{crit} = \texttt{sup}\ \{\phi \mid 0.5\phi - 1 < w\}$, $\phi \in (0, c_1 + c_2]$, the swarm has convergent behavior. This implies that the trajectory of the particle will converge *most of the time*, occasionally taking divergent steps.

The studies developed by Ozcan and Mohan, and van der Bergh, consider trajectories that are not constricted. In [9], Clerc and Kennedy provide a theoretical analysis of particle behavior in which they introduce a constriction coefficient whose objective is to prevent the velocity from growing out of bounds.

As we could see, the convergence of PSO has been proved. However, we can only ensure the convergence of PSO to the best position visited by all the particles of the swarm. In order to ensure convergence to the local or global optimum, two conditions are necessary:

1. The $gbest_{t+1}$ solution can be no worse than the $gbest_t$ solution (monotonic condition).
2. The algorithm must be able to generate a solution in the neighborhood of the optimum with nonzero probability, from any solution x of the search space.

In [67], van den Bergh provides a proof to show that the basic PSO is not a local (neither global) optimizer. This is due to the fact that, although PSO satisfies the monotonic condition indicated above, once the algorithm reaches the state where $x = pbest = gbest$ for all particles in the swarm, no further progress will be made. The problem is that this state may be reached before $gbest$ reaches a minimum, whether be local or global. The basic PSO is therefore said to prematurely converge. In this way, the basic PSO algorithm is not a local (global) search algorithm, since it has no guaranteed convergence to a local (global) minimum from an arbitrary initial state.

Also, van den Bergh suggests two ways of extending PSO in order to make it a global search algorithm. The first is related to the generation of new random solutions. In general, the introduction of a mutation operator is useful. Nevertheless, forcing PSO to perform a random search in an area surrounding the global best position, that is, forcing the global best position to change in order to prevent stagnation (by means of a hill-climbing search, for example), is also a suitable mechanism [20]. On the other hand, van den Bergh also proposes to use a “multi-start PSO”, in which when the algorithm has converged (under some criteria), it records the best solution found and the particles are randomly reinitialized.

To the best of our knowledge, until this date, there are no studies about the convergence properties of MOPSOs. From the discussion previously provided, we can conclude that it is possible to ensure convergence, by correctly setting the parameters of the flight formula. But, as in the case of single-optimization, such property does not ensure the convergence to the true Pareto front, in this case. In the case of multi-objective optimization, we may conclude that we still need conditions (1) and (2), to ensure convergence. However, in this case, condition (1) may change to:

1. The solutions contained in the external archive at iteration $t+1$ should be nondominated with respect to the solutions generated in all iterations t, $0 \leq t \leq t+1$, so far (monotonic condition).

The use of the ε-dominance based archiving as proposed in [36] ensures this condition, but the normal dominance-based strategies do not. In this way, given a MOPSO approach, and assuming it satisfies condition (1), it remains to explore if it satisfies condition (2), to ensure global convergence to the true Pareto front.

7 Future Research Paths

As we have seen, despite the fact that MOPSOs started to be developed less than ten years ago, the growth of this field has exceeded even the most optimistic expectations. By looking at the papers that we reviewed, the core of the work on MOPSOs has focused on algorithmic aspects, but there is much more to do in this area. In this section, we will provide some insights regarding some topics that we believe that are worth investigating within the next few years:

- **Emphasis on Efficiency:** The current MOPSOs are not algorithms particularly complex (in terms of their data structures, memory management and so on), and are quite effective (more than state-of-the-art multi-objective evolutionary algorithms in some cases). So, why to make things more complicated regarding algorithmic design? Is there room for new developments in this regard? We believe that there is, but we have to focus our work in a new direction. For example, few people have tried to exploit the very high convergence rate commonly associated with PSO to design an “ultra-efficient” MOPSO. It would be very useful (for real-world applications) to have a MOPSO that could produce reasonably good approximations of the Pareto front of multi-objective optimization problems with 20 or 30 decision variables with less than 5000 fitness function evaluations. A first attempt to design such a type of MOPSO is reported in [64] but more work in that direction is certainly expected, since this topic has been recently explored with other types of multi-objective evolutionary algorithms as well [33].
- **Self-Adaptation of Parameters in MOPSOs:** The design of MOPSOs with no parameters that have to be fine-tuned by the user is another topic that is worth studying. In evolutionary multi-objective optimization in general, the use of self-adaptation or on-line adaptation mechanisms is scarce (see for example [63, 1, 7]), and we are only aware of one multi-objective evolutionary algorithm which was designed to be parameterless: the microGA^2 [65]. The design

of a parameterless MOPSO requires a careful study of the velocity update formula adopted in PSO, and an assessment of the impact of each of its components in the performance of a MOPSO. Even the inertia and learning factors which are normally assumed constants in PSO may benefit from an on-line adaptation mechanism when dealing with multi-objective optimization problems.[7]

- **Theoretical Developments:** There is not much theoretical work on PSO in general (see for example [9]) and, therefore, the lack of research on theoretical aspects of MOPSOs is, by no means, surprising. It would be interesting to perform a theoretical study of the run-time and convergence properties of a MOPSO (see Section 6). Other aspects such as the fitness landscapes and dynamics of a MOPSO are also very attractive theoretical research topics.

- **Applications:** Evidently, no algorithm will ever be useful if we cannot find a good application for it. MOPSOs have been used in a few applications (see Section 5), but not so extensively as other multi-objective evolutionary algorithms. The reason may be that MOPSOs are younger and less known than, for example, multi-objective genetic algorithms. However, a well-designed MOPSO may be quite useful in real-world applications, mainly if, as we mentioned before, its very fast convergence rate is properly exploited. At some point in the near future, we believe that there will be an important growth in the number of applications that adopt MOPSOs as their search engine.

8 Conclusions

We have reviewed the state-of-the-art regarding extensions of PSO to handle multiple objectives. We have started by providing a short introduction to PSO in which we described its basic algorithm and its main topologies. We have also indicated the main issues that have to be considered when extending PSO to multi-objective optimization, and then we have analyzed each of them in more detail.

We have also proposed a taxonomy to classify the current techniques reported in the specialized literature, and we have provided a survey of approaches based on such a taxonomy.

Finally, we have provided some topics that seem (from the authors' perspective) as very promising paths for future research in this area. Considering the current rate of growth of this area, we expect a lot of more activity within the next few years. However, the switch to new areas different from pure algorithm development may attract newcomers to this field and may contribute to keep it alive for several more years.

Acknowledgments

The authors gratefully acknowledge the comments from Marco Laumanns and the anonymous reviewers, which greatly helped them to improve the contents of this paper.

The first author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Section of the Electrical Engineering Department at CINVESTAV-IPN. The second author acknowledges support from CONACyT project number 45683.

Bibliography

[1] Hussein A. Abbass. The self-adaptive pareto differential evolution algorithm. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 831–836, Piscataway, New Jersey, May 2002. IEEE Service Center.

[2] Julio E. Alvarez-Benitez, Richard M. Everson, and Jonathan E. Fieldsend. A MOPSO algorithm based exclusively on pareto dominance concepts. In *Third International Conference on Evolutionary Multi-Criterion Optimization, EMO 2005.*, pages 459–473, Guanajuato, México, 2005. LNCS 3410, Springer-Verlag.

[3] Peter J. Angeline. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In V.W. Porto, N. Saravanan, D. Waagen, and A.E. Eiben, editors, *Evolutionary Programming VII. 7th International Conference, EP 98*, pages 601–610. Springer. Lecture Notes in Computer Science Vol. 1447, San Diego, California, USA, March 1998.

[4] Richard Balling. The maximin fitness function; multiobjective city and regional planning. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Second International Conference on Evolutionary Multi-Criterion Optimization, EMO 2003*, pages 1–15, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[5] Thomas Bartz-Beielstein, Philipp Limbourg, Konstantinos E. Parsopoulos, Michael N. Vrahatis, Jörn Mehnen, and Karlheinz Schmitt. Particle swarm optimizers for pareto optimization with enhanced archiving techniques. In *Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 1780–1787, Canberra, Australia, December 2003. IEEE Press.

[6] U. Baumgartner, Ch. Magele, and W. Renhart. Pareto optimality and particle swarm optimization. *IEEE Transactions on Magnetics*, 40(2):1172–1175, March 2004.

[7] Dirk Büche, Sibylle Müller, and Petro Koumoutsakos. Self-adaptation for multi-objective evolutionary algorithms. In Carlos M. Fonseca, Peter J. Fleming, Eckart

---

[7] Readers interested in this topic may be interested in looking at the work of Maurice Clerc, available at: `http://clerc.maurice.free.fr/pso/`.

Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Second International Conference on Evolutionary Multi-Criterion Optimization, EMO 2003*, pages 267–281, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[8] Chi-kin Chow and Hung-tat Tsui. Autonomous agent response learning by a multi-species particle swarm optimization. In *Congress on Evolutionary Computation (CEC'2004)*, volume 1, pages 778–785, Portland, Oregon, USA, June 2004. IEEE Service Center.

[9] Maurice Clerc and James Kennedy. The particle swarm–explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, February 2002.

[10] Carlos A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, August 1999.

[11] Carlos A. Coello Coello and Maximino Salazar Lechuga. MOPSO: A proposal for multiple objective particle swarm optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1051–1056, Piscataway, New Jersey, May 2002. IEEE Service Center.

[12] Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, June 2004.

[13] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002.

[14] Indraneel Das and John Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69, 1997.

[15] Kalyanmoy Deb and David E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, San Mateo, California, June 1989. George Mason University, Morgan Kaufmann Publishers.

[16] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[17] Russell C. Eberhart, Roy Dobbins, and Patrick K. Simpson. *Computational Intelligence PC Tools*. Morgan Kaufmann Publishers, 1996.

[18] Russell C. Eberhart and Yuhui Shi. Comparison between genetic algorithms and particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, and A.E. Eibe, editors, *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pages 611–619. Springer-Verlag, March 1998.

[19] Andries P. Engelbrecht, editor. *Computational Intelligence: An Introduction*. John Wiley & Sons, England, 2002.

[20] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2005.

[21] Jonathan E. Fieldsend and Sameer Singh. A multiobjective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. In *Proceedings of the 2002 U.K. Workshop on Computational Intelligence*, pages 37–44, Birmingham, UK, September 2002.

[22] David E. Goldberg and Jon Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associates, 1987.

[23] S.L. Ho, Yang Shiyou, Ni Guangzheng, Edward W.C. Lo, and H.C. Wong. A particle swarm optimization-based method for multiobjective design optimizations. *IEEE Transactions on Magnetics*, 41(5):1756–1759, May 2005.

[24] Xiaohui Hu and Russell Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1677–1681, Piscataway, New Jersey, May 2002. IEEE Service Center.

[25] Xiaohui Hu, Russell C. Eberhart, and Yuhui Shi. Particle swarm with extended memory for multiobjective optimization. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pages 193–197, Indianapolis, Indiana, USA, April 2003. IEEE Service Center.

[26] Xiaohui Hu, Russell C. Eberhart, and Yuhui Shi. Swarm intelligence for permutation optimization: A case study on n-queens problem. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pages 243–246, Indianapolis, Indiana, USA, 2003. IEEE.

[27] Stefan Janson and Daniel Merkle. A new multiobjective particle swarm optimization algorithm using clustering applied to automated docking. In María J. Blesa, Christian Blum, Andrea Roli, and Michael Sampels, editors, *Hybrid Metaheuristics, Second International Workshop, HM 2005*, pages 128–142, Barcelona, Spain, August 2005. Springer. Lecture Notes in Computer Science Vol. 3636.

[28] Stefan Janson and Martin Middendorf. A hierarchical particle swarm optimizer. In *Congress on Evolutionary Computation (CEC'2003)*, pages 770–776, Camberra, Australia, 2003. IEEE Press.

[29] Yaochu Jin, Tatsuya Okabe, and Bernhard Sendhoff. Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how? In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 1042–1049, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[30] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, New Jersey, 1995. IEEE Service Center.

[31] James Kennedy and Russell C. Eberhart. A discrete binary version of the particle swarm algorithm. In *Proceedings of the 1997 IEEE Conference on Systems, Man, and Cybernetics*, pages 4104–4109, Piscataway, New Jersey, 1997. IEEE Service Center.

[32] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.

[33] Joshua Knowles and Evan J. Hughes. Multiobjective optimization on a budget of 250 evaluations. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Third International Conference on Evolutionary Multi-Criterion Optimization, EMO 2005*, pages 176–190, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.

[34] Joshua D. Knowles and David W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[35] Harold W. Kuhn and Albert W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, California, 1951. University of California Press.

[36] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282, Fall 2002.

[37] Xiaodong Li. A non-dominated sorting particle swarm optimizer for multiobjective optimization. In Erick Cantú-Paz et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2003)*, pages 37–48. Springer. Lecture Notes in Computer Science Vol. 2723, July 2003.

[38] Xiaodong Li. Better spread and convergence: Particle swarm multiobjective optimization using the maximin fitness function. In Kalyanmoy Deb et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2004)*, pages 117–128, Seattle, Washington, USA, June 2004. Springer-Verlag, Lecture Notes in Computer Science Vol. 3102.

[39] Mahdi Mahfouf, Min-You Chen, and Derek Arturh Linkens. Adaptive weighted particle swarm optimisation for multi-objective optimal design of alloy steels. In *Parallel Problem Solving from Nature - PPSN VIII*, pages 762–771, Birmingham, UK, September 2004. Springer-Verlag. Lecture Notes in Computer Science Vol. 3242.

[40] Kaisa Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, Massachusetts, 1999.

[41] Jacqueline Moore and Richard Chapman. Application of particle swarm to multiobjective optimization. Department of Computer Science and Software Engineering, Auburn University, 1999.

[42] S. Mostaghim, M. Hoffmann, P.H. König, Th. Frauenheim, and J. Teich. Molecular force field parametrization using multi-objective evolutionary algorithms. In *2004 Congress on Evolutionary Computation (CEC'2004)*, volume 1, pages 212–219, Portland, Oregon, USA, June 2004. IEEE Service Center.

[43] Sanaz Mostaghim and Jürgen Teich. The role of ε-dominance in multi objective particle swarm optimization methods. In *Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 1764–1771, Canberra, Australia, December 2003. IEEE Press.

[44] Sanaz Mostaghim and Jürgen Teich. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pages 26–33, Indianapolis, Indiana, USA, April 2003. IEEE Service Center.

[45] Sanaz Mostaghim and Jürgen Teich. Covering pareto-optimal fronts by subswarms in multi-objective particle swarm optimization. In *Congress on Evolutionary Computation (CEC'2004)*, volume 2, pages 1404–1411, Portland, Oregon, USA, June 2004. IEEE Service Center.

[46] Leandro Nunes de Castro and Jonathan Timmis. *An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm*. Springer-Verlag, 2002.

[47] Ender Ozcan and Chilukuri K. Mohan. Analysis of a simple particle swarm optimization system. In *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 253–258, 1998.

[48] Ender Ozcan and Chilukuri K. Mohan. Particle swarm optimization: Surfing the waves. In *Congress on Evolutionary Computation (CEC'1999)*, pages 1939–1944, Washington D.C., USA, 1999. IEEE Press.

[49] Konstantinos E. Parsopoulos, Dimitris K. Tasoulis, and Michael N. Vrahatis. Multiobjective optimization using parallel vector evaluated particle swarm optimization. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)*, volume 2, pages 823–828, Innsbruck, Austria, February 2004. ACTA Press.

[50] Konstantinos E. Parsopoulos and Michael N. Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'2002)*, pages 603–607, Madrid, Spain, 2002. ACM Press.

[51] Carlo R. Raquel and Jr. Prospero C. Naval. An effective use of crowding distance in multiobjective particle swarm optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005)*, pages 257–264, Washington, DC, USA, June 2005. ACM Press.

[52] Tapabrata Ray, Tai Kang, and Seow Kian Chye. An evolutionary algorithm for constrained optimization. In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 771–777, San Francisco, California, 2000. Morgan Kaufmann.

[53] Tapabrata Ray and K.M. Liew. A swarm metaphor for multiobjective design optimization. *Engineering Optimization*, 34(2):141–153, March 2002.

[54] Günter Rudolph. On a multi-objective evolutionary algorithm and its convergence to the pareto set. In *Proceedings of the 5th IEEE Conference on Evolutionary Computation*, pages 511–516, Piscataway, New Jersey, 1998. IEEE Press.

[55] Maximino Salazar-Lechuga and Jonathan Rowe. Particle swarm optimization and fitness sharing to solve multi-objective optimization problems. In *Congress on Evolutionary Computation (CEC'2005)*, pages 1204–1211, Edinburgh, Scotland, UK, September 2005. IEEE Press.

[56] J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.

[57] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[58] Yuhui Shi and Russell Eberhart. Parameter selection in particle swarm optimization. In *Evolutionary Programming VII: Proceedings of the Seventh annual Conference on Evolutionary Programming*, pages 591–600, New York, USA, 1998. Springer-Verlag.

[59] Yuhui Shi and Russell Eberhart. Empirical study of particle swarm optimization. In *Congress on Evolutionary Computation (CEC'1999)*, pages 1945–1950, Piscataway, NJ, 1999. IEEE Press.

[60] Margarita Reyes Sierra and Carlos A. Coello Coello. Improving PSO-based multi-objective optimization using crowding, mutation and ϵ-dominance. In *Third International Conference on Evolutionary Multi-Criterion Optimization, EMO 2005.*, pages 505–519, Guanajuato, México, 2005. LNCS 3410, Springer-Verlag.

[61] Dipti Srinivasan and Tian Hou Seow. Particle swarm inspired evolutionary algorithm (PS-EA) for multiobjective optimization problem. In *Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 2292–2297, Canberra, Australia, December 2003. IEEE Press.

[62] Andrew Stacey, Mirjana Jancic, and Ian Grundy. Particle swarm optimization with mutation. In *Proceedings of the Congress on Evolutionary Computation*, pages 1425–1430, Camberra, Australia, 2003. IEEE Press.

[63] Kay Chen Tan, Tong Heng Lee, and Eik Fun Khor. Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 5(6):565–588, December 2001.

[64] Gregorio Toscano Pulido. *On the Use of Self-Adaptation and Elitism for Multiobjective Particle Swarm Optimization*. PhD thesis, Computer Science Section, Department of Electrical Engineering, CINVESTAV-IPN, Mexico, September 2005.

[65] Gregorio Toscano Pulido and Carlos A. Coello Coello. The micro genetic algorithm 2: Towards online adaptation in evolutionary multiobjective optimization. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Second International Conference on Evolutionary Multi-Criterion Optimization, EMO 2003*, pages 252–266, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[66] Gregorio Toscano Pulido and Carlos A. Coello Coello. Using clustering techniques to improve

the performance of a particle swarm optimizer. In Kalyanmoy Deb et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2004)*, pages 225–237, Seattle, Washington, USA, June 2004. Springer-Verlag, Lecture Notes in Computer Science Vol. 3102.

[67] Frans van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[68] Mario Alberto Villalobos-Arias, Gregorio Toscano Pulido, and Carlos A. Coello Coello. A proposal to use stripes to maintain diversity in a multi-objective particle swarm optimizer. In *Proceedings of the 2005 IEEE Swarm Intelligence Symposium*, pages 22–29, Pasadena, California, USA, June 2005. IEEE Press.

[69] Zhang Xiao-hua, Meng Hong-yun, and Jiao Li-cheng. Intelligent particle swarm optimization in multiobjective optimization. In *Congress on Evolutionary Computation (CEC'2005)*, pages 714–719, Edinburgh, Scotland, UK, September 2005. IEEE Press.

[70] L.B. Zhang, C.G. Zhou, X.H. Liu, Z.Q. Ma, and Y.C. Liang. Solving multi objective optimization problems using particle swarm optimization. In *Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 2400–2405, Canberra, Australia, December 2003. IEEE Press.

[71] Bo Zhao and Yi jia Cao. Multiple objective particle swarm optimization technique for economic load dispatch. *Journal of Zhejiang University SCIENCE*, 6A(5):420–427, 2005.

[72] Yong-Ling Zhen, Long-Hua Ma, Li-Yan Zhang, and Ji-Xin Qian. On the convergence analysis and parameter selection in particle swarm optimization. In *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, pages 1802–1807. IEEE Press, 2003.

[73] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

[74] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.