Particle Swarm Optimization Method in Multiobjective Problems

K.E. Parsopoulos
Department of Mathematics,
University of Patras Artificial
Intelligence Research Center (UPAIRC),
University of Patras,
GR–26110 Patras, Greece
kostasp@math.upatras.gr

M.N. Vrahatis
Department of Mathematics,
University of Patras Artificial
Intelligence Research Center (UPAIRC),
University of Patras,
GR–26110 Patras, Greece
vrahatis@math.upatras.gr

ABSTRACT

This paper constitutes a first study of the Particle Swarm Optimization (PSO) method in Multiobjective Optimization (MO) problems. The ability of PSO to detect Pareto Optimal points and capture the shape of the Pareto Front is studied through experiments on well–known non–trivial test functions. The Weighted Aggregation technique with fixed or adaptive weights is considered. Furthermore, critical aspects of the VEGA approach for Multiobjective Optimization using Genetic Algorithms are adapted to the PSO framework in order to develop a multi–swarm PSO that can cope effectively with MO problems. Conclusions are derived and ideas for further research are proposed.

Keywords

Particle Swarm Optimization, Multiobjective Optimization

1. INTRODUCTION

Multiobjective Optimization (MO) problems are very common, especially in engineering applications, due to the multicriteria nature of most real–world problems. Design of complex hardware/software systems [20], atomic structure determination of proteins [2], potential function parameter optimization [18], x–ray diffraction pattern recognition [14], curve fitting [1] and production scheduling [19] are such applications, where two or more, sometimes competing and/or incommensurable objective functions have to be minimized simultaneously. In contrast to the single–objective optimization case, where the optimal solution is clearly defined, in MO problems there is a whole set of trade–offs giving rise to numerous *Pareto Optimal* solutions. These points are optimal solutions for the MO problem when all objectives are simultaneously considered.

Although the traditional Gradient–based optimization techniques can be used to detect Pareto Optimal solutions, this approach suffers from two critical drawbacks; (I) the objectives have to be aggregated in one single objective function and (II) only one solution can be detected per optimization run. The inherent difficulty to foreknow which aggregation of the objectives is appropriate in addition to the heavy computational cost of Gradient–based techniques, necessitates the development of more efficient and rigorous methods. Evolutionary Algorithms (EA) seem to be especially suited to MO problems due to their abilities, to search simultaneously for multiple Pareto Optimal solutions and, perform better global search of the search space [20].

The Particle Swarm Optimization (PSO) is a Swarm Intelligence method that models social behavior to guide swarms of particles towards the most promising regions of the search space [3]. PSO has proved to be efficient at solving Unconstrained Global Optimization and engineering problems [4, 10, 11, 12, 13, 17]. It is easily implemented, using either binary or floating point encoding, and it usually results in faster convergence rates than the Genetic Algorithms [7]. Although PSO's performance, in single–objective optimization tasks, has been extensively studied, there are insufficient results for MO problems thus far. In this paper a first study of the PSO's performance in MO problems is presented through experiments on well–known test functions.

In the next section the basic concepts and terminology of MO are briefly presented. In Section 3 the PSO method is described and briefly analyzed. In Section 4 the performance of PSO in terms of finding the Pareto Front in Weighted Aggregation cases is exhibited, while in Section 5 a modified PSO is used to perform MO similar to the VEGA system. In the last section conclusions are derived and further research directions are proposed.

2. BASIC CONCEPTS

Let X be an n–dimensional search space and $f_i(x)$, $i = 1, \ldots, k$, be k objective functions defined over X. Assuming,

$$g_i(x) \leqslant 0, \qquad i = 1, \ldots, m,$$

be m inequality constraints, the MO problem can be stated as finding a vector $x^* = (x_1^*, x_2^*, \ldots, x_n^*) \in X$ that satisfies the constraints and optimizes (without loss of generality we consider only the minimization case) the vector function $\mathbf{f}(x) = [f_1(x), f_2(x), \ldots, f_k(x)]$. The objective functions may be in conflict, thus, in most cases it is impossible

SAC 2002, Madrid, Spain
© 2002 ACM 1-58113-445-2/02/03 ...$5.00.

to obtain for all objectives the global minimum at the same point. The goal of MO is to provide a set of Pareto Optimal solutions to the aforementioned problem.

Let $u = (u_1, \dots, u_k)$ and $v = (v_1, \dots, v_k)$ be two vectors. Then, u *dominates* v if and only if $u_i \leqslant v_i$, $i = 1, \dots, k$, and $u_i < v_i$ for at least one component. This property is known as *Pareto Dominance* and it is used to define the Pareto Optimal points. Thus, a solution x of the MO problem is said to be *Pareto Optimal* if and only if there does not exist another solution y such that $\mathbf{f}(y)$ dominates $\mathbf{f}(x)$. The set of all Pareto Optimal solutions of an MO problem is called *Pareto Optimal Set* and we denote it with $\mathcal{P}^*$. The set $\mathcal{PF}^* = \{(f_1(x), \dots, f_k(x)) \mid x \in \mathcal{P}^*\}$ is called *Pareto Front*. A Pareto Front $\mathcal{PF}^*$ is called *convex* if and only if $\forall\, u, v \in \mathcal{PF}^*$, $\forall\, \lambda \in (0,1)$, $\exists\, w \in \mathcal{PF}^* : \lambda\|u\| + (1-\lambda)\|v\| \geqslant \|w\|$. Respectively, it is called *concave* if and only if $\forall\, u, v \in \mathcal{PF}^*$, $\forall\, \lambda \in (0,1)$, $\exists\, w \in \mathcal{PF}^* : \lambda\|u\| + (1-\lambda)\|v\| \leqslant \|w\|$. A Pareto Front can be convex, concave or partially convex and/or concave and/or discontinuous. The last three cases present the greatest difficulty for most MO techniques.

In the next section, the PSO technique is briefly presented.

3. PARTICLE SWARM OPTIMIZATION

The PSO is a Swarm Intelligence method that differs from well–known Evolutionary Computation algorithms, such as the Genetic Algorithms [5, 6, 7], in that the population is not manipulated through operators inspired by the human DNA procedures. Instead, in PSO, the population dynamics simulate the behavior of a "birds' flock", where social sharing of information takes place and individuals profit from the discoveries and previous experience of all other companions during the search for food. Thus, each companion, called *particle*, in the population, which is called *swarm*, is assumed to "fly" over the search space looking for promising regions on the landscape. For example, in the single–objective minimization case, such regions possess lower function values than others previously visited. In this context, each particle is treated as a point into the search space, which adjusts its own "flying" according to its flying experience as well as the flying experience of other particles.

First, let us define the notation adopted in this paper: assuming that the search space is D–dimensional, the i-th particle of the swarm is represented by the D–dimensional vector $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ and the best particle in the swarm, i.e. the particle with the smallest function value, is denoted by the index g. The best previous position (i.e. the position giving the best function value) of the i-th particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$, while the position change (velocity) of the i-th particle is represented as $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$. Following this notation, the particles are manipulated according to the following equations

$$v_{id} = w v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}), \quad (1)$$
$$x_{id} = x_{id} + \chi\, v_{id}, \quad (2)$$

where $d = 1, 2, \dots, D$; N is the size of the population; $i = 1, 2, \dots, N$; χ is a constriction factor which controls and constricts the velocity's magnitude; w is the inertia weight; c_1 and c_2 are two positive constants; r_1 and r_2 are two random numbers within the range $[0, 1]$.

Equation (1) determines the i-th particle's new velocity as a function of three terms: the particle's previous velocity; the distance between the best previous position of the particle and its current position, and finally; the distance between the swarm's best experience (the position of the best particle in the swarm) and the i-th particle's current position. Then, according to Equation (2), the i-th particle "flies" towards a new position. In general, the performance of each particle is measured according to a fitness function, which is problem–dependent. In optimization problems, the fitness function is usually the objective function under consideration.

The role of the inertia weight w is considered to be crucial for the PSO's convergence. The inertia weight is employed to control the impact of the previous history of velocities on the current velocity of each particle. Thus, the parameter w regulates the trade–off between global and local exploration ability of the swarm. A large inertia weight facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration, i.e. fine–tuning the current search area. A suitable value for the inertia weight w balances the global and local exploration ability and, consequently, reduces the number of iterations required to locate the optimum solution. A general rule of thumb suggests that it is better to initially set the inertia to a large value, in order to make better global exploration of the search space, and gradually decrease it to get more refined solutions. Thus, a time-decreasing inertia weight value is used. The initial swarm can be generated either randomly or using a Sobol sequence generator [15], which ensures that the particles will be uniformly distributed within the search space.

From the above discussion, it is obvious that PSO resembles, to some extent, the "mutation" operator of Genetic Algorithms through the position update Equations (1) and (2). Note, however, that in PSO the "mutation" operator is guided by the particle's own "flying" experience and benefits from the swarm's "flying" experience. In other words, PSO is considered as performing mutation with a *"conscience"*, as pointed out by Eberhart and Shi [4].

In the next section, some well–known benchmark MO problems are described and results from the application of PSO using Weighted Aggregation approaches are exhibited.

4. THE WEIGHTED AGGREGATION APPROACH

The *Weighted Aggregation* is the most common approach for coping with MO problems. According to this approach, all the objectives are summed to a weighted combination $F = \sum_{i=1}^{k} w_i f_i(x)$, where w_i, $i = 1, \dots, k$, are non–negative weights. It is usually assumed that $\sum_{i=1}^{k} w_i = 1$. These weights can be either fixed or dynamically adapted during the optimization.

If the weights are fixed then we are in the case of the *Conventional Weighted Aggregation* (CWA). Using this approach only a single Pareto Optimal point can be obtained per optimization run and a priori knowledge of the search space is required in order to choose the appropriate weights [9]. Thus, the search has to be repeated several times to obtain a desired number of Pareto Optimal points. Yet, this is not suitable in most problems due to time limitations and heavy computational costs. Moreover, CWA is unable to detect solutions in concave regions of the Pareto Front [9].

Other Weighted Aggregation approaches have been proposed to alleviate the limitations of the CWA. For a two-objective MO problem, the weights can be modified during the optimization, according to the following equations,

$$w_1(t) = \mathrm{sign}\big(\sin(2\pi t/F)\big),\ w_2(t) = 1 - w_1(t),$$

where t is the iteration's index and F is the weights' change frequency. This is the well-known *Bang-Bang Weighted Aggregation* (BWA) approach, according to which, the weights are changed abruptly due to the usage of the sign($\cdot$) function. Alternatively, the weights can be gradually modified according to the equations,

$$w_1(t) = |\sin(2\pi t/F)|,\ w_2(t) = 1 - w_1(t).$$

This is called *Dynamic Weighted Aggregation* (DWA). The slow change of the weights forces the optimizer to keep moving on the Pareto Front, if it is convex, performing better than in the BWA case. If the Pareto Front is concave then the performance using DWA and BWA is almost identical when Genetic Algorithms are used [9].

The three different approaches that are mentioned above have been applied in experiments using the PSO technique, with $F = 100$ for the BWA and $F = 200$ for the DWA case respectively. The benchmark problems that were used are defined in [8, 21]:

- Function F_1 (convex, uniform Pareto Front): $f_1 = \frac{1}{n}\sum_{i=1}^{n} x_i^2,\ f_2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - 2)^2$.
- Function F_2 (convex, non-uniform Pareto Front): $f_1 = x_1,\ g = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i,\ f_2 = g\left(1 - \sqrt{f_1/g}\right)$.
- Function F_3 (concave Pareto Front): $f_1 = x_1,\ g = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i,\ f_2 = g\left(1 - (f_1/g)^2\right)$.
- Function F_4 (neither purely convex nor purely concave Pareto Front): $f_1 = x_1,\ g = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$, $f_2 = g\left(1 - \sqrt[4]{f_1/g} - (f_1/g)^4\right)$.
- Function F_5 (Pareto Front that consists of separated convex parts): $f_1 = x_1,\ g = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$, $f_2 = g\left(1 - \sqrt{f_1/g} - (f_1/g)\ \sin(10\pi f_1)\right)$.

Although simple and consisting of only two objectives, these problems are considered difficult (especially F_3, F_4 and F_5) due to the shape of the Pareto Front (purely or partially concave, discontinuous etc.). In order to have comparable results in finding the Pareto Front of the benchmark problems with the results provided in [9] for the Evolutionary Algorithms case, we used the pseudocode provided in [9] to build and maintain the archive of Pareto solutions and we performed all simulations using the same parameter values. Thus, we performed 150 iterations of PSO for each problem, with $x \in [0, 1]^2$. The PSO parameters were fixed $c_1 = c_2 = 0.5$ and the inertia w was gradually decreased from 1 towards 0.4. The size of the swarm depended on the problem but never exceeded 40.

The first experiments were done using the CWA approach with a small swarm of 10 particles. The desired number of Pareto Optimal points was 20 for the functions F_1, F_2, F_3 and 40 for F_4. Thus we had to run the algorithm 20 times for the first three functions and 40 for the fourth. The obtained Pareto Fronts are exhibited in Figures 1 and 2 and they are similar to those obtained from the Evolutionary Algorithm in [9] but with very low computational cost and fast convergence rate (less than 2 minutes were needed for each function).

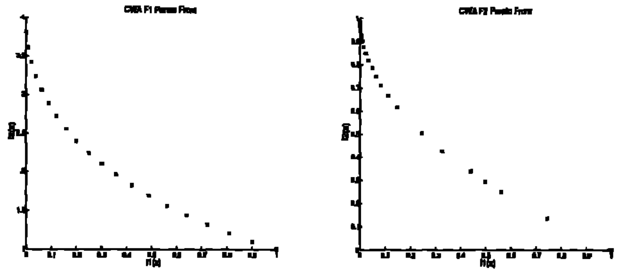

Figure 1: CWA approach results for F_1 and F_2.

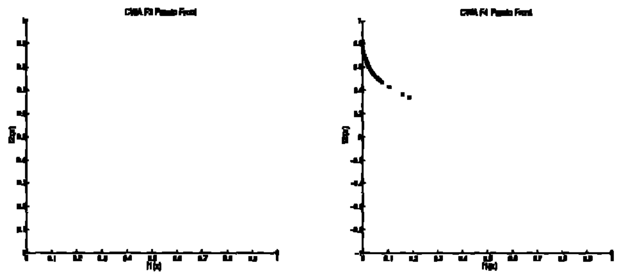

Figure 2: CWA approach results for F_3 and F_4.

As it was expected, the CWA approach was able to detect the Pareto Front in F_1, where it is convex and uniform, in F_2, where it is convex and non-uniform, and in F_4 it detected only the convex parts. In F_3 (concave case) it was unable to detect Pareto Optimal points other than the two ends of the Pareto Front. The obtained Pareto Fronts for the experiments using the BWA and DWA approaches are exhibited in Figures 3–6.

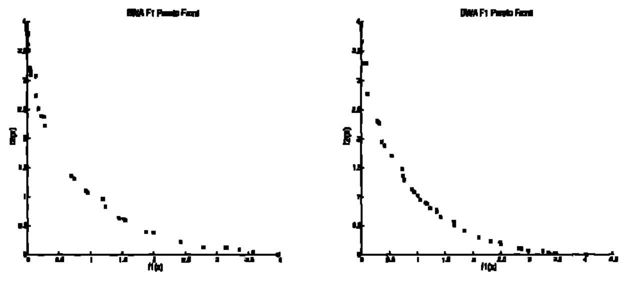

Figure 3: BWA (left) and DWA (right) approaches' results for the function F_1.

It is clear that PSO succeed in capturing the shape of Pareto Front in each case. The results are better using the DWA approach, with the exception of the case of the concave Pareto Front of the function F_3, at which the BWA approach performed better. Swarm size was equal to 20 for all simulations, except for the function F_5, for which it was set to 40.

The MO problems can be alternatively solved using population-based non-Pareto approaches instead of Weights Ag-

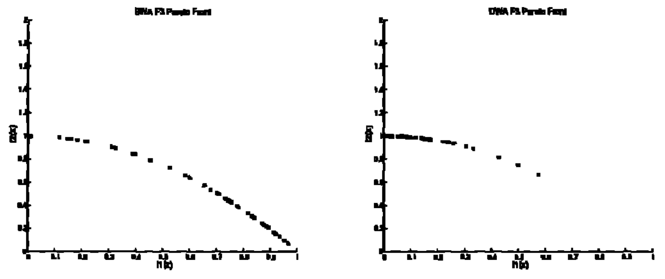
Figure 4: BWA (left) and DWA (right) approaches' results for the function F_3.

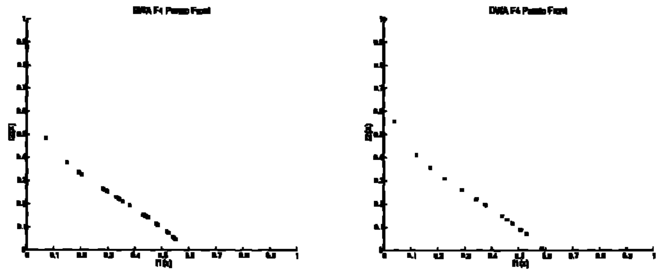
Figure 5: BWA (left) and DWA (right) approaches' results for the function F_4.

gregating approaches. One such approach is VEGA (Vector Evaluated Genetic Algorithm), developed by Schaffer [16]. In the next section, a modification of the PSO algorithm borrowing ideas from VEGA is used in MO problems.

5. A POPULATION-BASED NON-PARETO APPROACH

According to the VEGA approach, fractions of the next generation, or subpopulations, are selected from the old generation according to each of the objectives, separately. After shuffling all these sub-populations together, crossover and mutation are applied to generate the new population [16].

The main ideas of VEGA were adopted and modified to fit the PSO framework, developing the *VEPSO* algorithm. We used two swarms to solve the five benchmark problems F_1–F_5. Each swarm was evaluated according to one of the objectives but, information coming from the other swarm was used to determine the change of the velocities. Specifically, the best particle of the second swarm was used for the determination of the new velocities of the first swarm's particles, using Equation (1), and vice versa. Alternatively, the best positions of the second swarm can be used, in conjunction with the best particle of the second swarm, for the evaluation of the velocities of the first swarm, and vice versa. The obtained Pareto Fronts for all benchmark problems are exhibited in Figures 7– 11. The left part of each figure is the Pareto Front obtained using only the best particle of the other swarm, while the right part is obtained using both the best particle and the best previous positions of the other swarm. With the exception of Function F_2, no significant difference between the two approaches was observed. For each experiment, two swarms of size 20 were used and the algorithm ran for 200 iterations.

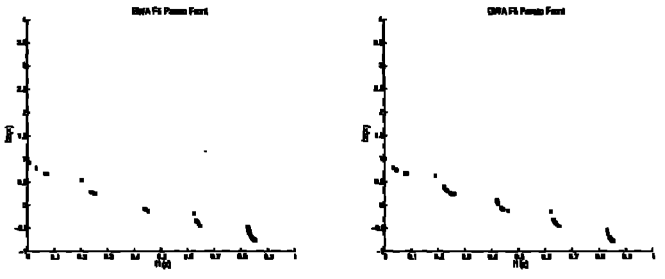
Figure 6: BWA (left) and DWA (right) approaches' results for the function F_5.

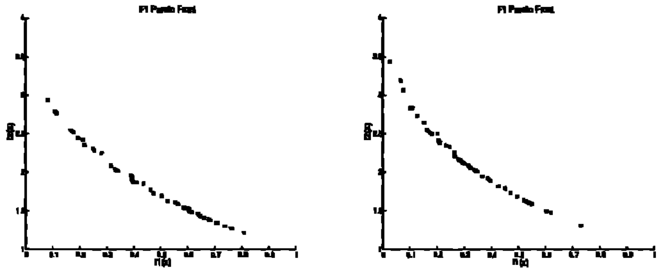
Figure 7: VEPSO results for the function F_1.

6. CONCLUSIONS

A first study of the performance of the PSO technique in MO problems has been presented. The PSO method solved efficiently well known test problems, including difficult cases for MO techniques, such as concavity and/or discontinuity of the Pareto Front. We used low dimensional objectives in order to investigate the simplest cases first. Besides, it is a general feeling that two objectives are sufficient to reflect essential aspects of MO [21]. Promising results were obtained even when the size of the swarm was very small. In addition to the Weighted Aggregation cases, a modified version of PSO (VEPSO) that resembles the VEGA ideas was also developed and applied on the same problems, with promising results.

Further research will include investigation of the performance of PSO in higher-dimensional problems with more than two objectives. Especially in the case of the Population-Based Non-Pareto Approach, a random selection of the objective, in problems with more than two objectives, seems very interesting. Theoretical work is also required to fully understand the swarm's dynamics and behavior during the optimization.

7. REFERENCES

[1] H. Ahonen, P.A. Desouza, and V.K. Garg. A Genetic Algorithm for Fitting Lorentzian Line-Shapes in Mossbauer-Spectra. *Nuclear Instruments & Methods in Physics Research*, 124(4):633–638, 1997.

[2] T.S. Bush, C.R.A. Catlow, and P.D. Battle. Evolutionary Programming Techniques for Predicting Inorganic Crystal-Structures. *J. Materials Chemistry*, 5(8):1269–1272, 1995.

[3] R.C. Eberhart and J. Kennedy. A New Optimizer Using Particle Swarm Theory. In *Proc. 6th Int. Symp.*

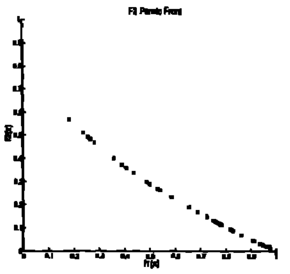

Figure 8: VEPSO results for the function F_2.

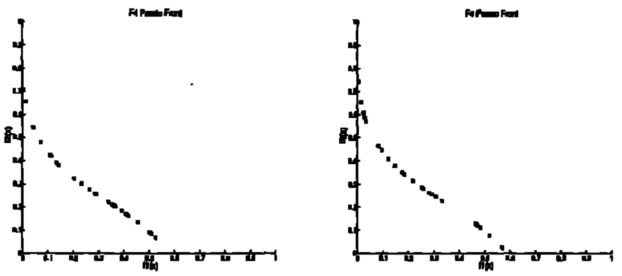

Figure 10: VEPSO results for the function F_4.

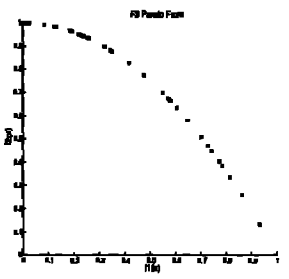

Figure 9: VEPSO results for the function F_3.

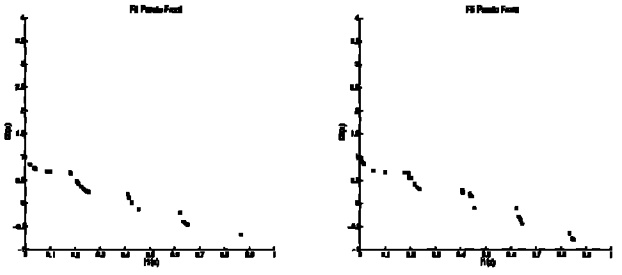

Figure 11: VEPSO results for the function F_5.

on Micro Mach. & Hum. Sci., pages 39–43, 1995.

[4] R.C. Eberhart and Y.H. Shi. Evolving Artificial Neural Networks. In *Proc. Int. Conf. on Neural Networks and Brain*, 1998.

[5] R.C. Eberhart, P.K. Simpson, and R.W. Dobbins. *Computational Intelligence PC Tools*. Academic Press Professional, Boston, 1996.

[6] J. Kennedy and R.C. Eberhart. Particle Swarm Optimization. In *Proc. of the IEEE Int. Conf. Neural Networks*, pages 1942–1948, 1995.

[7] J. Kennedy and R.C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[8] J.D. Knowles and D.W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategies. *Evolutionary Computation*, 8(2):149–172, 2000.

[9] Y. Jin, M. Olhofer, and B. Sendhoff. Dynamic Weighted Aggregation for Evolutionary Multi–Objective Optimization: Why Does It Work and How?. In *Proc. GECCO 2001 Conf.*, 2001.

[10] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, and M.N. Vrahatis. Objective Function "Stretching" to Alleviate Convergence to Local Minima. *Nonlinear Analysis, TMA*, 47(5):3419–3424, 2000.

[11] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, and M.N. Vrahatis. Stretching Technique for Obtaining Global Minimizers Through Particle Swarm Optimization. In *Proc. Particle Swarm Optimization Workshop*, pages 22–29, 2001.

[12] K.E. Parsopoulos and M.N. Vrahatis. Modification of the Particle Swarm Optimizer for Locating All the Global Minima. V. Kurkova, N. Steele, R. Neruda, M. Karny (Eds.), *Artificial Neural Networks and Genetic Algorithms*, pages 324–327, Springer, 2001.

[13] K.E. Parsopoulos and M.N. Vrahatis. Particle Swarm Optimizer in Noisy and Continuously Changing Environments. M.H. Hamza (Ed.), *Artificial Intelligence and Soft Computing*, pages 289–294, IASTED/ACTA Press, 2001.

[14] W. Paszkowicz. Application of the Smooth Genetic Algorithm for Indexing Powder Patterns: Test for the Orthorhombic System. *Materials Science Forum*, 228(1&2):19–24, 1996.

[15] W.H. Press, W.T. Vetterling, S.A. Teukolsky, and B.P. Flannery. *Numerical Recipes in Fortran 77*, Cambridge University Press, Cambridge, 1992.

[16] J.D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and their Applications: Proc. first Int. Conf. on Genetic Algorithms*, pages 93–100, 1985.

[17] Y. Shi, R.C. Eberhart, and Y. Chen. Implementation of Evolutionary Fuzzy Systems. *IEEE Trans. Fuzzy Systems*, 7(2):109–119, 1999.

[18] A.J. Skinner and J.Q. Broughton. Neural Networks in Computational Materials Science: Training Algorithms. *Modelling and Simulation in Material Science and Engineering*, 3(3):371–390, 1995.

[19] K. Swinehart, M. Yasin, and E. Guimaraes. Applying an Analytical Approach to Shop–Floor Scheduling: A Case–Study. *Int. J. Materials & Product Technology*, 11(1–2):98–107, 1996.

[20] E. Zitzler. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Ph.D. thesis, Swiss Fed. Inst. Techn. Zurich, 1999.

[21] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolution Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.