International Journal of Research Studies in Science, Engineering and Technology
Volume 2, Issue 9, September 2015, PP 30-49
ISSN 2349-4751 (Print) & ISSN 2349-476X (Online)

Heterogeneous Multi-Objective Particle Swarm Optimiser with a Spread Particle

Vytautas Jančauskas

Institute of Mathematics and Informatics
Vilnius University, Vilnius, Lithuania

Abstract: *In this paper we show how to use heterogeneous (consisting of different types of particle) particle swarms to improve the performance of Particle Swarm Optimization when solving multi-objective problems. We propose to use three different particle types -- two designed to get close to the real Pareto-optimal set and one to ensure uniform spread of solutions along it. By using them together, sharing information via the same non-dominated point archive, they are able to cancel out each other's weaknesses and ensure better performance measured via various metrics than either would be capable of alone. We describe the particles, algorithms and metrics and give results both graphically and numerically that support this thesis.*

Keywords: *Multi-objective optimization, swarm intelligence, particle swarm optimization.*

1. INTRODUCTION

Particle Swarm Optimization (PSO) is a global optimization meta-heuristic. It uses insights in to the behavior of social animals (bird flocks and fish schools in particular) to try and solve the problem of finding a functions global minimum. This method, with some modifications can also be used to solve problems with more than one objective. That is problems where more than one function has to be minimized with the same argument. Such problems arise frequently in all areas of human activity from economics to science and engineering. Solutions to such problems are sets called Pareto frontiers (PF) - sets of points in objective space where no objective can be improved without increasing the value of other objectives. It is up to the expert then to pick one solution depending on what priorities they have for different objectives. Since there are no general precise methods to solve these problems one often has to rely on various heuristics to obtain approximations of Pareto frontiers. These sets can then be evaluated, provided that the actual Pareto frontier is known. There are two primary properties that we look for in the approximation - how close it is to the actual Pareto frontier and how well do the points in the approximation represent the actual Pareto frontier. There has been quite a lot of research in to using PSO for these problems as of late as discussed in a later section. In developing such methods one often has to compromise and decide which attribute of the optimizers performance to prioritize. For example some methods will try to get as close to the Pareto optimal set as possible but are prone to getting stuck in local minima. Others will attempt to ensure uniform spread of solutions along the real Pareto optimal set yet may sacrifice the ability to decrease the distance between approximation and the real Pareto-optimal set. Uniform distribution is important since it lets us see all the different aspects of a continuous Pareto set. A successful algorithm will try to combine those two properties. We propose a method where we use particles designed separately for each of these aspects of performance together in a swarm sharing information via a non-dominated point archive. Such a swarm (called heterogeneous since it contains particles of different types) works by switching a particles' type to another one if that particle fails to improve over a set number of iterations. This way the swarm can adapt to the problem and use particle types that are most appropriate. We also propose a particle designed to ensure uniform spread of non-dominated points. We perform an analysis of this new PSO variant using a variety of metrics designed to evaluate multi-objective optimization algorithm performance.

In Section 2 we describe the problem of multi-objective optimization and define key terms and abbreviations used later in the text. In Section 3 we described the basic Particle Swarm Optimization (PSO) algorithm, how it can be used for multi-objective problems and the particles we used in our algorithm. One of the particles is original, proposed by us and described in this paper. In section 5 we describe the test problems used to test optimizer performance and different metrics we used to measure it. In Section 6 we provide tables with average values for different metrics, plots of the Pareto frontier (PF) approximations found by our algorithm and discuss these results. In Section 7 we discuss what we achieved and provide some interesting areas for future research.

2. MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION

Multi-objective optimization is concerned with mathematical optimization problems involving more than one objective function. These functions are supposed to be optimized simultaneously, with one solution minimizing all of them. In practice, however, this is rarely possible since objectives tend to contradict each other. For example: in manufacturing using light-weight materials may reduce strength while increasing strength and keeping weight constant may increase the price. Such problems where we try to find a single solution that will achieve several goals at once are very common in practice. Since a single best solution is often impossible we try to find several solutions that emphasize one objective or another without sacrificing any of the remaining ones. The user will then select one of the solutions depending on which objectives they want to emphasize more. Since we cannot say which solution of any two is better, a different concept is required than simple ordering of solutions in single objective optimization. Such a concept is called Pareto dominance. Below we describe the problem of multi-objective optimization and the concept of Pareto dominance more formally.

$$\min(f_1(x), f_2(x), \ldots, f_k(x)) \tag{1}$$

We can look at the result of a multi-objective function as a vector consisting of values of the corresponding objectives: $y = (f_1(x), f_2(x), ..., f_k(x))$. The problem of multi-objective optimization is to find solution x that minimizes each function f_i, where $i \in \{1, ..., k\}$ as shown in (1) formula. Since a single such solution is most of the time impossible we attempt to find a set of solutions that satisfy a certain requirement. That requirement is that a solution must not be Pareto dominated with respect to any other solution in the set.

$$(\exists i \in \{1,\ldots,k\} : y_{1i} < y_{2j}) \wedge (\forall j \in \{1,\ldots,k\} : y_{1j} \leq y_{2j}) \tag{2}$$

Pareto dominance formalized in Equation (2) states that solution y_1 is said to Pareto dominate another solution y_2 if there is at least one dimension of y_1 that is strictly lower than a corresponding dimension of y_2 and that all dimension of y_1 are lower than or equal to corresponding dimensions of y_2. Said simply it means that solution y_1 improves at least one objective over y_2 without making any of the remaining objectives worse (in the case of minimization worse means higher). A set of points that are not Pareto dominated by any other point in the set is called a Pareto frontier (PF) and a set of corresponding solutions is called a Pareto set (PS). A Pareto frontier is the south-western corner of the diagram representing objective function values over feasible solutions. An overview of using search methodologies to solve multi-objective problems can be found in a work by Kalyanmoy Deb [1].

Particle Swarm Optimization is a global optimization meta-heuristic originally designed for continuous problems. It was first proposed by James Kennedy and Russell C. Eberhart in 1995 [2]. The idea is to have a swarm of particles (points in multi-dimensional space) each having some other

particles as neighbors and exchanging information to find optimal solutions. Particles move in the solution space of some function by adjusting their velocities to move towards the best solutions they found so far and towards the best solutions found by their neighbors. These two attractors are further randomly weighted to allow more diversity in the search process. The idea behind this algorithm are observations from societies acting in nature. For example one can imagine a flock of birds looking for food by flying towards other birds who are signaling a potential food source as well as by remembering where this particular bird itself has seen food before and scouting areas nearby. Parts of it can also be viewed as modelling the way we ourselves solve problems - by imitating people we know, who we see as particularly successful, but also by learning on our own. Thus problem solving being influenced by our own experience and by the experience of people we know to have solved similar problems particularly well. The original algorithm is not presented here since it is very rarely used today and we go straight to more modern implementations.

Currently the ``Canonical" PSO algorithm is considered to be that by Maurice Clerc et al. [3] which is a variant of the original PSO algorithm. We, however, use a slightly different version that seems to give good results when used in the context of multi-objective optimization. Particles get a position vector corresponding to a solution and a velocity vector. Particles position are initialized to uniformly distributed values from the problems domain at the start.

$$x_i \leftarrow x_i + v_i \tag{3}$$

During each iteration of the PSO algorithm particle's with index i position is updated using Equation (3) where velocity is simply added to that particle's position.

$$v_i \leftarrow wv_i + c_1 U_{(0,1)}(x_i - p_i) + c_2 U_{(0,1)}(x_i - g_i) \tag{4}$$

After updating position, velocity is updated using Equation (4) where p_i is the best solution found by the particle so far and g_i is the best solution found by the swarm so far. Coefficient w is called inertia weight and it influences how much is the particle's current velocity influenced by its' previous velocity. Coefficients c_1 and c_2 are influence factors for personal and social experience respectively. They are used to regulate how much is a particle influenced by its' personal experience and by the experience of its neighbors. These vectors take on different meanings when used for multi-objective problems as described in later sections.

An important part of PSO recently are the mutation operators that make the algorithm a little less prone to getting stuck in local minima. These operators seem to be particularly important in multi-objective optimization. An overview of mutation operators is given by Paul S. Andrews [4]. PSO with Gaussian mutation operator is described by Natsuki Higashi et al. [5]. Application of mutation to PSO for multi-objective problems is examined by Margarita Reyes Sierra et al. [6].

It is relatively simple to use PSO for multi-objective problems although effectiveness in getting close to the real PF or good spread along the PF are not guaranteed. There are at least two changes to the standard PSO. First of all it is not clear how to update p_i after updating particle position. This is because simply checking if the new solution gives a smaller fitness value does not work for more than one objective. Usually this is resolved by updating p_i if the new solution dominates the previous one in objective space. The second problem is selecting g_i. This is usually done by selecting a point from the non-dominated point archive. This can be done in a variety of ways. Three such methods are discussed in the two following subsections. The algorithm for maintaining a non-dominated point

archive is given in Algorithm 2. This algorithm can be found in Yaochu Jin et al. [7] among other sources.

Algorithm 1 General multi-objective PSO algorithm when using a global non-dominated point archive.

```
1:  for i ← 1 to n do
2:      x_i ← U_(a,b)
3:      p_i ← x_i
4:      v_i ← 0
5:      append(archive, f(x_i))
6:  end for
7:  for j ← 1 to number of iterations do
8:      for i ← 1 to n do
9:          g_i ← pickGlobalBest(archive, i)
10:         v_i ← wv_i + c1U_(0,1)(p_i − x_i) + c2U_(0,1)(g_i − x_i)
11:         x_i ← x_i + v_i
12:         if better(x_i, p_i) then
13:             x_i ← p_i
14:         end if
15:     end for
16:     for i ← 1 to n do
17:         append(archive, f(x_i))
18:     end for
19: end for
```

Algorithm 1 gives an outline for a generalized multi-objective PSO algorithm that uses a Pareto point archive. At first particle positions are initialized to uniformly distributed values from a given range. Vector $U_{(a, b)}$ is a vector of uniformly distributed random numbers all from ranges given by vectors a and b where a is the vector with lower bounds for each coordinate and b is the vector with upper bounds for each coordinate. At first velocities are initialized to zero and p_i is set to the current position of particle i. Then for a given number of iterations particle positions and velocities are updated with the update rules discussed in a section about PSO. Depending on particle type g_i and p_i are selected and updated differently. After each position update the algorithm will attempt to store it in the non-dominated point archive. The procedure for storing solutions is given in Algorithm 2.

Algorithm 2 Adding candidate solution o to a bounded non-dominated point archive.

```
1:  if o is not dominated by any solutions in the archive then
2:      if archive is not full then
3:          add o to archive
4:      else if o dominates any solution a in the archive then
5:          replace a with o
6:      else if solution a_1 dominates solution a_2 in the archive then
7:          replace a_2 with o
8:      else
9:          discard o
10:     end if
11: end if
12: for a_1 in the archive do
13:     if there is solution a_2 in the archive that dominates a_1 then
14:         remove a_1 from the archive
15:     end if
16: end for
```

A good overview of how PSO is used for multi-objective problems can be found in a work by Margarita Reyes-Sierra et al. [8] or for an older review of basic approaches see work by Konstantinos E. Parsopoulos et al. [9]. Common approaches to using include dynamic aggregation as for example described by Carlos A. Coello Coello et al. [10]. Approaches related to ours are described by Xiaohui Hu et al. [11], Carlos A. Coello Coello et al. [12] and Jonathan Fieldsend et al. [13].

2.1. "Sigma" Particle

This strategy was proposed by Sanaz Mostaghim et al. [14]. It works by assigning certain vectors of values to each solution in the solutions in the non-dominated point archive. These values are calculated using the formula in Equation (5) for two-objective case and Equation (6) for three-objective case. It can be generalized to more objectives.

$$\sigma = \frac{f_1^2 - f_2^2}{f_1^2 + f_2^2} \tag{5}$$

$$\sigma = \begin{pmatrix} f_1^2 - f_2^2 \\ f_2^2 - f_3^2 \\ f_3^2 - f_1^2 \end{pmatrix} / \left(f_1^2 + f_2^2 + f_3^2\right) \tag{6}$$

When using this method for selecting g_i we first calculate σ_i for that particle and then we look for a point in the archive that has value of σ that is closest in terms of Euclidean distance. This point is then used as the global best. Authors state that this method allows the particles to fly directly towards the Pareto-optimal front. The problem of updating p_i is solved by updating it only if the new solution dominates the previous p_i. The authors also use a mutation operator of the form $x_{i,j} = x_{i,j} + U_{(0,1)}x_{i,j}$ where $U_{(0,\ 1)}$ is a uniformly distributed random number from the range $(0,\ 1)$ and $x_{i,j}$ is the j-th coordinate of the position of i-th particle. This mutation operator is applied with probability 0.05.

2.2. ``Closest'' Particle

This particle works the same way as the ``sigma" particle, with one exception – $g_i$ is set to the point in the non-dominated point archive that is closest in terms of Euclidean distance in objective space to our particle. All other parameters and logic is exactly the same as in the ``sigma" particle just the procedure by which g_i is chosen is different.

2.3. ``Spread'' Particle

This is a particle we propose to handle ``horizontal" exploration of the Pareto-optimal approximation. That is it is designed to make sure that the approximated PF contains uniformly distributed points. This, of course, presupposes that the PF is continuous. To achieve this two important changes are introduced. First of all when selecting the g_i a solution is chosen from the archive is the one that has the lowest score on the crowding distance metric that is discussed in Subsection 5.5. Furthermore when updating p_i the new solution replaces the old one when the new one is not dominated by the old one. We found that this is essential in promoting the spread of points in the approximated PF. All the other parameters are left exactly the same as in the case of sigma particle. A similar method is described by Carlo R. Raquel et al. [15].

2.4. Heterogeneous Swarm

We propose to use a swarm where particle types are decided dynamically. Initially particle types are chosen from the three discussed above with equal probabilities. During the evolution of the swarm if the particle's p_i does not change for a set number of iterations that particle's type is changed and its x_i and all other properties are reset to values they had at the beginning of the run. All particle types use the same non-dominated point archive which is the primary medium of information exchange for this swarm. Similar particle swarms for single objective optimization were described by Andries P. Engelbrecht [16] and Marco Antonio Montes de Oca et al. [17].

3. EXPERIMENTAL PROCEDURE

Here we describe test problems used in this paper to perform experiments. Their definitions and properties are briefly described. For each of these problems Pareto Set and Pareto Frontier are both known which is convenient for our study. In all problems given below dimensionality *d* is set to 30. In all these cases the problem is that of minimizing the objectives. They were compiled by Quingfu Zhang et al. [18] for use in the CEC 2009 multi-objective optimization algorithm competition. More information and original definitions of these problems can be found in work by Hui Li et al. [19]. We only use two objective problems since the results are easier to interpret that way and researchers often limit the scope to two objectives at first. There are seven of them. The use of these problems have two main advantages:

- They are varied - the 7 problem set contains problems with convex, non-convex and discrete PF's. If an algorithm is to solve all of them well it has to be extremely versatile.
- Important information is known about the problems such as formulas for calculating PF and PS which make it easier to evaluate optimizer performance since most metrics require this information.

There is the added advantage that it is possible to test your algorithm against the algorithms that entered the contest which include several popular and powerful approaches that PSO has to aim to match if it is to be considered a viable alternative to genetic algorithms for multi-objective problems.

3.1. CEC 2009 Unconstrained Problem 1

The two objectives are:

$$f_1(x) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}\left(x_j - sin\left(6\pi x_1 + \frac{j\pi}{d}\right)\right)^2 \tag{7}$$

$$f_2(x) = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}\left(x_j - sin\left(6\pi x_1 + \frac{j\pi}{d}\right)\right)^2 \tag{8}$$

With J_1 and J_2 defined below.

$$J_1 = \{j \mid j \text{ is odd and } 2 \le j \le d\}$$

$$J_2 = \{j \mid j \text{ is even and } 2 \le j \le d\}$$

The search space for this problem is $[0, 1] \times [-1, 1]^{d-1}$. Pareto Frontier and Pareto Set (PF and PS) are given in the two equations below.

$$f_2 = 1 - \sqrt{f_1}, \quad 0 \le f_1 \le 1$$

$$x_j = sin\left(6\pi x_1 + \frac{j\pi}{d}\right), \quad j = 2, \ldots, d, \quad 0 \le x_1 \le 1$$

3.2. CEC 2009 Unconstrained Problem 2

The two objectives are:

$$f_1(x) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1} y_j^2 \tag{9}$$

$$f_2(x) = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2} y_j^2 \tag{10}$$

$$y_j = \begin{cases} x_j - \left(0.3x_1^2 cos\left(24\pi x_1 + \frac{4j\pi}{d}\right) + 0.6x_1\right) cos\left(6\pi x_1 + \frac{j\pi}{d}\right) & j \in J_1 \\ x_j - \left(0.3x_1^2 cos\left(24\pi x_1 + \frac{4j\pi}{d}\right) + 0.6x_1\right) sin\left(6\pi x_1 + \frac{j\pi}{d}\right) & j \in J_2 \end{cases} \tag{11}$$

With J_1 and J_2 defined below.

$$J_1 = \{j \mid j \text{ is odd and } 2 \le j \le d\}$$

$$J_2 = \{j \mid j \text{ is even and } 2 \le j \le d\}$$

The search space for this problem is $[0,1] \times [-1, 1]^{d-1}$. Pareto Frontier and Pareto Set (PF and PS) are given in the two equations below.

$$f_2 = 1 - \sqrt{f_1}, \quad 0 \le f_1 \le 1$$

$$x_j = \begin{cases} \left(0.3x_1^2 cos\left(24\pi x_1 + \frac{4j\pi}{d}\right) + 0.6x_1\right) cos\left(6\pi x_1 + \frac{j\pi}{d}\right) & j \in J_1 \\ \left(0.3x_1^2 cos\left(24\pi x_1 + \frac{4j\pi}{d}\right) + 0.6x_1\right) sin\left(6\pi x_1 + \frac{j\pi}{d}\right) & j \in J_2 \end{cases}$$

3.3. CEC 2009 Unconstrained Problem 3

The two objectives are:

$$f_1(x) = x_1 + \frac{2}{|J_1|}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1} cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right) \tag{12}$$

$$f_2(x) = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2} cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right) \tag{13}$$

$$y_j = x_j - x_1^{0.5\left(1.0 + \frac{3(j-2)}{d-2}\right)}, \quad j = 2, \ldots, d \tag{14}$$

J_1 and J_2 are defined in the same way as in previous problems.

The search space for this problem is $[0, 1]^d$. Pareto Frontier and Pareto Set (PF and PS) are given in the two equations below.

$$f_2 = 1 - \sqrt{f_1}, \quad 0 \le f_1 \le 1$$

$$x_j = x_1^{0.5\left(1.0+\frac{3(j-2)}{d-2}\right)}, \quad j = 2, \ldots, d, \quad 0 \le x_1 \le 1$$

3.4. CEC 2009 Unconstrained Problem 4

The two objectives f_1 and f_2 are given in the equations below.

$$f_1(x) = x_1 + \frac{2}{|J_1|}\left(\sum_{j \in J_1} h(y_j)\right) \tag{15}$$

$$f_2(x) = 1 - x_1^2 + \frac{2}{|J_2|}\left(\sum_{j \in J_2} h(y_j)\right) \tag{16}$$

$$y_i = x_j - sin\left(6\pi x_1 + \frac{j\pi}{d}\right), \quad j = 2, \ldots, d \tag{17}$$

$$h(t) = \frac{|t|}{1 + e^{2|t|}} \tag{18}$$

J_1 and J_2 are defined in the same way as in previous problems.

The search space for this problem is $[0,1] \times [-2,2]^{d-1}$. Pareto Frontier and Pareto Set (PF and PS) are given in the two equations below.

$$f_2 = 1 - f_1^2, \quad 0 \le f_1 \le 1$$

$$x_j = sin\left(6\pi x_1 + \frac{j\pi}{n}\right), \quad j = 2, \ldots, d, \quad 0 \le x_1 \le 1$$

3.5. CEC 2009 Unconstrained Problem 5

The two objectives f_1 and f_2 are given in the equations below.

$$f_1(x) = x_1 + \left(\frac{1}{2N} + \varepsilon\right)|\sin(2N\pi x_1)| + \frac{2}{|J_1|}\sum_{j \in J_1} h(y_j) \tag{19}$$

$$f_2(x) = 1 - x_1 + \left(\frac{1}{2N} + \varepsilon\right)|\sin(2N\pi x_1)| + \frac{2}{|J_2|}\sum_{j \in J_2} h(y_j) \tag{20}$$

$$y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{d}\right), \quad 2,\ldots,d \tag{21}$$

$$h(t) = 2t^2 - \cos(4\pi t) + 1 \tag{22}$$

In the equations above N is an integer and $\varepsilon > 0$, J_1 and J_2 are defined the same way as in the problems above. We set these values to $N=10$ and $\varepsilon = 0.1$, same as in CEC 09 algorithm contest.

The search space for this problem is $[0,1] \times [-1, 1]^{d-1}$. The PF for this problem consists of $2N + 1$ Pareto Optimal solutions that have the form:

$$\left(\frac{i}{2N}, 1 - \frac{i}{2N}\right), \quad i = 0,1,\ldots,2N$$

3.6. CEC 2009 Unconstrained Problem 6

The two objectives f_1 and f_2 are given in the equations below.

$$f_1(x) = x_1 + \max\left\{0, 2\left(\frac{1}{2N} + \varepsilon\right)\sin(2N\pi x_1)\right\} +$$

$$+\frac{2}{|J_1|}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right) \tag{23}$$

$$f_2(x) = 1 - x_1 + \max\left\{0, 2\left(\frac{1}{2N} + \varepsilon\right)\sin(2N\pi x_1)\right\} +$$

$$+\frac{2}{|J_2|}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right) \tag{24}$$

$$y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{d}\right), \quad j = 2,\ldots,d \tag{25}$$

In the equations above N is an integer and $\varepsilon > 0$, J_1 and J_2 are defined the same way as in the problems above. We set these values to $N = 2$ and $\varepsilon = 0.1$, same as in CEC 09 algorithm contest.

The search space for this problem is $[0,1] \times [-1, 1]^{d-1}$. The PF for this problem consists of one isolated point $(0, 1)$ and N disconnected parts of the form:

$$f_2 = 1 - f_1, \quad f_1 \in \bigcup_{i=1}^{N}\left[\frac{2i-1}{2N}, \frac{2i}{2N}\right]$$

3.7. CEC 2009 Unconstrained Problem 7

The two objectives f_1 and f_2 are given in the equations below.

$$f_1 = \sqrt[5]{x_1} + \frac{2}{|J_1|}\sum_{j\in J_1} y_j^2 \tag{26}$$

$$f_1 = 1 - \sqrt[5]{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2} y_j^2 \tag{27}$$

$$y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{d}\right), \quad j = 2,\ldots,d \tag{28}$$

In these equations J_1 and J_2 are defined the same way as in the problems above.

The search space for this problem is $[0,1] \times [-1, 1]^{d-1}$. The PF and PS are given in the equations below.

$$f_2 = 1 - f_1, \quad 0 \le f_1 \le 1$$

$$x_j = sin\left(6\pi x_1 + \frac{j\pi}{d}\right), \quad j = 2,\ldots,d, \quad 0 \le x_1 \le 1$$

3.8. Experimental Procedure

To test the performance of the proposed algorithms we use test problems from 2009 IEEE Congress on Evolutionary Computation (CEC 2009) special session and competition for multi-objective optimization algorithms. We used only the first 7 unconstrained problems with two objectives. We did not try our algorithm on problems with 3 or 5 objectives and constrained problems since that would add an extra layer of complexity. However there are no reasons to believe that our results don't translate to more objectives. The problems in question can be found in a technical report by Qingfu Zhang et al. [18]. They are referred to as UF01 to UF07 in the same order as they appear in the aforementioned technical report. We use PF reference sets provided by the competition organizers when the metric used required them. You can see the PF reference sets plotted in the results section of this paper.

We ran the experiments for swarms containing ``sigma" particles, ``spread" particles and ``closest" particles only as well as a heterogeneous swarm containing all three particle types. Each swarm consisted of 250 particles and ran for 1200 iterations. The large number of particles was chosen because the problems are 30 dimensional. There is a rule of thumb in genetic algorithms to use a population 10 times the size of the dimensionality of the problem. The number of function evaluations is consistent with the CEC 2009 competition conditions of not exceeding 300000 function evaluations during the run of the algorithm. For each swarm 30 runs were done and the resulting PF approximations were recorded. Each approximation was then measured using the metrics described below and the average value of those metrics was recorded in to tables that have the metrics for columns and problems for rows with the mean value of the metric for that problem at the intersection. One such table was created for each swarm type. Particle parameters are as follows: $c_1 = c_2 = 1.0$, $w = 0.4$. This is contrary to what constitutes ``good" values of those parameters for single objective PSO, however empirical tests suggest these work better. Also mutation Gaussian mutation was used with probability 0.05. Value $N(0, 0.1r)$, where r is the range between lower and upper bounds for that

coordinate was added to each coordinate with aforementioned probability. When using heterogeneous swarm particle type was changed to another one with position and velocity being re-initialized if particle stagnated for 25 iterations. Stagnation is defined as particle's p_i not being updated.

For two objective problems the size of the Pareto point archive was limited to 100. The tables are given as part of the Results section along with the plots of the approximated PF for each problem and swarm that had the lowest value of IGD metric. IGD was chosen because it is purported to measure both closeness to the true PF and uniformity of the approximation so it is sometimes used as the only figure of merit for measuring optimizer performance. Metrics used to analyze optimizer performance are described in the subsections below. They were chosen so as to cover both closeness to the real Pareto frontier and the uniformity of spread of solutions along it.

Generational Distance (GD)

Generational distance is defined as given in Equation (29). Here, and in the explanations of other metrics, A is the approximate set to the PF, P^* is a set of uniformly distributed points along the PF (in the objective space) and $d(v, P^*)$ is the minimum distance between point v and one of the points in P^*.

$$GD(A, P^*) = \frac{\sum_{v \in A} d(v, P^*)}{|A|} \tag{29}$$

This metric measures how close the approximate set is to the real PF. It is important to note that as long as the approximate set A lies close to the real PF this metric will report a value close to zero even if large portions of the PF are missing in the approximation. This is a well-known measure and can be found for example in David A. Van Veldhuizen et al. [20].

Inverted Generational Distance (IGD)

This metric is similar to the previous one. It measures how close actual PF is to the approximated one. For this we simply swap the sets in Equation (29) places as given in Equation (30).

$$IGD(A, P^*) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|} \tag{30}$$

This metric will give lower values for approximations that are close to the actual PF and furthermore no large portions of the PF can be missing in approximation. Because of this, this metric is sometimes used as the only figure of merit when comparing multi-objective optimization algorithms on problems where PF is known. This metric can be said to measure both closeness to the real PF and uniformity of approximation. Description of the use of this metric can be found, for example, in Hui Li et al. [19]. There is however a downside when using this metric as the only measure of merit for multi-objective optimization algorithms. It is not clear to what extent do the two properties influence the value of this metric. It is also not clear what is their exact relationship when calculating IGD. Therefore it may be a better idea to instead use two separate metrics when possible. One to measure the uniformity of spread of solutions and one to measure how close the approximation is to the actual PF in terms of average Euclidean distance. Measuring distance is trivial with the help of GD while measuring spread is not as straightforward and most commonly used metrics have serious downsides.

Spacing Metric (SP)

Originally proposed by Jason R. Schott [21] and sometimes referred to in literature as Schott's metric. This metric measures the deviation in the distance between nearest neighbors in terms of Manhattan distance. The formula to calculate this metric is given in Equation (31).

$$SP(A) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2} \tag{31}$$

Here $\bar{d}$ is the mean value of all d_i, $i \in \{1, ..., n\}$ and d_i is defined in Equation (32).

$$d_i = \min_{j \in \{1,...,n\}} \left(\left| f_1^i(x) - f_1^j(x) \right| + \ldots + \left| f_m^i(x) - f_m^j(x) \right| \right) \tag{32}$$

The problem with this metric is that it is not sensitive to large gaps in the approximated set. It only looks at the distance to the closest neighbor. This means that if an approximation consists of tightly packed clusters of points with large gaps between them the value of this metric will be low. However a solution like this can only be regarded as poor provided that the actual PF is continuous. For this reason this metric should be used carefully if at all.

Maximum Gap Metric (MG)

This metric is similar to Generational Distance and Inverted Generational Distance and is used to measure maximum distance between the real PF and the approximation *A*. The formula to compute this metric is given in Equation (33).

$$MG(A) = \max_{v \in P^*} d(v, A) \tag{33}$$

Here *d(v, A)* is defined the same way as in Equation (30). This metric in essence measures the extent of the largest missing part of the real PF in approximate set *A*.

Crowding Distance Metric (CD)

This metric is best explained with the help of Figure 1. This way of measuring how crowded is the area around a particle is described in, for example, Kalyanmoy Deb et al. [22] where it is called crowding-distance. We define a metric using it. It is defined as standard deviation of *ci, i ∈{1, ..., n}*. Value c_i is half the perimeter of the box shown in Figure 1. We only consider the two objective case. First the elements of *A* are sorted lexicographically starting with the first objective. Then for each point *i* we measure the perimeter of the rectangle formed by points *i – 1* and *i + 1* as opposing corners. The first and last points don't have an element on the left and right respectively and thus we use the perimeter of the rectangle formed by the first/last point and the point to the right/left. This way of measuring how crowded is the area around a particle is described in, for example, Kalyanmoy Deb et al. [22] where it is called crowding-distance.

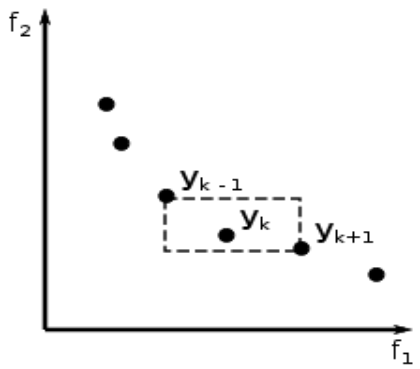


Fig1. *Illustration of crowding distance*

$$CD(A) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{c} - c_i)} \tag{34}$$

4. Results and Analysis

We can see the results for different swarm types running on problem UF01 in Figures 2a, 2b, 2c and 2d. Also in Tables 1, 2, 3 and 4. The inclination of the ``spread" particle to get a uniformly covered non-dominated set can plainly be seen in Figure 2b. On the other hand ``sigma" particle tries to get as close as possible to the real PF which results in it uncovering small portions of it that are separated by large gaps. Same can be said of ``closest" particle type. If we look at IGD metric we see that heterogeneous swarm does best. In this one case our method does not seem to significantly improve performance compared to the ``closest" particle alone although CM and SP metrics do show a small improvement in the uniformity of the approximated PF. Another interesting thing is that ``closet" particle performs better than the more elaborate ``sigma" particle

Table1. *Results for a swarm containing only sigma particles.*

Problem	IGD	GD	SP	MG	CM
UF01	0.09608	0.00390	0.00384	0.41662	0.10021
UF02	0.05913	0.00293	0.01857	0.30574	0.03731
UF03	0.22118	0.08248	0.02415	0.62886	0.05426
UF04	0.05633	0.05566	0.00879	0.10533	0.04965
UF05	0.60519	0.63110	0.08325	0.84161	0.15362
UF06	0.31305	0.19439	0.02680	0.53425	0.10645
UF07	0.11991	0.01057	0.02311	0.35233	0.09894

Table2. *Results for a swarm containing only closest particles.*

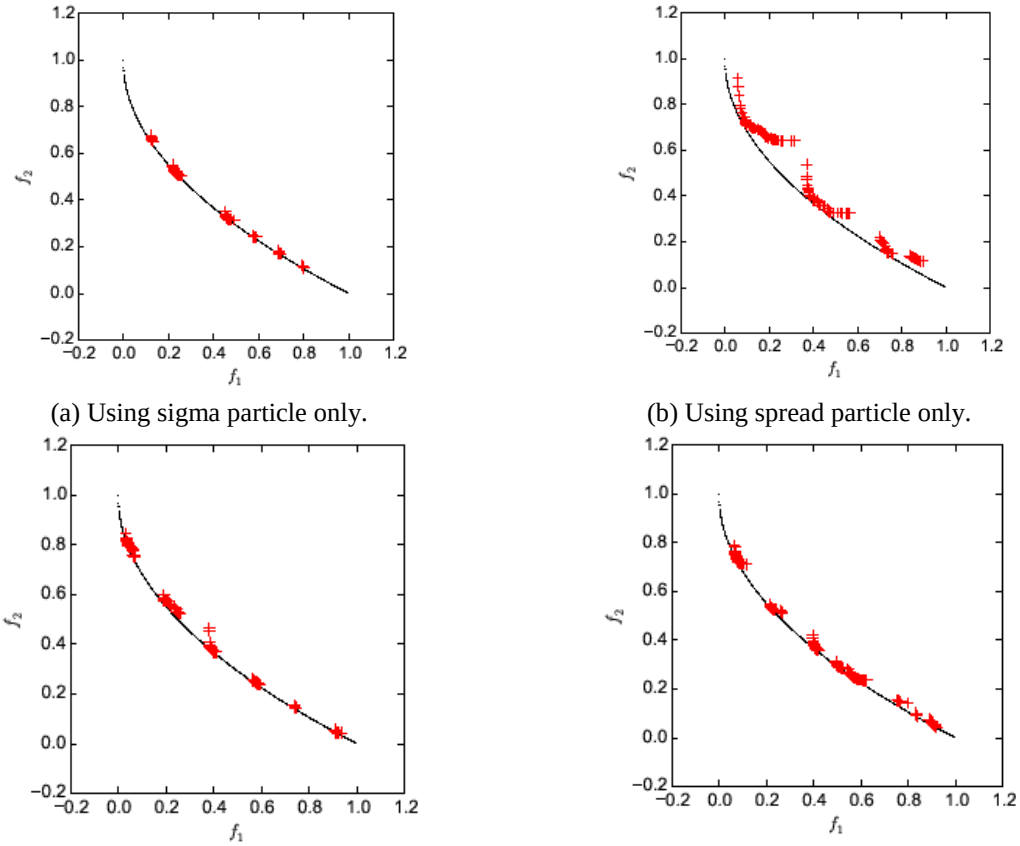
Problem	IGD	GD	SP	MG	CM
UF01	0.06189	0.00791	0.00623	0.24514	0.08910
UF02	0.04491	0.00258	0.01143	0.23481	0.04935
UF03	0.17045	0.17015	0.01771	0.39712	0.04349
UF04	0.05487	0.05489	0.00827	0.09442	0.04307
UF05	0.99020	1.17301	0.06908	1.12065	0.17490
UF06	0.31116	0.33294	0.02348	0.43837	0.12244
UF07	0.05386	0.01030	0.01586	0.18942	0.08037

Table3. *Results for a swarm containing only spread particles.*

Problem	IGD	GD	SP	MG	CM
UF01	0.08357	0.05679	0.01199	0.28950	0.06294
UF02	0.05285	0.02067	0.00674	0.20797	0.01283
UF03	0.24235	0.09586	0.00637	0.67538	0.01457
UF04	0.09224	0.10241	0.00807	0.15422	0.03516
UF05	0.40920	0.23988	0.01869	0.81787	0.04059
UF06	0.41591	0.21311	0.01111	0.74441	0.02480
UF07	0.14598	0.04280	0.01451	0.41531	0.03978

Table4. *Results for a heterogeneous swarm*

Problem	IGD	GD	SP	MG	CM
UF01	0.06110	0.01138	0.00499	0.25806	0.08174
UF02	0.02405	0.01082	0.00709	0.13189	0.02580
UF03	0.17354	0.09969	0.01031	0.52551	0.01965
UF04	0.06313	0.06704	0.00840	0.10469	0.02758
UF05	0.47906	0.51033	0.03454	0.75055	0.07856
UF06	0.28878	0.21472	0.01679	0.48759	0.04913
UF07	0.05653	0.02431	0.01062	0.17192	0.04860



(a) Using sigma particle only.

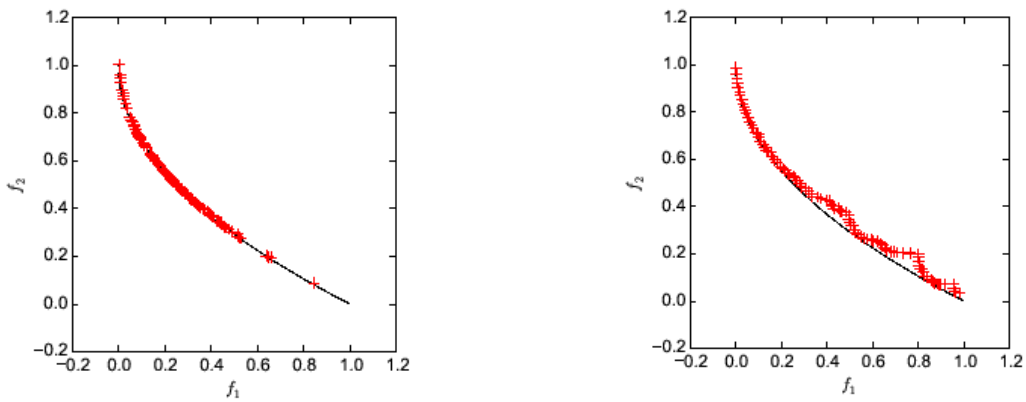
(b) Using spread particle only.

(c) Using closest particle only.

(d) Using heterogeneous swarm.

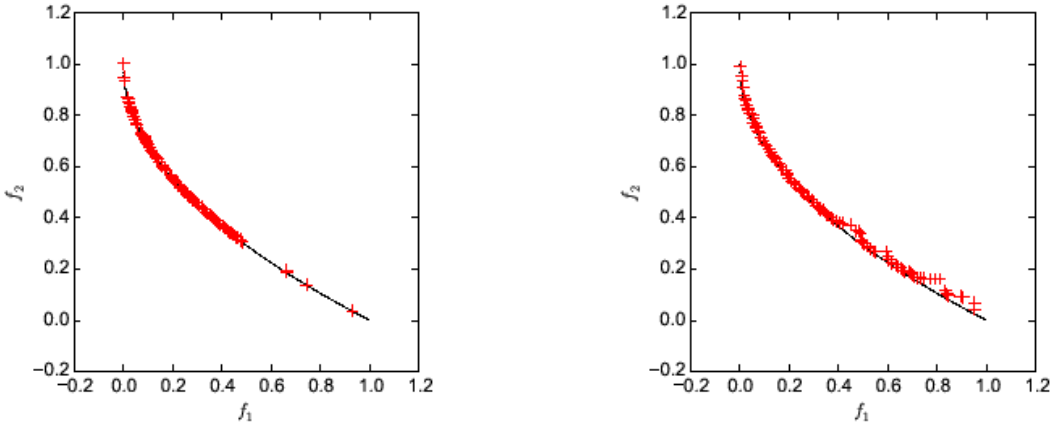
Fig2. *Results for UF01 using various swarms described in this paper.*

The results for problem UF02 can be seen in Figures 3a, 3b, 3c and 3d. It can be seen that the swarm using only ``sigma" particle has large parts of the PF missing although otherwise it lies almost exactly on the PF. The same can be said of ``closest" particle. ``Spread" particle show very good coverage of the PF although it is slightly further away. The same conclusion is indicated by the results tables. We can see that for ``spread" particle CM is the lowest indicating best coverage. We can also see that GD metric is an order of magnitude lower for ``sigma" and ``closest" particles as opposed to ``spread" particle. This indicates that those two particle types give results much closer to the real PF. If we look at the heterogeneous swarm we can see that it lowers CM compared to ``sigma" and ``closest" particles and it lowers GD compared to ``spread" particle. Which is in effect taking the best properties of the two particle types and negating each other's disadvantages. This can be most dramatically seen when comparing Figure 3d to Figures 3a, 3b and 3c. We also see that IGD is lowest for heterogeneous swarm.



(a) Using sigma particle only.

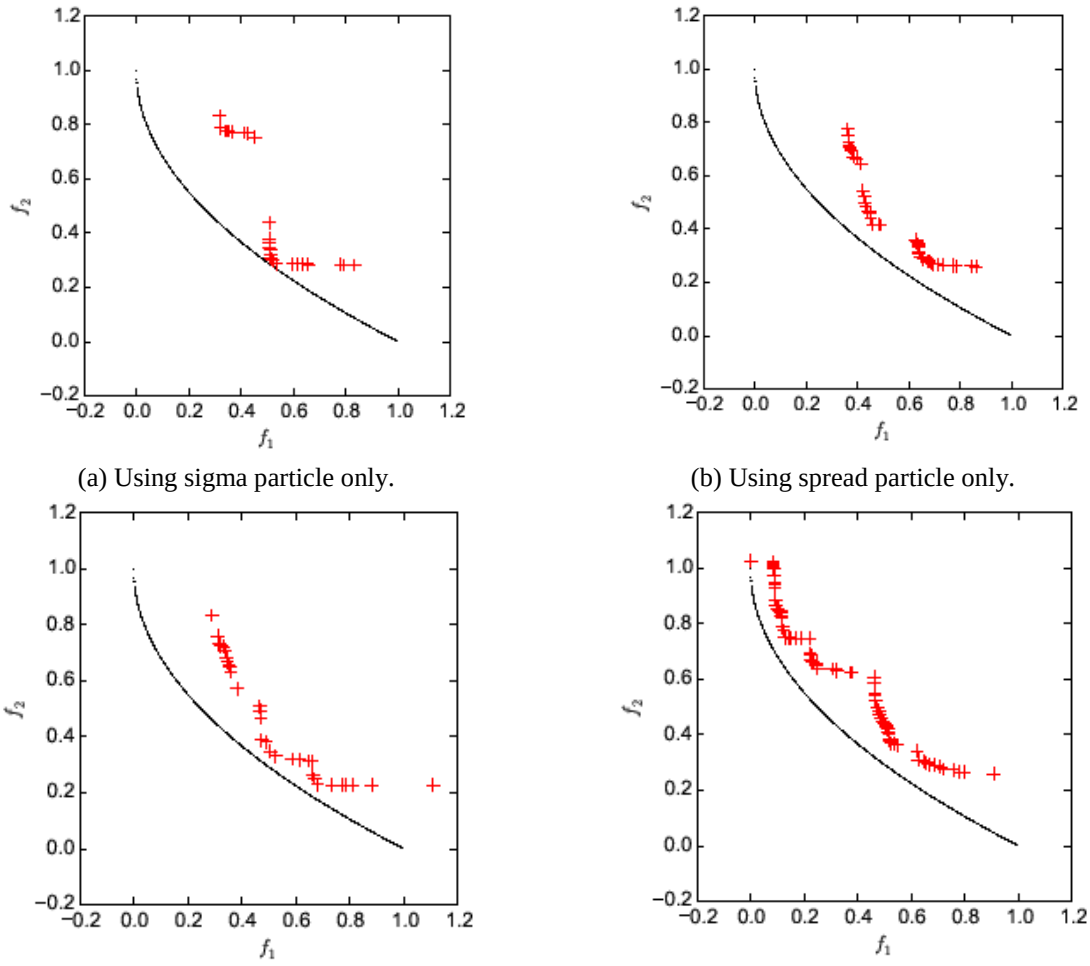
(b) Using spread particle only.



(c) Using closest particle only. (d) Using heterogeneous swarm.

Fig3. *Results for UF02 using various swarms described in this paper.*

If we look at the results for problem UF03 in the results tables we see an improvement in both GD and CM if using the heterogeneous swarm. The improvement is even more apparent in Figure 4d where we see that the heterogeneous swarm covers tries to cover both uniformity and closeness to PF unlike ``spread", ``closest" or ``sigma" particle swarms alone.

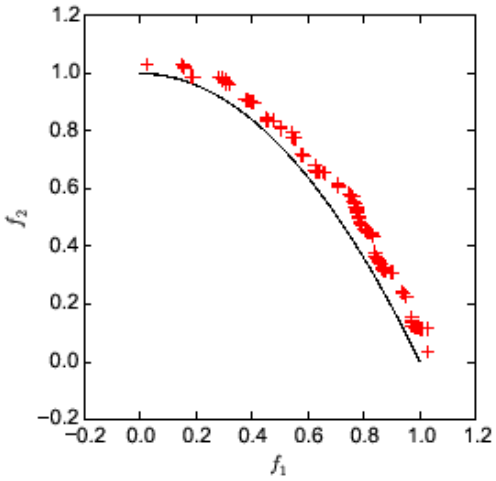


(a) Using sigma particle only. (b) Using spread particle only.

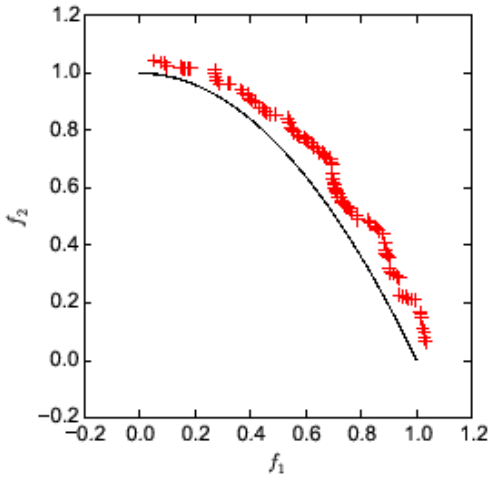
(c) Using closest particle only. (d) Using heterogeneous swarm.

Fig4. *Results for UF03 using various swarms described in this paper.*

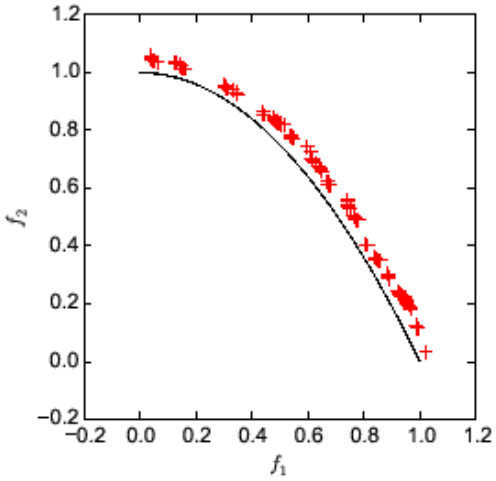
For problem UF04 we see the same results. GD metric is reduced for the heterogeneous swarms compared to ``spread" particle only and CM metric is reduced for heterogeneous swarms compared to using ``sigma" or ``closest" particles only. This can also be seen in Figure 5d.



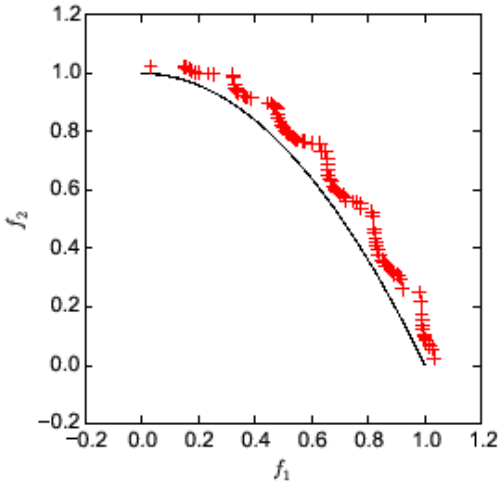
(a) Using sigma particle only.



(b) Using spread particle only.



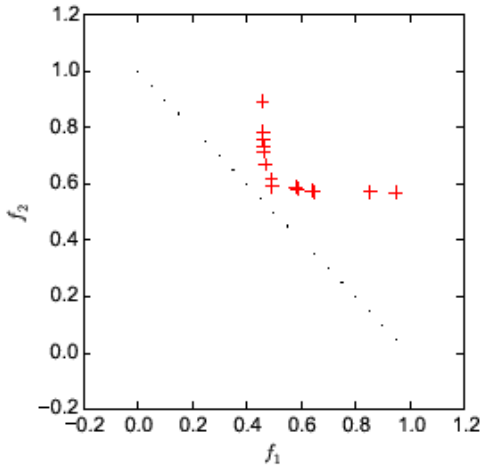
(c) Using closest particle only.



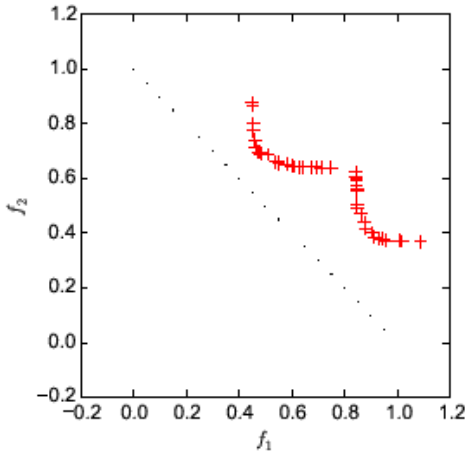
(d) Using heterogeneous swarm.

Fig5. *Results for UF04 using various swarms described in this paper.*

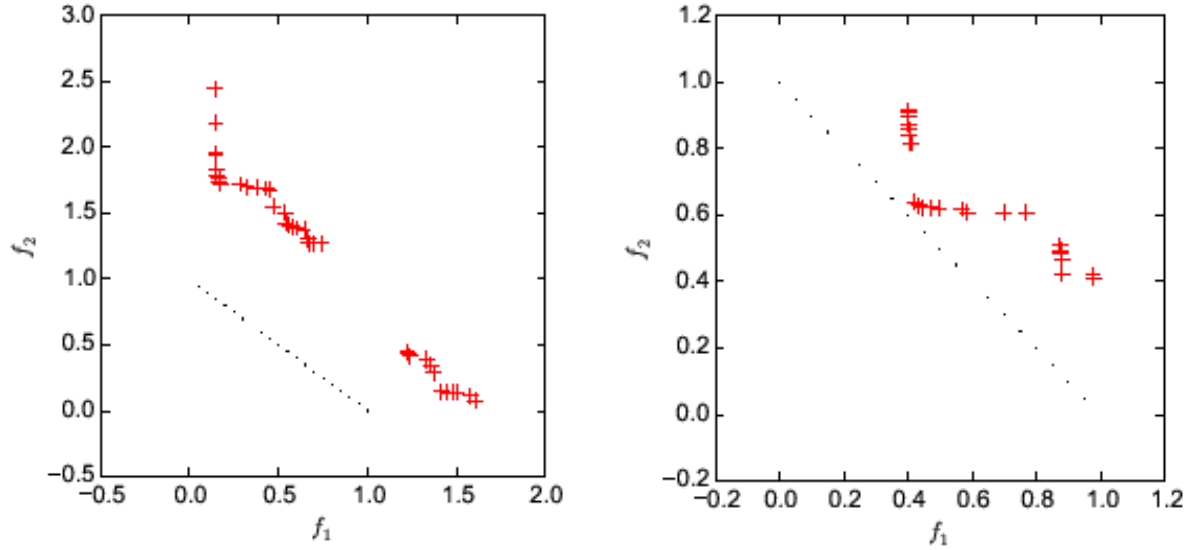
Problems UF05 and UF06 do not seem to be handled well by our algorithms. Their shared property is that they both have discrete Pareto sets which means that methods designed expecting continuous Pareto sets do not work well with them. It can maybe be solved by designing a separate particle for handling such problems although what that would entail is not entirely clear to us at the moment. Despite of this the trend of the two types of particles complementing one another's advantages is maintained as can be seen in the results tables.



(a) Using sigma particle only.

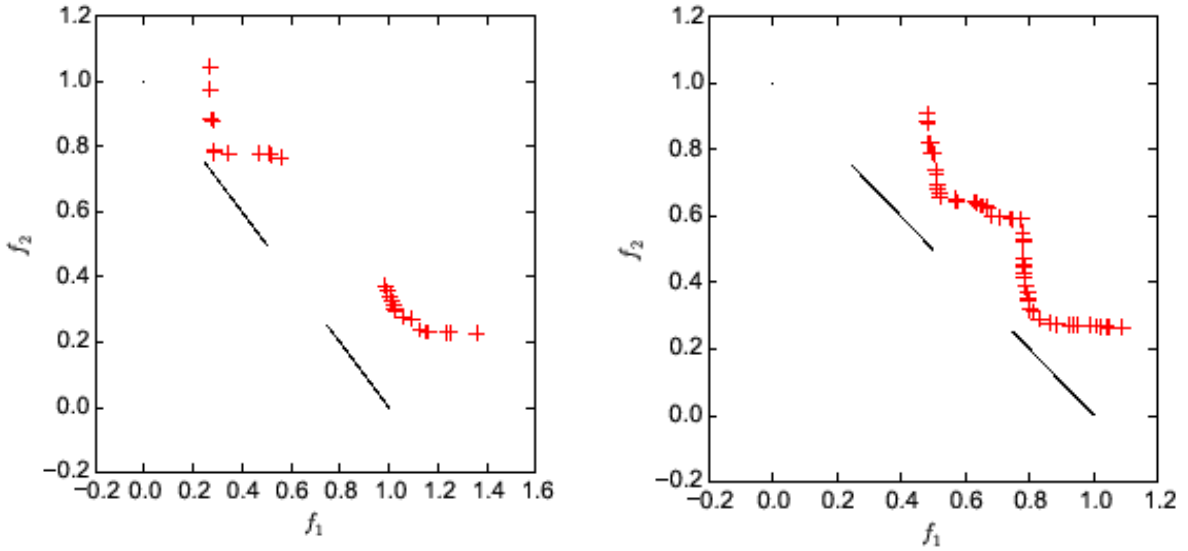


(b) Using spread particle only.

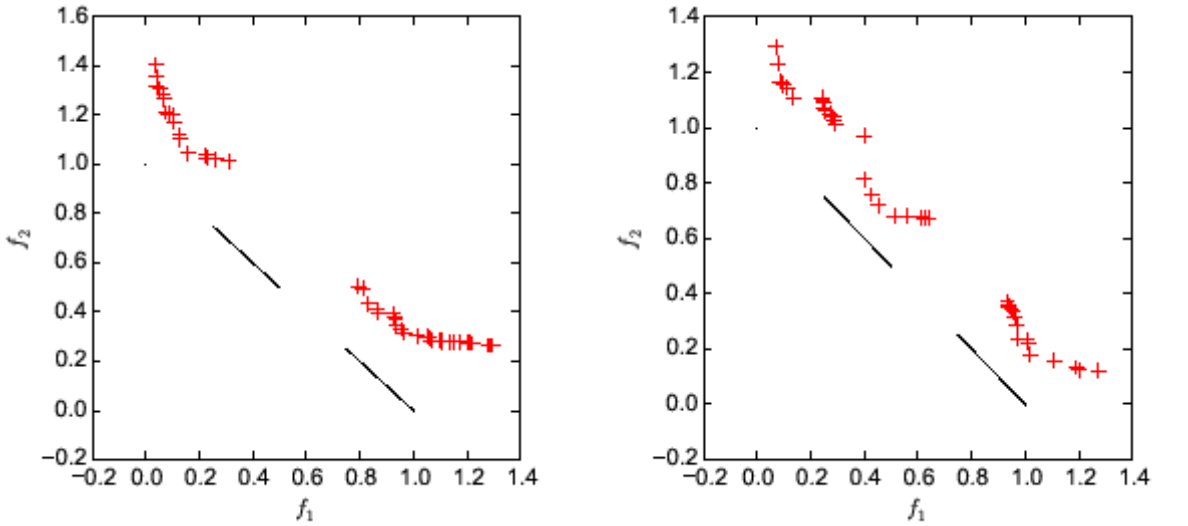


(c) Using closest particle only. (d) Using heterogeneous swarm.

Fig6. *Results for UF05 using various swarms described in this paper.*



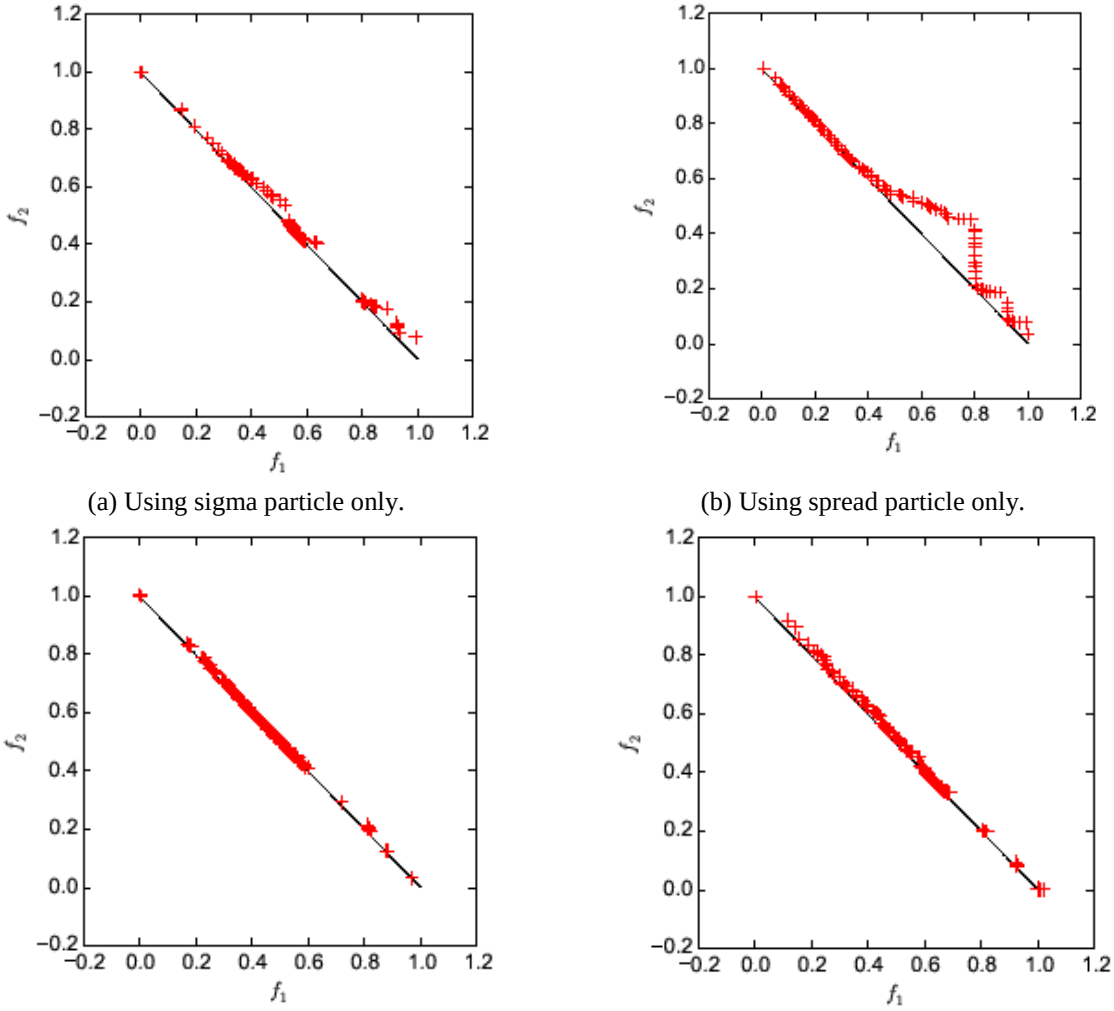
(a) Using sigma particle only. (b) Using spread particle only.



(c) Using closest particle only. (d) Using heterogeneous swarm.

Fig7. *Results for UF06 using various swarms described in this paper.*

The advantages of suing the proposed heterogeneous swarm can be seen in the case of UF07 as well. When comparing Figure 8d to Figures 8a, 8b and 8c we see that heterogeneous swarm gives the best performance of the four types visually. The resulting set has fewer missing areas of the PF but also is quite close to the real PF. Using only the ``spread" particle gives good coverage but poor distance and using ``sigma" or ``closest" particles give poor coverage. Using heterogeneous swarm on the other hand we see an improvement in both areas. The same can be seen when looking at the values of various metrics in the results tables.



(a) Using sigma particle only.

(b) Using spread particle only.

(c) Using closest particle only.

(d) Using heterogeneous swarm.

Fig8. *Results for UF07 using various swarms described in this paper.*

5. CONCLUSIONS

In this paper we have shown how disadvantages of certain types of PSO particle's position and velocity update rules can be offset by using different types of particles in the same swarm sharing information via a common non-dominated point archive. We also propose a new particle type that is designed to take in to account information about uniformity of the PF approximation. This particle type flies towards the areas of the search space that correspond to poorly covered areas of the PF assuming continuous PF. To further develop these ideas there are several key areas of work that we think are worthwhile:

- Develop more particle types with different properties using different kinds of information to emphasize different areas of multi-objective algorithm performance. An example could include particle that relies on previous p_i values similar to the NSGA-II algorithm and others.

- Incorporate the concept of PSO topology. It was shown for the global optimization case that the way in which particles' are connected in to a neighborhood is important.
- Improve the spread particle so that it can be better generalized to more than two objectives.
- Develop particles designed for problems with discrete Pareto frontiers. This will allow to hopefully get more accurate solutions to problems UF05 and UF06, since these are the ones where our method does worst.

Data in support of our thesis is given in terms of values of various metrics designed to measure multi-objective algorithm performance as well as graphically.

REFERENCES

[1] Paul S Andrews. An investigation into mutation operators for particle swarm optimization. In Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, pages 1044–1051. IEEE, 2006.

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

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

[4] Carlos A Coello Coello and Maximino Salazar Lechuga. Mopso: A proposal for multiple objective particle swarm optimization. In Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on, volume 2, pages 1051– 1056. IEEE, 2002.

[5] Marco Antonio Montes de Oca, Jorge Pe˜na, Thomas Stutzle, Carlo Pinciroli, and Marco Dorigo. Heterogeneous particle swarm optimizers. In Evolutionary Computation, 2009. CEC'09. IEEE Congress on, pages 698–705. IEEE, 2009.

[6] Kalyanmoy Deb. Multi-objective optimization. In Search methodologies, pages 403–449. Springer, 2014.

[7] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. Evolutionary Computation, IEEE Transactions on, 6(2):182–197, 2002.

[8] Andries P Engelbrecht. Heterogeneous particle swarm optimization. In Swarm Intelligence, pages 191–202. Springer, 2010.

[9] Jonathan E Fieldsend and Sameer Singh. A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. 2002.

[10] Natsuki Higashi and Hitoshi Iba. Particle swarm optimization with gaussian mutation. In Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE, pages 72–79. IEEE, 2003.

[11] Xiaohui Hu and Russell Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In Computational Intelligence, Proceedings of the World on Congress on, volume 2, pages 1677–1681. Ieee, 2002.

[12] Russell C. Eberhart James Kennedy. Particle swarm optimization. IEEE International Conference on Neural Networks, Proceedings, 4:1942 – 1948, 1995.

[13] Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how? 2001.

[14] Hui Li and Qingfu Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. Evolutionary Computation, IEEE Transactions on, 13(2):284–302, 2009.

[15] Sanaz Mostaghim and J¨urgen Teich . Strategies for finding good local guides in multi-objective particle swarm optimization (mopso). In Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE, pages 26–33. IEEE, 2003.

[16] Konstantinos E Parsopoulos and Michael N Vrahatis. Particle swarm optimization method in multiobjective problems. In Proceedings of the 2002 ACM symposium on Applied computing, pages 603–607. ACM, 2002.

[17] Carlo R Raquel and Prospero C Naval Jr. An effective use of crowding distance in multiobjective particle swarm optimization. In Proceedings of the 2005 conference on Genetic and evolutionary computation, pages 257–264. ACM, 2005.

[18] Margarita Reyes-Sierra and CA Coello Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. International journal of computational intelligence research, 2(3):287–308, 2006.

[19] Jason R Schott. Fault tolerant design using single and multicriteria genetic algorithm optimization. Technical report, DTIC Document, 1995.

[20] Margarita Reyes Sierra and Carlos A Coello Coello. Improving pso-based multi-objective optimization using crowding, mutation and-dominance. In Evolutionary multi-criterion optimization, pages 505–519. Springer, 2005.

[21] David A Van Veldhuizen and Gary B Lamont. On measuring multiobjective evolutionary algorithm performance. In Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, volume 1, pages 204–211. IEEE, 2000.

[22] Qingfu Zhang, Aimin Zhou, Shizheng Zhao, Ponnuthurai Nagaratnam Suganthan, Wudong Liu, and Santosh Tiwari. Multiobjective optimization test instances for the cec 2009 special session and competition. University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report, 2008

AUTHOR'S BIOGRAPHY

Vytautas Jančauskas is currently pursuing a PhD degree in Computer Science from Vilnius University in Vilnius, Lithuania.