An Introduction to Multi-Objective Particle Swarm Optimizers

Carlos A. Coello Coello*

CINVESTAV-IPN
Evolutionary Computation Group (EVOCINV)
Departamento de Computación, Av. IPN No. 2508
Col. San Pedro Zacatenco, México, D.F. 07360,MEXICO
ccoello@cs.cinvestav.mx

Summary. This paper provides a discussion on the main changes required in order to extend particle swarm optimization to the solution of multi-objective optimization problems. A short discussion of some potential paths for future research in this area is also included.

keywords: particle swarm optimization, multi-objective optimization, metaheuristics.

1 Introduction

There is a wide variety of real-world problems which have two or more (normally conflicting) objectives that we aim to optimize at the same time. Such problems are called multi-objective, and their solution involves the search of solutions that represent the best possible compromise among all the objectives.

Particle Swarm Optimization (PSO) is a bio-inspired metaheuristic that simulates the movements of a flock of birds or fish which seek food. Its relative simplicity (with respect to evolutionary algorithms) have made it a popular optimization approach, and a good candidate to be extended for multi-objective optimization.

The first multi-objective particle swarm optimizer (MOPSO) was proposed by Moore and Chapman in an unpublished manuscript from 1999[2] [11]. This

* The author acknowledges support from CONACyT project no. 103570.

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

paper was published the following year [12], but the actual interest in developing MOPSOs really started in 2002. Due to obvious space limitations, this paper does not intend to provide a survey on MOPSOs (see, for example, [13] for a survey of that sort). Here, we only provide a short review of PSO and the way in which it has to be modified so that it can solve multi-objective optimization problems.

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. In Section 5, possible paths of future research are discussed and, finally, we present our conclusions in Section 6.

2 Basic Concepts

We are interested in solving the so-called *multi-objective optimization problem* (MOP) which has the following form[3]:

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

subject to:

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

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

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ 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.

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

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

Definition 3. We say that a vector of decision variables $\mathbf{x}^* \in \mathcal{F} \subset \mathbb{R}^n$ ($\mathcal{F}$ 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:

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

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

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

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

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). In practice, it is normally the case that only some elements of the Pareto optimal set is required, although multi-objective metaheuristics normally aim to find as many elements of the Pareto optimal set as possible [2].

3 An Introduction to Particle Swarm Optimization

PSO is a population-based metaheuristic which was originally introduced by James Kennedy and Russell C. Eberhart in the mid-1990s [8]. PSO was originally adopted for balancing weights in neural networks, but it soon became a very popular global optimizer, maybe in problems in which the decision variables are real numbers [6, 9].

Although some authors consider PSO as another evolutionary algorithm (EA), other authors such as Angeline [1], make important distinctions between them:

1. EAs rely on three main mechanisms: parents encoding, selection of individuals and fine tuning of their parameters. In contrast, PSO only relies on two mechanisms, since it does not adopt an explicit selection function (this is compensated by the use of leaders to guide the search, but there is no notion of offspring generation in PSO as in EAs).
2. PSO adopts a velocity value for each particle, and this is used to guide the search. The velocity 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). In contrast, EAs use a randomized mutation operator that can set an individual in any direction. Clearly, PSO has a more limited operator, and such limitations have led to several researchers to incorporate a randomized mutation operator.

In order to make this paper self-contained, we provide next a small glossary of terms used by the PSO community:

- **Swarm:** Number of particles adopted (i.e., population size).
- **Particle:** One member (or 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 that it currently represents.
- ***pbest*** (*personal best*)**:** The best position that a given particle has achieved so far. That is, the position of the particle that has provided the greatest success (measured in terms of a scalar value defined by the user, which is analogous to the fitness value adopted in EAs).

- ***lbest*** (*local best*)**:** Position of the best particle member belonging to 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:** The inertial weight (denoted by W) is adopted to control the impact of the previous history of velocities on the current velocity of a given particle.
- **Learning factor:** It represents the attraction that a particle has towards either its own best previous value or that of its neighbors. Two learning factors are adopted in PSO: C_1, which is the *cognitive* learning factor and represents the attraction that a particle has toward its own success, and C_2, which is the *social* learning factor and represents the attraction that a particle has toward the success of its neighbors. Both of them are normally defined as constants.
- **Neighborhood topology:** It determines the way in which particles are interconnected and thus defines the way in which they contribute to the computation of the *lbest* value of a given particle.

In PSO, the position of a particle (within the search space being explored) changes based on its own experience and the success of its neighbors.

Let $\mathbf{x}_i(t)$ denote the position of particle p_i, at time step t. The position of p_i is then changed by adding a velocity $\mathbf{v}_i(t)$ value to the current position of the particle, i.e.:

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

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

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

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

Particles are influenced by the success of any particle connected to them. It is worth noting, however, that the way this influence information is propagated depends on the neighborhood topology adopted. Any of the possible neighborhood topologies that can be adopted in PSO can be represented as a graph. The following are the most commonly adopted neighborhood topologies:

- **Empty graph:** In this topology, each particle is connected only with itself, and it compares its current position only to its own best position found so far (*pbest*) [5]. 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 (*lbest*), as well as by their own past experience (*pbest*) [5]. When $k = 2$, this structure is equivalent to a ring topology. In this case, *leader=lbest* in equation (5).
- **Fully connected graph:** This topology connects all members of the swarm to one another. This structure is also called *star* topology in the PSO community [5]. 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) [5]. 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 [7]. 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) [6].

Figure 1 shows the way in which the general (single-optimization) PSO algorithm works. First, the swarm (both positions and velocities) is randomly initialized. 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 equations (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 MOPs, it is obvious that the original scheme has to be modified. As we saw in Section 2, in multi-objective

```
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
```

Fig. 1. Pseudocode of the general PSO algorithm for single-objective optimization.

optimization, we aim to find not one, but a set of different solutions (the so-called Pareto optimal set). In general, when solving a MOP, the main goals are to converge to the true Pareto front of the problem (i.e., to the solutions that are globally nondominated) and to have such solutions as well-distributed as possible along the Pareto front. 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 main issues to be considered when extending PSO to multi-objective optimization are [2]:

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 just saw, 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, when dealing with MOPs, 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. Only solutions that are nondominated with respect to the

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

contents of the entire archive are retained. 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.

```
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
```

Fig. 2. Pseudocode of a general MOPSO algorithm.

Figure 2 shows the way in which a general MOPSO works. We have marked with *italics* the processes that make this algorithm different from the general PSO algorithm for single objective optimization shown before. 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

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

quality measure of the set of leaders is re-calculated. This process is repeated for a certain (usually fixed) number of iterations.

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 [14]:

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 a mutation (turbulence) operator.

Regarding the selection of leaders, the most simple approach is to adopt aggregating functions (i.e., weighted sums of the objectives) or approaches that optimize each objective separately. However, most researchers redefine the concept of leader, incorporating the definition of Pareto optimality. However, since all the nondominated solutions currently available can be considered as potential leaders, a quality measure is required in order to choose one of them at a given time. Several authors have proposed the use of density measures for this sake. The two most commonly adopted are:

- **Nearest neighbor density estimator** [4]. 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 as vertices the nearest neighbors.
- **Kernel density estimator** [3]: 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 that defines the radius of the neighborhood. Such neighborhoods are called *niches*.

As indicated before, most MOPSOs adopt an external archive that retains solutions that are nondominated with respect to all the previous populations (or swarms). 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).

Mainly due to practical reasons, archives tend to be bounded [2], 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 [15] and geographical-based schemes that place the nondominated solutions in cells in order to favor less crowded cells when deleting in-excess nondominated solutions [10]).

It is worth noting that, strictly speaking, three archives should be used when implementing a MOPSO: 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.

In a MOPSO, diversity can be promoted through the selection of leaders. However, this can be also done through the two main mechanisms used for creating new solutions: (a) updating of positions (topologies that define neighborhoods smaller than the entire swarm for each particle can also preserve diversity within the swarm a longer time), and (b) through the use of a mutation (or turbulence) operator (this will help a MOPSO to escape from local optima).

5 Future Research Paths

Most of the work in this are has focused on algorithm development, but we believe that there are several other topics that constitute very promising paths for future research:

- **Self-Adaptation:** The design of MOPSOs with no parameters that have to be fine-tuned by the user is a topic that is worth studying. The design of a parameterless MOPSO requires a in-depth knowledge of the relationship between its parameters and its performance in problems with different features.
- **Theoretical Developments:** There is an evident lack of research on even the most basic aspects of a MOPSO (e.g., convergence properties, run-time analysis, population dynamics, etc.), but it is expected that some work in this direction will be conducted in the next few years.
- **Applications:** The applications of MOPSOs have steadily grown in the last few years, but more are expected to arise, as MOPSOs become better developed and widespread multi-objective optimization tools.

6 Conclusions

This paper has provided a review of the basic concepts of the PSO algorithm, including its basic equation, neighborhood topologies and leader selection mechanisms. Then, the main changes required to extend PSO to the solution of MOPs were briefly described, including the use of external archives, the mechanisms to select leaders and the promotion of diversity in a swarm. In the final part of the paper, some of the possible paths for future research on MOPSOs were briefly addressed.

References

1. Angeline PJ (1998) Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In: Porto V, Saravanan N, Waagen D, Eiben A (eds) Evolutionary Programming VII. 7th International Conference, EP 98, Springer. Lecture Notes in Computer Science Vol. 1447, San Diego, California, USA, pp 601–610
2. Coello Coello CA, Lamont GB, Van Veldhuizen DA (2007) Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York, iSBN 978-0-387-33254-3
3. Deb K, Goldberg DE (1989) An investigation of niche and species formation in genetic function optimization. In: Schaffer JD (ed) Proceedings of the Third International Conference on Genetic Algorithms, George Mason University, Morgan Kaufmann Publishers, San Mateo, California, pp 42–50
4. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation 6(2):182–197
5. Engelbrecht AP (ed) (2002) Computational Intelligence: An Introduction. John Wiley & Sons, England
6. Engelbrecht AP (2005) Fundamentals of Computational Swarm Intelligence. John Wiley & Sons
7. Janson S, Middendorf M (2003) A hierarchical particle swarm optimizer. In: Congress on Evolutionary Computation (CEC'2003), IEEE Press, Camberra, Australia, pp 770–776
8. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, IEEE Service Center, Piscataway, New Jersey, pp 1942–1948
9. Kennedy J, Eberhart RC (2001) Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco, California
10. Knowles JD, Corne DW (2000) Approximating the nondominated front using the pareto archived evolution strategy. Evolutionary Computation 8(2):149–172
11. Moore J, Chapman R (1999) Application of particle swarm to multiobjective optimization, department of Computer Science and Software Engineering, Auburn University
12. Moore J, Chapman R, Dozier G (2000) Multiobjective Particle Swarm Optimization. In: Turner AJ (ed) Proceedings of the 38th Annual Southeast Regional Conference, 2000, ACM Press, Clemson, South Carolina, USA, pp 56–57
13. Reyes-Sierra M, Coello Coello CA (2006) Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. International Journal of Computational Intelligence Research 2(3):287–308
14. Toscano Pulido G (2005) 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
15. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3(4):257–271