An Introduction to Particle Swarm Multi-Objective Optimizers

Carlos A. Coello Coello

CINVESTAV-IPN
Evolutionary Computation Group (EVOCINV)
Computer Science Department
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07360, MEXICO

Luxembourg, May 2011

Outline of Topics

Basic Notions of Particle Swarm Optimization


Although originally adopted for balancing weights in neural networks [Eberhart, 1996], particle swarm optimization (PSO) soon became a very popular global optimizer, mainly in problems in which the decision variables are real numbers. Alternative encodings are also possible (e.g., binary [Kennedy, 1997] and integer [Eberhart, 2003]), but none of them has been as popular as the original proposal, which operates using vectors of real numbers.

Basic Notions of Particle Swarm Optimization

Some of the terminology commonly used with PSO is the following:

- **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 fitness).
- ***lbest*** (*local best*)**:** Position of the best particle member of the neighborhood of a given particle.

Basic Notions of Particle Swarm Optimization

- ***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.

Basic Notions of Particle Swarm Optimization

- **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.

Basic Notions of Particle Swarm Optimization

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.:

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

Basic Notions of Particle Swarm Optimization

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

$$\vec{v}_i(t) = W\vec{v}_i(t-1) + C_1 r_1(\vec{x}_{pbest_i} - \vec{x}_i(t)) + C_2 r_2(\vec{x}_{leader} - \vec{x}_i(t)) \quad (2)$$

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

Basic Notions of Particle Swarm Optimization

According to Angeline [1998], 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.

Basic Notions of Particle Swarm Optimization

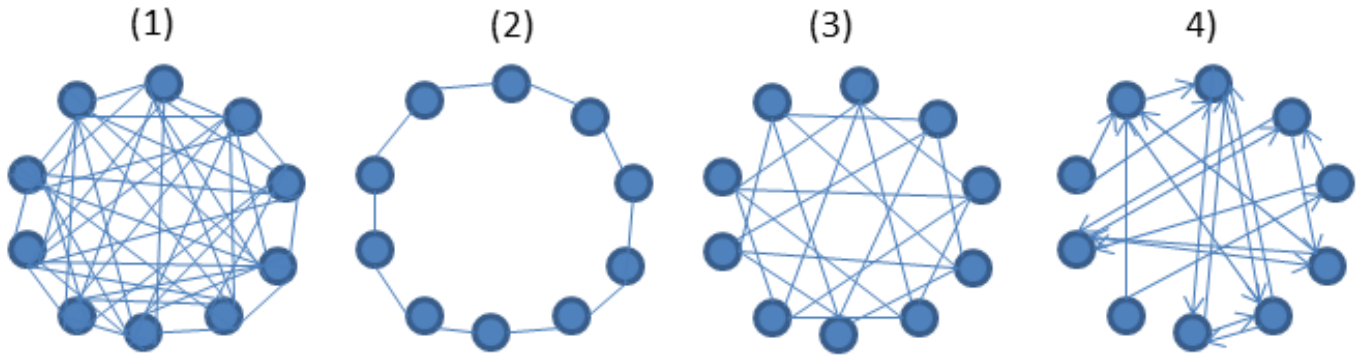
2. A second difference is that PSO uses an operator that sets the velocity of a particle to a particular direction. This can be seen as a directional mutation operator in which the direction is defined by both the particle's personal best and the global best (of the swarm). If the direction of the personal 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 a mutation operator that can set an individual in any direction (although the relative probabilities for each direction may be different). In fact, the limitations of this PSO operator have led to the use of mutation operators similar to those adopted EAs.

Basic Notions of Particle Swarm Optimization

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.

Basic Notions of Particle Swarm Optimization



Graphical representation of (1) fully connected, (2) ring, (3) von Neumann and (4) random topology

Particles tend to be influenced by the success of anyone they are connected to. These neighbors define the social structure of the swarm. Particles can be connected to each other in any kind of neighborhood topology represented as a graph.

Basic Notions of Particle Swarm Optimization

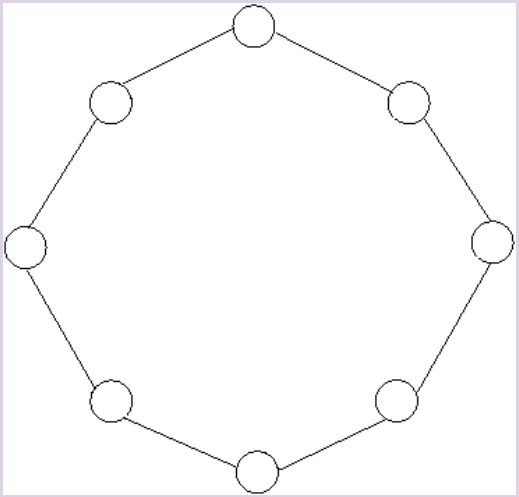
The following are types of neighborhood topologies that have been used with PSO:

- **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*). In this case, $C_2 = 0$ in equation (2).
- **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*). When $k = 2$, this structure is equivalent to a ring topology. In this case, *leader=lbest* in equation (2).

Basic Notions of Particle Swarm Optimization

Ring Topology

Below, we show the ring neighborhood topology that represents the *local* best scheme, when $k = 2$. In this *local* best case, each particle is affected only by its two immediate adjacent neighbors.



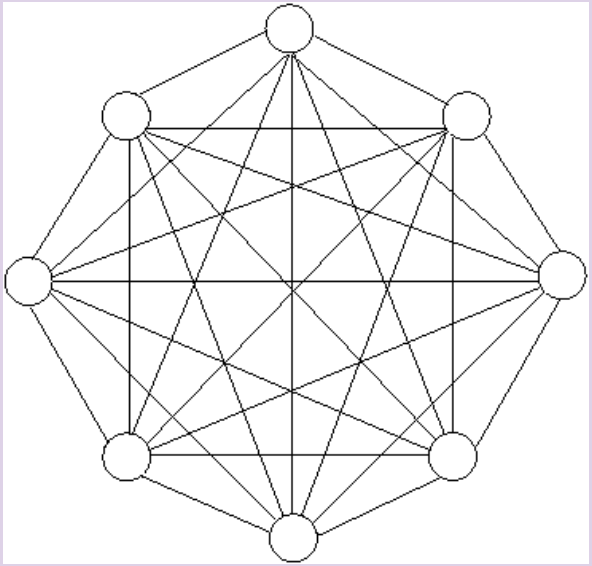
Basic Notions of Particle Swarm Optimization

- **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. In this case, *leader=gbest* in equation (2).

Basic Notions of Particle Swarm Optimization

Star Topology

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.



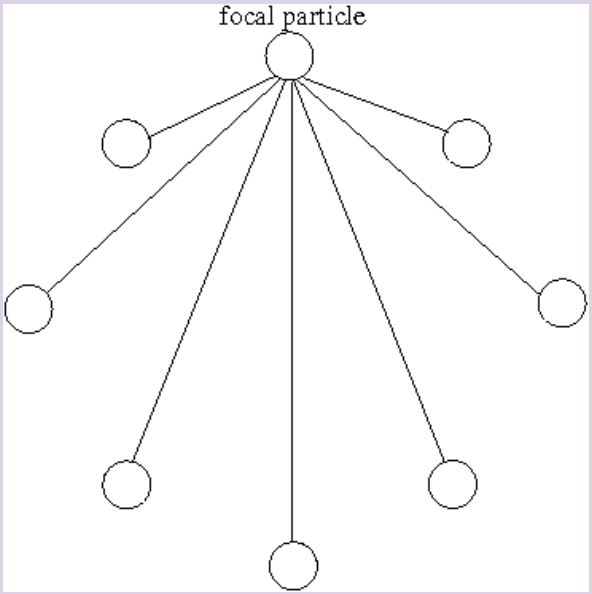
Basic Notions of Particle Swarm Optimization

- **Star network:** In this topology, one particle is connected to all others and they are connected to only that one (called *focal* particle). 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 (2).

Basic Notions of Particle Swarm Optimization

Wheel Topology

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.



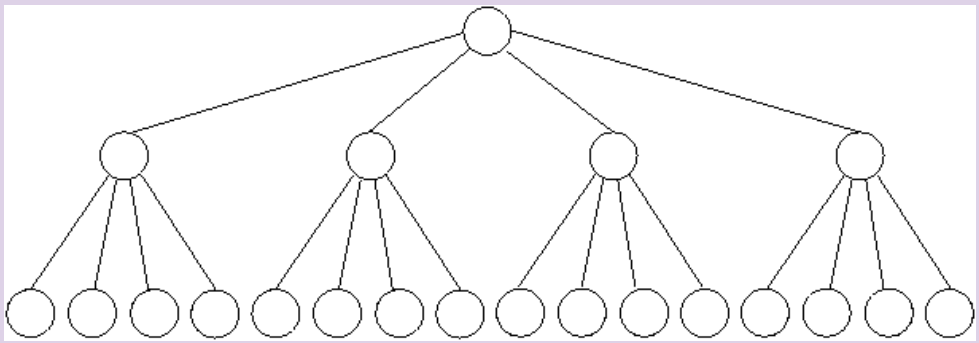
Basic Notions of Particle Swarm Optimization

- **Tree network:** In this topology, all particles are arranged in a tree and each node of the tree contains exactly one particle. 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. So, 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 (2).

Basic Notions of Particle Swarm Optimization

Tree Topology

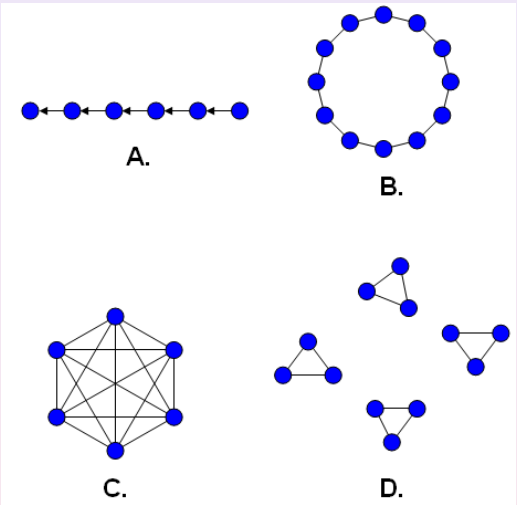
The tree network topology (each circle represents a particle). All particles are arranged in a tree. A particle is influenced by its own best position so far (*pbest*) and by the best position of the particle that is directly above in the tree. Here, we show an example of a topology defined by a regular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



Basic Notions of Particle Swarm Optimization

The neighborhood topology affects the performance of a PSO algorithm. For example, when using a *fully connected* topology, the swarm tends to converge more rapidly than when using *local best* topologies, because in the first case the information is spread more rapidly. However, a *fully connected* topology is also more prone to make that a PSO algorithm gets trapped in local optima.

Basic Notions of Particle Swarm Optimization



The influence of population topologies in PSO performance has been studied for the single-objective case (see for example Rui Mendes' PhD thesis), but not for the multi-objective case.

Basic Notions of Particle Swarm Optimization

The general (single-optimization) PSO algorithm works in the following way: First, the swarm is (randomly) 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 equations (1) and (2)) and its *pbest* and, finally, the leader is updated too.

PSO for Multi-Objective Optimization

In order to apply the PSO strategy for solving multi-objective optimization problems, it is obvious that the original scheme has to be modified. 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 the following:

1. How to select particles (to be used as leaders) in order to give preference to nondominated solutions over those that are dominated?

PSO for Multi-Objective Optimization

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?

PSO for Multi-Objective Optimization


Leader Selection

In a MOPSO, all the nondominated solutions currently available can serve as leaders. In the early days, a leader was randomly selected, but later on, this became a research topic (Toscano, 2005; Reyes Sierra, 2005; Branke, 2006).

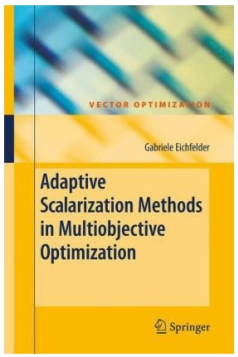
PSO for Multi-Objective Optimization

MOPSOs generally use an *external archive* (as most MOEAs) in which the nondominated solutions that are produced during the search are retained. However, external archives normally contain some mechanism to maintain diversity (e.g., based on the use of crowding or adaptive grids) as well.

PSO for Multi-Objective Optimization

Another interesting question is: How to promote diversity through the two main mechanisms to create new solutions: updating of positions (equations (1) and (2)) and a mutation (turbulence) operator (not part of the original PSO algorithm). Also, if we use mutation, do we apply it on the decision variables or on the velocity of the particles? Next, we will analyze each of these topics in more detail.

PSO for Multi-Objective Optimization



Selection of Leaders

Some researches have avoided the problem of defining the concept of leader for MOPs by adopting approaches that optimize each objective separately (e.g., scalarizing functions) or simple linear aggregating functions. See for example: MOPSO/D (Peng and Zhang, 2008).

PSO for Multi-Objective Optimization

However, the majority of the currently proposed MOPSO approaches redefine the concept of leader. The most straightforward approach is to consider every nondominated solution as a new leader and then, choose one of them using some *quality* measure. Obviously, such quality measure can be defined in several different ways.

PSO for Multi-Objective Optimization

Selection of Leaders

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.

PSO for Multi-Objective Optimization

Selection of Leaders

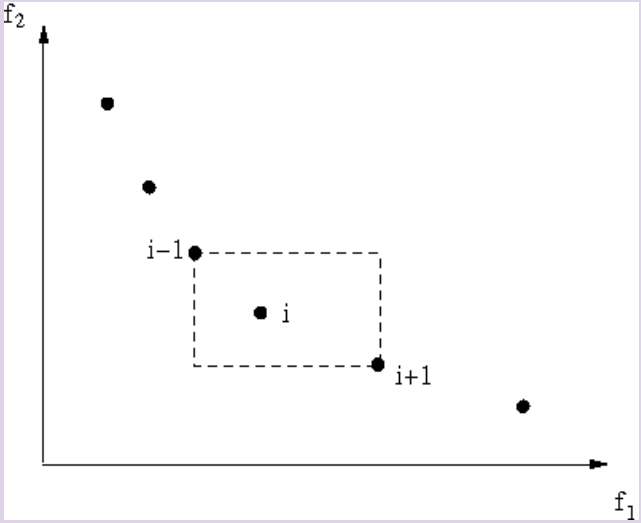
The two main density estimators currently used in multi-objective optimization are the following:

- **Nearest neighbor density estimator**: 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.

PSO for Multi-Objective Optimization

Nearest neighbor density estimator

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



PSO for Multi-Objective Optimization

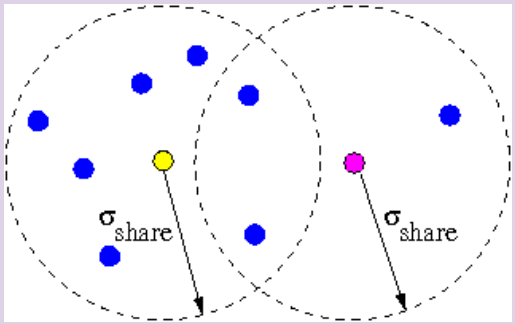
Selection of Leaders

- **Kernel density estimator**: 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*.

PSO for Multi-Objective Optimization

Kernel density estimator

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



PSO for Multi-Objective Optimization

Retaining and Spreading Nondominated Solutions

It is important to retain the nondominated solutions found along *all* the search process not only for pragmatic reasons, but also for theoretical ones [Rudolph, 1998]. 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).

PSO for Multi-Objective Optimization

Retaining and Spreading Nondominated Solutions

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, mainly due to practical reasons, archives tend to be bounded, which makes necessary the use of an additional criterion to decide which nondominated solutions to retain, once the archive is full.

PSO for Multi-Objective Optimization

Retaining and Spreading Nondominated Solutions

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: 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.

PSO for Multi-Objective Optimization

Retaining and Spreading Nondominated Solutions

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.

PSO for Multi-Objective Optimization

Promoting Diversity while Creating New Solutions

There are two main mechanisms used for creating new solutions that promote diversity:

1. **Updating of positions.** Topologies that define neighborhoods smaller than the entire swarm for each particle can preserve diversity within the swarm a longer time. Additionally, diversity can also be promoted by means of the inertia weight (W in equation (2)).

PSO for Multi-Objective Optimization

Promoting Diversity while Creating New Solutions

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. 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.

PSO for Multi-Objective Optimization

Promoting Diversity while Creating New Solutions

Shi [1999] asserted that by linearly decreasing the inertia weight from a relatively large value to a small one through the course of the PSO run, 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. [2003] 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.

PSO for Multi-Objective Optimization

Promoting Diversity while Creating New Solutions

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" [Shi, 1998].

PSO for Multi-Objective Optimization

Promoting Diversity while Creating New Solutions

2. **Through the use of a mutation (or turbulence) operator**. When a particle updates its position, a mutation with "conscience" occurs. Sometimes, however, some unconciousness or "craziness", as called by Kennedy and Eberhart, is needed. Craziness, also referred as turbulence, reflects the change in a particle's flight which is out of its control. 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.

PSO for Multi-Objective Optimization

Promoting Diversity while Creating New Solutions

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.

PSO for Multi-Objective Optimization

Promoting Diversity while Creating New Solutions

Several proposed approaches have used different mutation operators (whose selection is normally not easy). 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.

PSO for Multi-Objective Optimization

Algorithms

Moore and Chapman proposed the first extension of the PSO strategy for solving multi-objective problems in an unpublished manuscript from 1999 (an extended abstract of this work was published in 2000). After this early attempt, a great interest to extend PSO arose among researchers, but interestingly, the next proposal was not published until 2002.

PSO for Multi-Objective Optimization

Algorithms

Today, a wide variety of MOPSOs exist, based on aggregating functions, lexicographic ordering, sub-populations, Pareto ranking, and hybrids with other approaches.

To Know More

Margarita Reyes-Sierra and Carlos A. Coello Coello, **Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art**, *International Journal of Computational Intelligence Research*, Vol. 2, No. 3, pp. 287–308, 2006.

PSO for Multi-Objective Optimization

What is Missing?

- **Impact of the Parameters and Scalability:** Detailed studies of the impact of the parameters of the performance of a MOPSO are still scarce. This includes aspects such as the role of the leader selection scheme, and the impact of the neigborhood topology on performance. The performance of a MOPSO as we increase its number of objectives and/or decision variables is also important. Some studies in this direction are already available (see for example (Toscano, 2005; Durillo et al., 2009), but more work is still needed.

PSO for Multi-Objective Optimization

What is Missing?

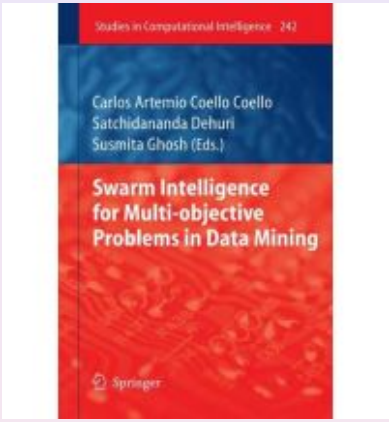
- **Theoretical studies:** Aspects such as convergence and run-time analysis of a MOPSO are currently missing in the literature. It is particularly important to identify possible advantages of MOPSOs over other MOEAs, since that would pave the way towards more applications.

PSO for Multi-Objective Optimization

What is Missing?

- **Algorithms:** More MOPSOs will certainly appear, but more creativity is needed here. Efficiency is now something that several researchers are aiming when proposing new MOPSOs (see for example (Toscano and Coello, 2007)). However, something that we will probably see more often in the next few years will be leader selection schemes based on performance measures. More hybrids are also expected to appear in the next few years.

PSO for Multi-Objective Optimization



What is Missing?

- **Applications:** More exciting applications of MOPSOs are expected to appear in the following years, given the relative success of this type of algorithms in certain domains (e.g., engineering).

To Conclude

Conclusions

To conclude, I believe that research on MOPSOs will continue for several more years. This area offers several exciting challenges (e.g., regarding theoretical foundations of MOPSOs), has the potential for the development of more applications (e.g., in data mining, classification and bioinformatics), and still maintains certain topics with very little (or no) research done (e.g., many-objective optimization, and hardware implementations of MOPSOs).