An Extension of Particle Swarm Optimization that Can Handle Multiple Objectives

Carlos A. Coello Coello[1], Gregorio Toscano Pulido[1] and Maximino Salazar Lechuga[2]

1. CINVESTAV-IPN, Sección de Computación, Dept. of Elect. Eng., México, D.F. 07300 - MEXICO
ccoello@cs.cinvestav.mx, gtoscano@computacion.cs.cinvestav.mx
2. Maestría en Inteligencia Artificial, Universidad Veracruzana, Xalapa, Veracruz 91000 - MEXICO
reactivo@prodigy.net.mx

Abstract: In this paper, we present an extension of the heuristic called "particle swarm optimization" (PSO) that is able to deal with multiobjective optimization problems. Our approach uses the concept of Pareto dominance to determine the flight direction of a particle and it maintains previously found nondominated vectors in a global repository that is later used by other particles to guide their own flight. We also incorporate a mutation operator that increases the exploratory capabilities of the algorithm. The approach is validated using several test functions taken from the evolutionary multiobjective optimization literature. Our results indicate that the approach is competitive with state-of-the-art multiobjective evolutionary algorithms.

Keywords: Multiple Objectives - Particle Swarm Optimization - Multiobjective Optimization

1 Introduction

The use of evolutionary algorithms for multiobjective optimization (an area called "evolutionary multiobjective optimization", or EMO for short) has significantly grown in the last few years, giving rise to a wide variety of algorithms [3]. As any other research area, EMO currently presents certain trends. One of them is to improve the efficiency both of the algorithms and of the data structures used to store nondominated vectors[1]. EMO researchers have produced some clever techniques to maintain diversity (e.g., the adaptive grid used by PAES [8]), new algorithms that use very small populations (e.g., the micro-GA [2]), and data structures that allow to handle unconstrained external archives (e.g., the dominated tree [5]).
Particle swarm optimization (PSO) is a relatively recent heuristic inspired by the choreography of a bird flock. PSO has been found to be successful in a wide variety of optimization tasks [6], but until recently it had not been extended to deal with multiple objectives.
PSO seems particularly suitable for multiobjective optimization mainly because of the high speed of convergence that the algorithm presents for single-objective optimization [6]. In this abstract, we present a proposal, called "multi-objective particle swarm optimization" (MOPSO), which allows the PSO algorithm to be able to deal with multiobjective optimization problems. Our current proposal is an improved version of the algorithm reported in [1]. The proposed approach is relatively simple to implement, it is population-based, it uses an external memory (called "repository") and a geographically-based approach to maintain diversity. This revised version also incorporates a mutation operator and a scheme based on dominance tournaments to handle constraints. MOPSO is validated using some standard test functions reported in the specialized literature and compared against two highly competitive EMO algorithms: the Nondominated Sorting Genetic Algorithm II [4] (NSGA II), and the micro-genetic algorithm for multiobjective optimization (microGA) [2].

[1] It's important to clarify that most EMO researchers consider important to generate as many Pareto optimal solutions as possible, and most of them are not concert with the further multicriteria decision making process required to reach a single response to the problem at hand. In this work, we are adopting this same assumption.

2 Description of the Approach

The algorithm of MOPSO is the following:

1. Initialize the population POP:
   (a) FOR $i = 0$ TO MAX /* MAX = number of particles */
   (b) Initialize $POP[i]$
2. Initialize the speed of each particle:
   (a) FOR $i = 0$ TO MAX
   (b) $VEL[i] = 0$
3. Evaluate each of the particles in POP.
4. Store the positions of the particles that represent nondominated vectors in the repository REP.
5. Generate hypercubes of the search space explored so far, and locate the particles using these hypercubes as a coordinate system where each particle's coordinates are defined according to the values of its objective functions.
6. Initialize the memory of each particle (this memory serves as a guide to travel through the search space. This memory is also stored in the repository):
   (a) FOR $i = 0$ TO MAX
   (b) $PBESTS[i] = POP[i]$
7. WHILE maximum number of cycles has not been reached DO
   (a) Compute the speed of each particle[2] using the following expression:

$$VEL[i] = W \times VEL[i] + R_1 \times (PBESTS[i] - POP[i]) + R_2 \times (REP[h] - POP[i])$$

   where W (inertia weight) takes a value of 0.4; R_1 and R_2 are random numbers in the range [0..1]; $PBESTS[i]$ is the best position that the particle i has had[3]; $REP[h]$ is a value that is taken from the repository; the index h is selected in the following way: those hypercubes containing more than one particle are assigned a fitness equal to the result of dividing any number $x > 1$ (we used $x = 10$ in our experiments) by the number of particles that they contain. Then, we apply roulette-wheel selection using these fitness values to select the hypercube from which we will take the corresponding particle. Once the hypercube has been selected, we select randomly a particle within such hypercube. $POP[i]$ is the current value of the particle i.
   (b) Compute the new positions of the particles adding the speed produced from the previous step:

$$POP[i] = POP[i] + VEL[i] \quad (1)$$

[2]Each particle has a dimensionality that can vary depending on the problem solved. When we say that we compute the speed of a particle, we refer to computing the speed for each of its dimensions.

[3]We will explain later on how do we define "better" in this context.

(c) Maintain the particles within the search space in case they go beyond its boundaries (avoid generating solutions that do not lie on valid search space).

(d) Evaluate each of the particles in POP.

(e) Update the contents of REP together with the geographical representation of the particles within the hypercubes. Since the size of the repository is limited, whenever it gets full, we apply a secondary criterion for retention: those particles located in less populated areas of objective space are given priority over those lying in highly populated regions.

(f) When the current position of the particle is better than the position contained in its memory, the particle's position is updated using:

$$PBESTS[i] = POP[i] \tag{2}$$

The criterion to decide what position from memory should be retained is simply to apply Pareto dominance (i.e., if the current position is dominated by the position in memory, then the position in memory is kept; otherwise, the current position replaces the one in memory; if neither of them is dominated by the other, then we select one of them randomly).

(g) Increment the loop counter

8. END WHILE

Additionally, we added a mutation operator that decreases its range of operation over time, such that at the beginning it covers the full range of each variable, and it deacreases according to a a nonlinear function. The number of individuals that are subject to mutation decreases over time, too. This operator was introduced to increase the exploratory power of PSO which we found to be limited in some functions in which there were important attractors in certain portions of the Pareto front. We also added a relatively simple scheme to handle constraints: Whenever two individuals are compared, we check their constraints. If both are feasible, nondominance is directly applied to decide who is the winner. If one is feasible and the other is infeasible, the feasible dominates. If both are infeasible, then the one with the lowest amount of constraint violation dominates the other.

3 Comparison of Results

Several test functions were taken from the specialized literature to compare our approach. However, only one is included in this abstract due to obvious space limitations. To compare our results in a quantitative way we used four criteria: average running time of the algorithm (using the same number of fitness function evaluations), the generational distance (as defined in [10]), the error ratio (as defined in [10]) and spacing (as defined in [9]). These metrics allow us to evaluate the behavior of a multiobjective optimization technique with respect to closeness to the true Pareto front (generational distance), number of nondominated individuals produced that belong to the true Pareto front of the problem (error ratio) and how uniform is the distribution of solutions along the Pareto front (spacing).

MOPSO was compared against two recent algorithms that are representative of the state of the art in evolutionary multiobjective optimization: the NSGA-II (using binary representation) [4], and the micro-genetic algorithm for multiobjective optimization [2]. In the following examples, the NSGA-II was run using a population size of 100, a crossover rate of 0.8, tournament selection, and a mutation rate of $1/L$, where L = chromosome length, and the maximum number of generations was set to 119. The micro-GA used a crossover rate of 0.9, an external memory of 100 individuals, a number of iterations to achieve nominal convergence of two, a population memory of 50 individuals, a percentage of non-replaceable memory of 0.05, a population size (for

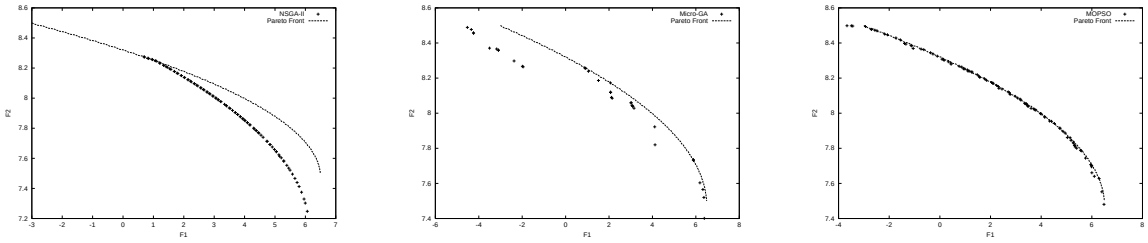

Figure 1: Pareto fronts produced by the NSGA II (left), the micro-GA (middle), and MOPSO for the test function

the micro-GA itself) of four individuals, and 25 subdivisions of the adaptive grid. The mutation rate was set to $1/L$ (L = length of the chromosomic string). MOPSO used a population of 100 particles, a repository size of 100 particles and 30 divisions for the adaptive grid. Our implementation uses a real-numbers representation and it is therefore intended for continuous search spaces. Note, however, that PSO can also be used with binary representation (see [6] for details).

3.1 Test Function

We used the following problem proposed by Kita [7]: Maximize $F = (f_1(x,y), f_2(x,y))$, where

$$f_1(x,y) = -x^2 + y, \quad f_2(x,y) = \frac{1}{2}x + y + 1$$

subject to:

$$0 \geq \frac{1}{6}x + y - \frac{13}{2}, \quad 0 \geq \frac{1}{2}x + y - \frac{15}{2}, \quad 0 \geq 5x + y - 30$$

and: $x, y \geq 0$.

Figure 1 shows the graphical results produced by the NSGA-II, the micro-GA and our MOPSO in the test function previously indicated. The true Pareto front of the problem is also shown. The values for the metrics for MOPSO were the following: Generational Distance: average = 0.019479, best =0.002562, worst = 0.081061, standard deviation = 0.018906; Error ratio: average = 0.568231, best = 0.43, worst = 0.67, standard deviation = 0.063414; Spacing: average = 0.142568, best = 0.051923, worst = 0.753783, standard deviation = 0.160711. The values for the metrics for the NSGA-II were the following: Generational Distance: average = 0.084239, best =0.003885, worst = 0.678449, standard deviation = 0.165244; Error ratio: average = 0.8965, best = 0.75, worst = 0.99, standard deviation = 0.067143; Spacing: average = 0.098486, best = 0.001032, worst = 1.48868, standard deviation = 0.32738. The values for the metrics for the micro-GA were the following: Generational Distance: average = 0.150763, best =0.00513, worst = 0.912065, standard deviation = 0.216558; Error ratio: average = 0.927706, best = 0.734694, worst = 1.01639, standard deviation = 0.068739; Spacing: average = 0.31502, best = 0.06561, worst = 1.64386, standard deviation = 0.421742.

These results are representative of the performance of MOPSO which defeated the two other algorithms with which it was compared with respect to all the metrics adopted. It also provided the lowest standard deviations overall.

4 Conclusions and Future Work

This abstract presents a multiobjective optimization technique based on the particle swarm optimization algorithm. The proposed algorithm seems very promising since it is capable of outperforming other techniques that represent the state of the art in evolutionary multiobjective optimization.

A sensitivity analysis of the technique is currently under way, so that we can determine the role of each of the parameters used in the performance of the algorithm (particularly those controlling the flight direction of the particles). We are also interested in using spatial data structures to store nondominated individuals and in incorporating a crowding operator that can improve the distribution of solutions produced by the algorithm. Both tasks are part of our current research work.

Acknowledgements

The first author gratefully acknowledges support from CONACyT through project 34201-A. The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Section of the Electrical Engineering Department at CINVESTAV-IPN.

References

[1] Carlos A. Coello Coello and Maximino Salazar Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 1051–1056, Piscataway, New Jersey, May 2002. IEEE Service Center.

[2] Carlos A. Coello Coello and Gregorio Toscano Pulido. Multiobjective Optimization using a Micro-Genetic Algorithm. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[3] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, Boston, 2002. ISBN 0-3064-6762-3.

[4] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[5] Richard M. Everson, Jonathan E. Fieldsend, and Sameer Singh. Full Elite Sets for Multi-Objective Optimisation. In I.C. Parmee, editor, *Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture (ACDM 2002)*, volume 5, pages 343–354, University of Exeter, Devon, UK, April 2002. Springer-Verlag.

[6] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.

[7] Hajime Kita, Yasuyuki Yabumoto, Naoki Mori, and Yoshikazu Nishikawa. Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature—PPSN IV*, Lecture Notes in Computer Science, pages 504–512. Springer-Verlag, Berlin, Germany, September 1996.

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

[9] J. R. Schott. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.

[10] David A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.