Particle Swarm Optimization and Fitness Sharing to solve Multi-Objective Optimization Problems

Maximino Salazar-Lechuga
School of Computer Science
The University of Birmingham, UK
m.s.lechuga@cs.bham.ac.uk

Jonathan E. Rowe
School of Computer Science
The University of Birmingham, UK
j.e.rowe@cs.bham.ac.uk

Abstract- The particle swarm optimization algorithm has been shown to be a competitive heuristic to solve multi-objective optimization problems. Also, fitness sharing concepts have shown to be significant when used by multi-objective optimization methods. In this paper we introduce an algorithm that makes use of these two main concepts, particle swarm optimization and fitness sharing to tackle multi-objective optimization problems.

1 Introduction

The Particle Swarm Optimization (PSO) is a non-linear function optimization technique of recent development [13]. This technique has good performance, low computational cost and is easy to implement. Due to those characteristics, plus the similarities that this technique share with evolutionary algorithms, evolutionary computation scientists have been attracted to study this heuristic more closely.

Originally PSO was conceived to solve single-objective problems. Although previous work has been done to enhance the PSO to deal with multi-objective optimization problems (MOPs) [2] and it has been paired with fitness sharing for multimodal function optimization [17], to the best of our knowledge, PSO techniques have not made use of fitness sharing to guide (or improve) the search of the particles for the global optima (Pareto front), for solving MPOs.

One of the characteristics that made PSO (or evolutionary algorithms) so attractive to solve MOPs is due to its population-based solutions mechanism. Thanks to this mechanism, these kind of heuristics are capable of providing several solutions in one execution, in contrast to traditional techniques where one execution is capable to produce just one single solution.

First we will briefly talk about particle swarm optimization, fitness sharing and multi-objective optimization concepts. Then we will introduce an algorithm to tackle multi-objective problems based on particle swarm optimization and fitness sharing. After that, we will show the results of experiments performed against specialized test functions found in the literature and we compare it against three state-of-the-art heuristics in multi-objective optimization. Then we will discuss our results. At the end, conclusions and future work are given.

2 Background

In this section we will introduce some of the main concepts that we have used to produce our work.

2.1 Particle Swarm Optimization

This technique developed by Kennedy and Eberhart [14], is basically inspired by bird flocking. The main idea is based in the way birds travel when trying to find sources of food, or similarly the way a fish school will behave. The way this behavior is modeled, is that the "particles" inside the "swarm" (or population) are treated as solutions to a given problem; the solution space for that problem is where the particles will be moving or traveling through, searching for the best solutions to the problem. The particles will travel following two points in the space; a) a leader in the swarm, which is chosen according to the global best solution found so far[1], and b) its memory. Every particle has a memory, which is the best solution visited by that specific particle. Particle swarm optimization shares many similarities with genetic algorithms and/or evolutionary heuristics, and this is one of the main reasons which makes this technique attractive to use to solve problems like the ones we are interested in.

2.2 Fitness Sharing

The main idea of fitness sharing (Goldberg and Richardson [11], [8]) is to distribute a population of individuals along a set of resources. When an individual i is sharing resources with other individuals, its fitness f_i is degraded in proportion to the number and closeness to individuals that surround it. Fitness sharing for and individual i is defined as:

$$fShar_i = \frac{f_i}{\sum_{j=0}^{n} sharing_i^j} \tag{1}$$

where n is the number of individuals in the population,

$$sharing_i^j = \begin{cases} 1 - (d_i^j/\sigma_{share})^2 & \text{if } d_i^j < \sigma_{share} \\ 0 & \text{Otherwise} \end{cases} \tag{2}$$

σ_{share} is the distance we want the individuals to remain distant from each other, and d_i^j is a measure of distance between individual i and j.

[1] In PSO there are two main models in the way particles interact, the model used here is known as the *gbest* model, the other model is called *lbest* and is based in local connections (finding local best solutions, rather than a global one).

2.3 Multi-Objective Problems

When trying to solve single-objective problems, we only need to focus on the search for a single point in our search space (usually, it is what techniques solving single-objective problems do). On the other hand, when trying to solve MOPs we are looking not just for one single solution; instead, we are trying to find a set of solutions.

From this set of trade-off solutions the one that will be chosen, will depend on the needs of the decision maker (the person who takes the decisions).

MOPs are not trivial problems to solve and in order to solve them we need to have in mind some factors; as we just saw, we need to find a set of solutions for the problem and we need them as diverse as possible.

When tackling MOPs with traditional mathematical programming techniques, they tend to generate a single element of the set of solutions in one run. Moreover, traditional methods are susceptible to the shape or continuity of the set of solutions. EC paradigms are very suitable to solve MOPs because of their population-based nature, which can generate a set of solutions in just one run.

Formalization Regarding MOPs there are some definitions that formalize the problem. The general definition for a Multi-objective Optimization Problem is defined in Coello Coello et al. [4, pg. 6] as:

Definition 1. *Find a vector* $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ *satisfying* m *inequality constraints:*

$$g_i(\vec{x}) \geq 0 \quad i = 1, 2, \ldots, m \tag{3}$$

p *equality constraints:*

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p \tag{4}$$

and optimizing[2] *the vector function:*

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{5}$$

where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ *is the vector of decision variables.*

In order to optimize a vector function, there is another important concept tied to MOPs called "domination". In Deb [6, pg. 28] the concept of domination is defined as:

Definition 2. *A solution* $\vec{u}$ *is said to dominate another solution* $\vec{v}$*, if conditions 1 and 2 are true:*

1. *The solution* $\vec{u}$ *is no worse than* $\vec{v}$ *in all objectives, or* $f_i(\vec{u}) \not\rhd f_i(\vec{v})$ *for all* $i = 1, 2, ..., k$.
2. *The solution* $\vec{u}$ *is strictly better than* $\vec{v}$ *in at least one objective, or* $f_{\bar{i}}(\vec{u}) \lhd f_{\bar{i}}(\vec{v})$ *for at least one* $\bar{i} \in 1, 2, ..., k$.

The notation $\lhd$ used, is to express that a solution i is better than a solution j ($i \lhd j$), regardless of the type of problem (minimization or maximization). The notation $\rhd$ is used in the same way, $i \rhd j$ means that solution j is better than solution i.

Deb [6, pg. 31] defines a *non-dominated set* as follows:

[2]The concept of *optimization* is defined in terms of *domination*, explained later on.

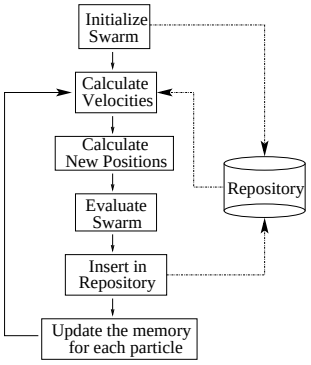

Figure 1: Diagram of the Algorithm

Definition 3. *Among a set of solutions* P*, the non-dominated set of solutions* P' *are those that are not dominated by any member of the set* P.

If within the definition we replace the set of solutions P by the feasible search space F ($P = F$), then the set of solutions in P' will be what is called *Pareto-optimal set* or *Pareto front*.

3 Proposed Approach

Goldberg [10], in his book, suggested the use of niche and speciation methods into the MOEA area. Back then, he thought they may be especially useful to avoid competition between distant members of the population, and in this way promoting and maintaining diversity. Following Goldberg's suggestions, multiple independent groups implemented his ideas [12]. The four main techniques are MOGA, NPGA, NSGA, and the Pareto-optimal ranking GA with sharing. This has lead us to the proposal of the work that we will describe in this section.

Our idea is to use the PSO technique to guide the search with help of fitness sharing to spread the particles along the Pareto front. Because of that we will be using fitness sharing in the objective space. Fitness sharing will help to our algorithm to maintain diversity between solutions, particles within high populated areas in the objective space will be less likely to be follow. In each iteration of the algorithm, the best particles found (those not dominated) will be inserted into an external repository (or external memory). This repository will help to guide the search for the next generations and will maintain a set of not dominated solutions until the end of the run, which is what we are looking for, the set of solutions forming the Pareto front.

The flow of the algorithm is shown in Figure 1. A briefly explanation follows for each of the steps given in the diagram:

1. In the first step all variables used by the algorithm are initialized. Particles ($pop[i]$) are initialized inside the search space and their memories ($pbest[i]$) are filled with the current positions. The external repository ($gbest[i]$) is filled with all the non-dominated particles. Fitness sharing ($fShar[i]$) is calculated for each of the particles in the repository.

According to the fitness sharing principle, which in words states that particles (or solutions) which have more particles in their vicinity will be less fit than those that have fewer particles surrounding their vicinity. The fitness assigned is given by:

$$fShar[i] = x/nCount_i \tag{6}$$

where $x = 10$; the value for x was arbitrarily chosen, a high value for $fShar$ (close to, or 10) will mean that the particle is not surrounded by other particles, or at least that there are particles not so close to this one.

$$nCount_i = \sum_{j=0}^{n} sharing_i^j \tag{7}$$

where n is the number of particles in the repository.

$$sharing_i^j = \begin{cases} 1 - (d_i^j/\sigma_{share})^2 & \text{if d} < \sigma_{share} \\ 0 & \text{Otherwise} \end{cases} \tag{8}$$

σ_{share} is the distance we want the particles to remain distant from each other; and d is a measure of distance between particles.

$$d_i^j = \sqrt{(part_i - part_j)^2} \tag{9}$$

2. Having a fitness sharing assigned for each particle in the repository, particles from the repository which will guide to the others into the next cycle will be chosen as leaders to be followed. They will be chosen according to a stochastic universal sampling method (Roulette Wheel). Particles with higher levels of fitness will be selected over the less fit ones. This will allow them to explore places less explored in the search space. The velocity for the particles is calculated as:

$$\begin{aligned} vel[i] = {} & w \times vel[i] + \\ & r_1 \times (pbest[i] - pop[i]) + \\ & r_2 \times (gbest[h] - pop[i]) \end{aligned} \tag{10}$$

where w is an inertia weight (we used a value of 0.4 for all of our experiments), $vel[i]$ is the previous velocity value, r_1 and r_2 are random values between 0 and 1, $pbest[i]$ is the best position found by the particle, $gbest[h]$ is the particle to be follow, and $pop[i]$ is the current position of the particle in the variable space.

3. New positions of the particles are calculated according to the velocities obtained in the previous step:

$$pop[i] = pop[i] + vel[i] \tag{11}$$

4. The new positions of the swarm are evaluated.

5. The repository is updated with the current solutions found by the particles. The criteria used to update the repository is dominance and fitness sharing. The particles that dominate the ones inside the repository will be inserted and all solutions dominated will be deleted, in this way we maintain the repository as the Pareto front found so far. In the case where the repository is full of non-dominated particles and a particle which is non-dominated by any in the repository wants to get into it, we compare their fitness sharing. We calculate the fitness sharing for the particle that wants to get into the repository and if is better than the worst fitness sharing for a particle in the repository, then the particle with worst fitness sharing is replaced by this new one. Fitness sharing for all particles is updated when inserting or deleting a particle from the repository. This is to maintain fitness sharing in an up to date state in case is used again when calculating velocities or when inserting particles into the repository.

6. Finally the memory of each particle is updated with the criteria of dominance if the current location of the particle dominates the one stored in its memory the current one replaces the one in memory.

4 Test and Comparison

For the purpose of this paper we will show the performance of our approach with four test functions and we will compare them against three well known techniques in multi-objective literature. The techniques are: MOPSO [3], NSGA-II [7] and PAES [15][3].

To measure performance of our approach against these three heuristics we chose three metrics. As is well known, assessing performance in multi objective problems is a multi-objective problem per se. Mainly, we need to accomplish these objectives:

- Proximity to the real Pareto front, we need to get as close as possible to the optimal solutions
- Diversity among solutions, we need to have a wide range of variety
- Maximize the extension of the solutions found, this will allow more diversity

The metrics to measure the performance are:

- **Generational Distance** [19] finds the average distance of the non-dominated set of solutions found from the Pareto optimal set:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{12}$$

where d_i is the Euclidean distance between solution i from the set of n non-dominated solutions found and the closest element from the Pareto optimal set (in objective space). Because this metric the average distance from the Pareto optimal set, a smaller value indicates more proximity.

[3] Source code for these heuristics was written by their respective authors and obtained from [1]. Source code of our approach can be obtained by e-mailing the first author of this paper.

- **Spacing** [18] measures how well distributed (spaced) the solutions in the non-dominated set found are:

$$S = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \overline{d})^2} \quad (13)$$

where d_i is the minimum value of the sum of the absolute difference for every objective function value between the i-th solution and all the n non-dominated solutions found, $d_i = min^n_{j=1 \wedge j \neq i}(\sum_{m=1}^{M} |f^i_m - f^j_m|)$, $\overline{d}$ is the mean value for all d_i, $\overline{d} = \sum_{i=1}^{n} d_i/n$. Because this metric measures the standard deviations of the distances between the solutions found, a smaller value will indicate that the solutions are uniformly spaced.

- **Maximum Spread** [20] gives a value which represents the maximum extension between the farthest solutions in the non-dominated set found, in a problem with two objectives, the value will be the Euclidean distance between the two farther solutions.

$$D = \sqrt{\sum_{m=1}^{M}\left(max^n_{i=1} f^i_m - min^n_{i=1} f^i_m\right)^2} \quad (14)$$

As previously, n is number of solutions in the non-dominated set, and M is the number of objectives in a given problem. In this metric a bigger value indicates better performance.

For this paper we have performed two sets of experiments. In the first set of experiments we have set the heuristics to find only 10 non-dominated solutions per run, and in the second set 100 non-dominated solutions. The purpose is to show how well our technique makes use of fitness sharing. Our believe is that, using a small size in the repository, our technique will still spread its solutions thanks to fitness sharing.

To allow a fair comparison between all the heuristics, they performed the same number of evaluations to the objective function in each test function. To compare and obtain statistics for each test function, we performed 30 runs for each technique used. The parameters used by MOPSO, NSGA-II and PAES for all the test functions were; MOPSO used: a mutation rate of 0.5, 30 divisions for its adaptive grid, and a real number representation, for the first set of experiments a population of 10 particles and a repository size of 10 particles, for the second set of experiments a population of 100 particles and repository size of 100 particles. NSGA-II used: a crossover rate of 0.8, a mutation probability of $1/x$, where x is the number of variables for the given problem, and a real number representation with tournament selection, for the first set of experiments a population of 10 individuals and for the second set of experiments a population of 100 individuals. PAES used: a depth value of 5, a mutation probability of $1/L$, where L refers to the length of the chromosomic binary string, that encodes the decision variables, for the first set of experiments an archive size of 10 individuals and for the second an archive size of 100. MOPSO-fs used a population of 10 particles and a repository size of 10 particles for the first set of experiments, and for the second a population of 100 particles and a repository size of 100 particles was used, a σ_{share} value was empirically set for each test function. The value of σ_{share} and the number of evaluations for each of the test functions will be specified in each test function section. In the following all the test functions will be minimization problems, unless stated otherwise.

4.1 Test function 1

The first test function, proposed by Fonseca [9], is the following:

$$f_1(\vec{x}) = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right) \quad (15)$$

$$f_2(\vec{x}) = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right) \quad (16)$$

where: $-4 \leq x_i \leq 4$, $i = 1, 2, 3$

For this problem all the heuristics were set to evaluate the objective function $30,000$ times (in both set of experiments, when trying to find 10 and 100 non-dominated solutions). Our technique was set with a σ_{share} value of 0.1 and 0.01 for the first and second set of experiments, respectively. In table 1 we can observe the statistical results obtained when comparing the four different approaches. In figure 2 we can observe a graphical comparison of the results for the four different techniques.

10 Non-Dominated Solutions					
		MOPSO-*fs*	**MOPSO**	**NSGA-II**	**PAES**
GD	*Avg.*	0.00445	0.00129	0.01230	0.01496
	S.D.	0.00202	4.78568 E-4	0.00438	0.00787
S	*Avg.*	0.03462	0.00915	0.09940	0.08875
	S.D.	0.01295	0.00626	0.02882	0.11219
D	*Avg.*	1.26037	0.15049	1.38831	0.72016
	S.D.	0.05572	0.08106	9.66090 E-6	0.11155
100 Non-Dominated Solutions					
		MOPSO-*fs*	**MOPSO**	**NSGA-II**	**PAES**
GD	*Avg.*	7.76357 E-4	7.55519 E-4	7.95112 E-4	0.00438
	S.D.	2.50235 E-5	2.36099 E-5	3.16017 E-5	0.00168
S	*Avg.*	0.00370	0.00885	0.00776	0.00886
	S.D.	4.99291 E-4	7.47956 E-4	5.85683 E-4	0.01536
D	*Avg.*	1.37259	1.36445	1.38826	0.73446
	S.D.	0.00792	0.01072	1.48355 E-4	0.10006

Table 1: This table corresponds to statistical results obtained from test function 1.

4.2 Test function 2

Our second test function, is a maximization problem proposed by Poloni [19]:

$$\text{Maximize } f_1(x_1, x_2) = -\left[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2\right] \quad (17)$$

$$\text{Maximize } f_2(x_1, x_2) = -\left[(x_1 + 3)^2 + (x_2 + 1)^2\right] \quad (18)$$

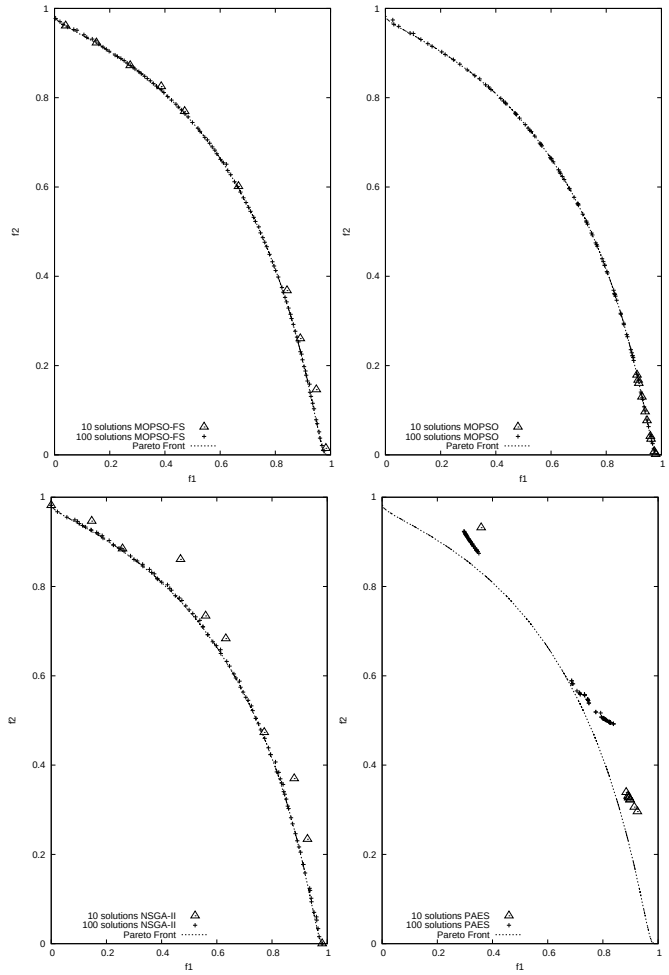

Figure 2: This graphical results were obtained from test function 1.

where: $-3.1416 \leq x_1, x_2 \leq 3.1416$,

$$A_1 = 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2 \quad (19)$$

$$A_2 = 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2 \quad (20)$$

$$B_1 = 0.5\sin x_1 - 2\cos x_1 + \sin x_2 - 1.5\cos x_2 \quad (21)$$

$$B_2 = 1.5\sin x_1 - \cos x_1 + 2.0\sin x_2 - 0.5\cos x_2 \quad (22)$$

For the second test function our technique used a σ_{share} value of 2.0 and 0.2 for the first and second set of experiments, respectively. All the heuristics performed $10,000$ evaluations to the objective function. Table 2 has the statistical values that we obtained from measuring the results, and figure 3 has the graphical representations of the set of non-dominated solutions found by each of the heuristics.

4.3 Test function 3

Kursawe's [16] is our third test function:

$$f_1(\vec{x}) = \sum_{i=1}^{n-1}\left(-10e^{(-0.2)*\sqrt{x_i^2+x_{i+1}^2}}\right) \quad (23)$$

10 Non-Dominated Solutions					
		MOPSO-*fs*	**MOPSO**	**NSGA-II**	**PAES**
GD	*Avg.*	0.03097	0.04759	0.18863	0.08499
	S.D.	0.06381	0.08258	0.19352	0.17326
S	*Avg.*	0.61763	0.40905	1.68223	3.05431
	S.D.	0.15088	0.54906	0.47373	3.05777
D	*Avg.*	27.91334	8.48245	30.75301	20.17722
	S.D.	4.37469	9.28997	1.36576	8.22772
100 Non-Dominated Solutions					
		MOPSO-*fs*	**MOPSO**	**NSGA-II**	**PAES**
GD	*Avg.*	0.00899	0.01195	0.00346	0.02144
	S.D.	0.01372	0.01517	0.00750	0.03703
S	*Avg.*	0.10479	0.14831	0.09264	0.20094
	S.D.	0.09395	0.09037	0.00754	0.22917
D	*Avg.*	30.10451	30.26833	29.63863	23.74446
	S.D.	1.12996	1.24542	0.47579	8.40573

Table 2: This table corresponds to statistical results obtained from test function 2.

$$f_2(\vec{x}) = \sum_{i=1}^{n}\left(|x_i|^{0.8} + 5\sin\left(x_i\right)^3\right) \quad (24)$$

where: $-5 \leq x_i \leq 5$, $i = 1, 2, 3$

For this test function a σ_{share} value of 1.0 and 0.1 was used for the first and second set of experiment, respectively, and the number of evaluations for the test function was $30,000$ times. Table 3 contains the statics of the comparison for this test function between the 4 heuristics. Figure 4 has a graphical representation of the set of non-dominated solutions for all of the techniques.

10 Non-Dominated Solutions					
		MOPSO-*fs*	**MOPSO**	**NSGA-II**	**PAES**
GD	*Avg.*	0.03271	0.00489	0.06392	0.04310
	S.D.	0.02578	0.00151	0.04363	0.06881
S	*Avg.*	0.31937	0.03071	0.30970	1.12292
	S.D.	0.09857	0.04100	0.24977	0.51570
D	*Avg.*	12.07941	0.51892	2.96087	9.61927
	S.D.	0.74749	0.64138	1.26063	2.53561
100 Non-Dominated Solutions					
		MOPSO-*fs*	**MOPSO**	**NSGA-II**	**PAES**
GD	*Avg.*	0.00482	0.00426	0.00425	0.00583
	S.D.	8.02159 E-4	6.65039 E-4	0.00215	0.00760
S	*Avg.*	0.07846	0.09540	0.06722	0.20549
	S.D.	0.015005	0.01919	0.02968	0.09822
D	*Avg.*	12.93078	12.90576	11.08729	11.98795
	S.D.	0.02938	0.03919	0.46939	1.20020

Table 3: This table corresponds to statistical results obtained from test function 3.

4.4 Test function 4

Our fourth test function, proposed by Deb [5], is

$$f_1(x_1, x_2) = x \quad (25)$$

$$f_2(x_1, x_2) = (1 + 10y) * \left[1 - \left(\frac{x}{1+10y}\right)^{\alpha} - \frac{x}{1+10y}\sin\left(2\pi q x\right)\right] \quad (26)$$

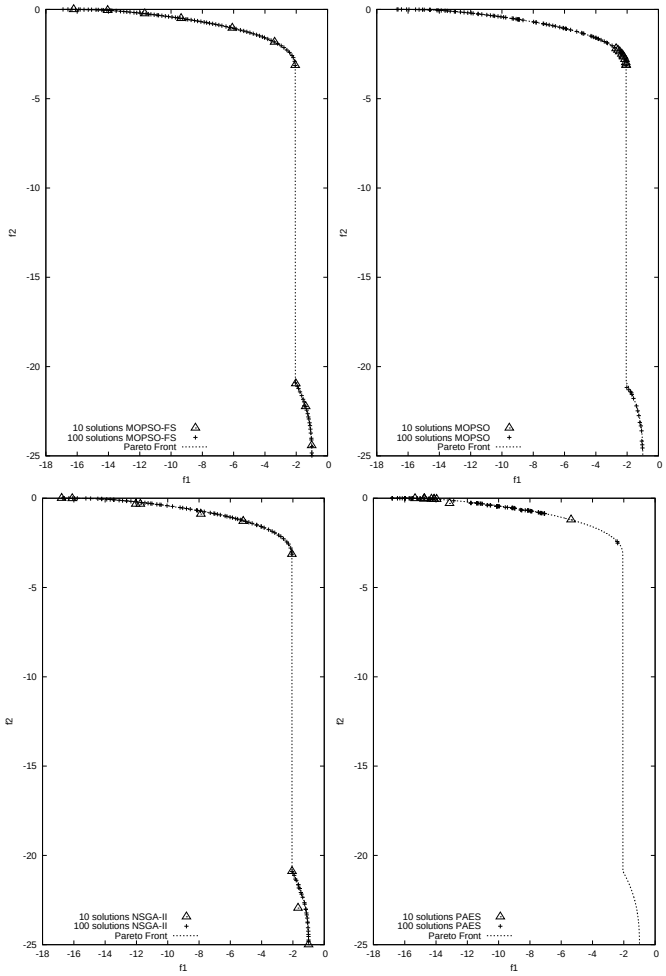

Figure 3: This plots correspond to the results obtained from function 2.

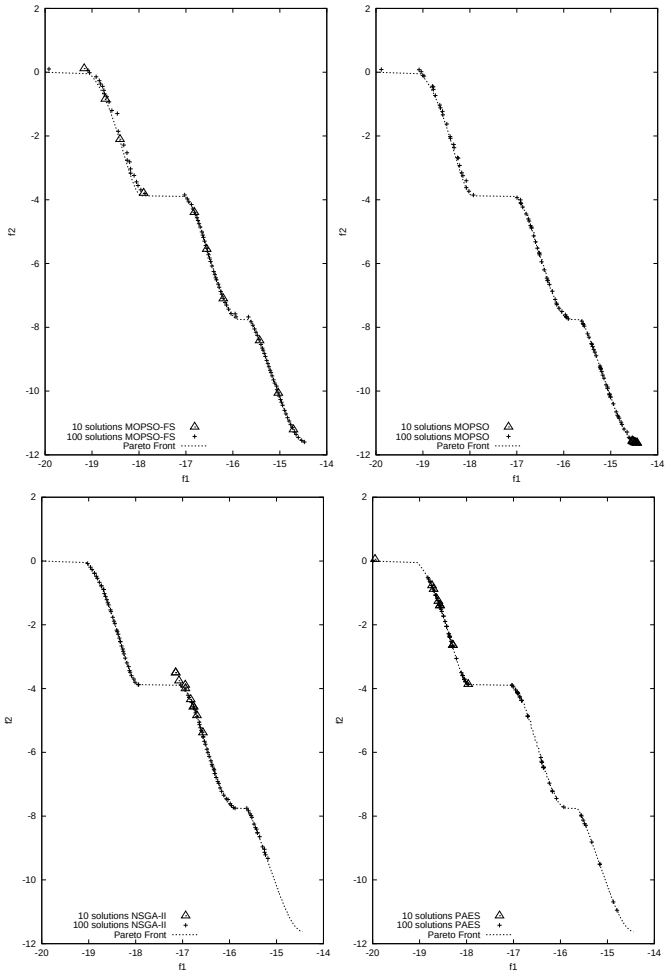

Figure 4: This plots correspond to the results obtained from test function 3.

where: $0 \leq x_1, x_2 \leq 1$, and $q = 4, \alpha = 2$

For the fourth test function we used a σ_{share} value of 0.2 and 0.01 for the first and second set of experiments, respectively, and each of the heuristics performed $5,000$ evaluations to the objective function. Table 4 and figure 5 correspond to statistics and graphical representations to this problem, respectively.

5 Discussion of Results

Regarding our first set of experiments, which involves finding 10 non-dominated solutions, in the graphical representations (figures 2, 3, 4 and 5) of the results obtained by all the heuristics for all the test functions, we can observe that our MOPSO-*fs* obtains better results than the other techniques. We attribute this to the way fitness sharing distributes the particles along the Pareto front.

From a statistical point of view, for the first test function, as we can observe in table 1, MOPSO has the better values for the GD and S metric, but this is due to the fact that MOPSO is concentrating all its solutions in a very small portion of the Pareto front (see figure 2), not necessarily due to a better set of dispersed solutions. NSGA-II is the only heuristic that graphically (figure 2) shows comparable results to our MOPSO-*fs*, but statistically our technique shows better results to approach the Pareto front and also to disperse the solutions (smaller GD and S values in table 1).

For the second test function, a very similar situation as that presented with the first test function can be observed. MOPSO presents better values for the S metric (see table 2), but this is due to a concentration of all its solutions found in a very small portion of the Pareto front (figure 3). NSGA-II again is the only heuristic with graphical comparable results to our technique, but again ours presents better values for GD and S (table 2).

For the third test function, again MOPSO presents better values for the GD and S metric (see table 3). This due to a concentration of all its solutions found in a very small portion of the Pareto front, as can be seen by the very small D value that it presents (see table 3 and figure 4). As for the performance against NSGA-II and PAES for this problem our MOPSO-*fs* performs much better than them.

For the fourth test function, NSGA-II and PAES are the ones that show similar graphical performance than our MOPSO-*fs* (figure 5), but as can be seen in the statistical results our technique outperforms the way it spaces its solutions found (see S values in table 4).

10 Non-Dominated Solutions					
		MOPSO-*fs*	**MOPSO**	**NSGA-II**	**PAES**
GD	*Avg.*	0.00603	0.00888	0.00315	0.10488
	S.D.	0.02426	0.01409	0.00557	0.29243
S	*Avg.*	0.06018	0.06091	0.11928	0.23787
	S.D.	0.03104	0.18165	0.04093	0.38772
D	*Avg.*	1.58116	0.38268	1.58140	2.35599
	S.D.	0.44776	0.42131	0.40759	2.19558
100 Non-Dominated Solutions					
		MOPSO-*fs*	**MOPSO**	**NSGA-II**	**PAES**
GD	*Avg.*	0.00197	3.07034 E-4	3.29989 E-4	0.01149
	S.D.	0.00434	2.61333 E-5	2.49222 E-5	0.02366
S	*Avg.*	0.01696	0.00881	0.00732	0.03591
	S.D.	0.03240	0.00211	5.90846 E-4	0.04753
D	*Avg.*	1.83941	1.68699	1.69026	2.48424
	S.D.	0.38098	0.00573	9.20826 E-5	1.57197

Table 4: This table corresponds to statistical results obtained from test function 4.

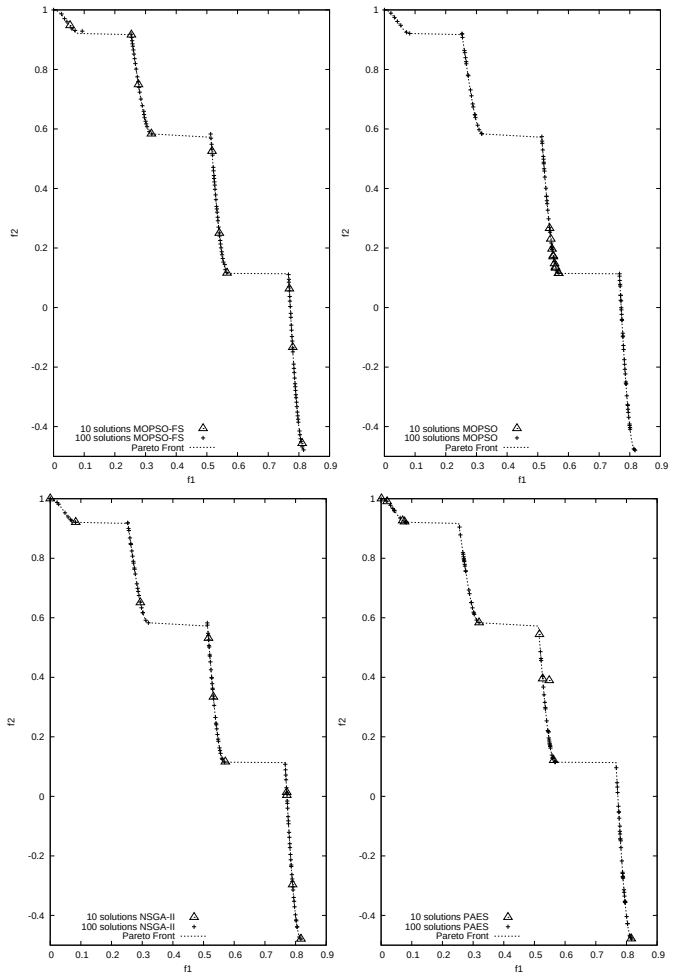

Figure 5: This plots correspond to the results obtained from test function 4.

Regarding the second set of experiments, finding 100 non-dominated solutions, statistics show that all methods are very competitive for all the test functions. For the graphical comparisons MOPSO-*fs* seems to perform slightly better in terms of distribution and extension over the Pareto front, again due to its fitness sharing mechanism. In figure 2 we can see that all the techniques (except PAES) reach the Pareto Front but the one that has better distribution of solutions is our technique (with a lower S value in table 1). The third test function presents three disjointed Pareto fronts, for the two most inferior fronts our technique presented a very good distribution of its solutions, while having a bit more difficulty on the top most front, but as can be seen the rest of the heuristics present difficulties in at least one of the fronts, and in general don't present evenly distributed solutions. In figures 3 and 5, we can notice that the MOPSO-*fs* solutions are more evenly distributed than the solutions given for the rest of the techniques.

6 Conclusions and Future Work

As we have seen from the experiments, we can conclude that we have built a competitive heuristic using a particle swarm optimizer and fitness sharing to help it to deal with multi-objective problems. Our experiments have shown us that MOPSO-*fs* is specifically superior when finding a small number of non-dominated solutions. Because of the way the mechanism of selection of particles that enter into the repository (where we store the non-dominated solutions) is implemented, first by filling it with non-dominated solutions, and second by discarding solutions with lower fitness sharing, we have ensured convergence (getting in as much non-dominated particles as possible) and diversity (removing particles in populated areas).

Still we believe some work can be done in order to find improvements for this algorithm:

- We are currently working in ways to help this algorithm to automatically find an appropriate value of σ_{share}, which until now has been empirically tuned. More research has to be done in this area in order to improve our results.

- As mentioned, PSO has two main models of connection between particles *lbest* and *gbest*, which is the way in which particles share knowledge. We have just experimented with the *gbest* connection model. Experimenting with a different connection model to notice any valuable improvements in the technique is another area in which we want to work.

In the work presented here we have used test functions of low dimensionality and relatively low complexity, we need to experiment with test functions of higher dimensionality and more complexity.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions.

The first author acknowledges support from CONACyT (Consejo Nacional de Ciencia y Tecnología) through a scholarship to pursue a Ph.D. degree in computer science at the University of Birmingham, Birmingham, U.K.

Bibliography

[1] Coello Coello, C. A. (2001). List of references on evolutionary multiobjective optimization. Online. `http://www.lania.mx/~ccoello/EMOO/`.

[2] Coello Coello, C. A. and Salazar Lechuga, M. (2002). MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Procedings of the Congress on Evolutionay Computation (CEC'02)*, volume 1, pages 1051–1056, Honolulu, HI. IEEE Press.

[3] Coello Coello, C. A., Toscano Pulido, G., and Salazar Lechuga, M. (2004). Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279.

[4] Coello Coello, C. A., Van Veldhuizen, D. A., and Lamont, G. B. (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer, New York.

[5] Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230.

[6] Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, Chichester.

[7] Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T. (2000). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858. Springer.

[8] Deb, K. and Goldberg, D. E. (1989). An Investigation of Niche and Species Formation in Genetic Function Optimization. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50. George Mason University, Morgan Kaufmann Publishers.

[9] Fonseca, C. M. and Fleming, P. J. (1995). Multiobjective genetic algorithms made easy: selection sharing and mating restriction. In *Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sheffield, UK.

[10] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

[11] Goldberg, D. E. and Richardson, J. (1987). Genetic Algorithms with Sharing for Multimodal Function Optimization. In Grefenstette, J., editor, *Proceedings of the 2nd International Conference on Genetic Algorithms*, pages 41–49, Hillsdale, New Jersey. Lawrence Erlbaum Associates.

[12] Horn, J. (1997). Multicriterion decision making. In Bäck, T., Fogel, D., and Michalewicz, Z., editors, *The Handbook of Evolutionary Computation*, pages F1.9:1–F1.9:15. Oxford University Press.

[13] Kennedy, J. and Eberhart, R. C. (1995). Particle Swarm Optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, New Jersey. IEEE Service Center.

[14] Kennedy, J. and Eberhart, R. C. (2001). *Swarm Intelligence*. Morgan Kaufmann, San Francisco, California.

[15] Knowles, J. and Corne, D. (1999). The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation. In *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 98–105, Mayflower Hotel, Washington D.C., USA. IEEE Press.

[16] Kursawe, F. (1991). A variant of evolution strategies for vector optimization. In *PPSN I: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, pages 193–197, London, UK. Springer-Verlag.

[17] Li, T., Wei, C., and Pei, W. (2003). PSO with sharing for multimodal function optimization. In *Proceedings of the 2003 International Conference on Neural Networks and Signal Processing*, volume 1, pages 450–453.

[18] Schott, J. R. (1995). Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis, Massachusetts Institute of Technology.

[19] Van Veldhuizen, D. A. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations*. PhD thesis, Air Force Institute of Technology Air University.

[20] Zitzler, E. and Kalyanmoy Deb, L. T. (1999). Comparison of multiobjective evolutionary algorithms: Empirical results (revised version). Technical report, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich, Switzerland.