Particle Swarm Optimization to Solve Optimization Problems

Gregorio Toscano-Pulido and Carlos A. Coello Coello

Evolutionary Computation Group at CINVESTAV-IPN (EVOCINV)
Electrical Eng. Department, Computer Science Dept.
Av. IPN No. 2508 Col. San Pedro Zacatenco, México D.F. 07300, MÉXICO
gtoscano@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

Abstract. Particle swarm optimization (PSO) is a relatively recent biologically-inspired heuristic that has been found to be very successful in a wide variety of optimization tasks. Our work addresses two main issues that have been scarcely dealt with in the specialized literature: constraint-handling (for global optimization) and multiobjective optimization. Our results, summarized in this paper, indicate the high viability of PSO for both single and multiobjective optimization.

1 Introduction

Kennedy & Eberhart [8] proposed an approach called "particle swarm optimization" (PSO) which was inspired on the choreography of a bird flock. The approach can be seen as a distributed behavioral algorithm that performs (in its more general version) multidimensional search. In the simulation, the behavior of each individual is affected by either the best local (i.e., within a certain neighborhood) or the best global individual. The approach uses then the concept of population and a measure of performance similar to the fitness value used with evolutionary algorithms. Also, the adjustments of individuals are analogous to the use of a crossover operator. The approach also introduces the use of flying potential solutions through hyperspace (used to accelerate convergence) which can be seen as a mutation operator. It is worth noticing that when the research reported herein started, there were no attempts to extend PSO to handle multiple objectives and there was very limited work regarding constraint-handling. These are precisely the two main issues addressed in this work.

1.1 Handling Constraints with PSO

The first problem of interest to us was the general nonlinear programming problem in which we want to:

$$\text{Find } \boldsymbol{x} \text{ which optimizes } f(\boldsymbol{x}) \tag{1}$$

subject to:

$$g_i(\boldsymbol{x}) \leq 0, \ \ i = 1, \ldots, n \tag{2}$$

$$h_j(\boldsymbol{x}) = 0, \ \ j = 1, \ldots, p \tag{3}$$

where $\boldsymbol{x}$ is the vector of solutions $\boldsymbol{x} = [x_1, x_2, \ldots, x_r]^T$, n is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or nonlinear).

PSO, like other evolutionary algorithms, lacks an explicit mechanism to incorporate constraints. Remarkably, there has been very little work related to the incorporation of constraints into PSO, despite the fact that most real-world applications have constraints. The main goal of this initial research was to develop a relatively simple mechanism to incorporate constraints into PSO, as to make this algorithm competitive with the state-of-the-art approaches in the area [1].

1.2 Our Proposed Approach

For computing the velocity of a particle, we used the expression proposed in [12]:

$$V_{id} = w \times V_{id} + c_1 \times rand_1() \times (p_{best,id} - x_{id}) + c_2 \times rand_2() \times (g_{best,id} - x_{id})$$

where V_{id} is the velocity of the id dimension, c_1 and c_2 are two values randomly generated in the range $[1.5, 2.5]$ (this range was empirically derived), $rand_1()$ and $rand_2()$ refer to functions that return a random value within the range $[0.0, 1.0]$, w is the inertia weight, which in our case takes a value randomly generated within the range $[0.1, 0.5]$, p_{best} is the best position of the current particle found so far and g_{best} is the best position of the best particle found so far.

The constraint-handling mechanism proposed by us consists of two main components: a turbulence operator (i.e., some form of mutation intended to improve the exploratory capabilities of PSO), and a simple decision-making scheme based on closeness to the feasible region to choose a leader when dealing with constrained search spaces. The **turbulence** consists of an alteration to the flight velocity of a particle.[1] This modification is performed in all the dimensions (i.e., in all the decision variables), such that the particle can move to a completely isolated region. This mechanism aims to perturb the swarm as to avoid that the particles get trapped in local optima. The turbulence operator acts based on a probability that considers the current generation and the total number of iterations to be performed. The idea is to have a much higher probability to perturb the flight of the particles at the beginning of the search. Over time, this probability will be decreased as we progress in the search. The mechanism to handle constraints that we proposed is applied when selecting a leader. What we did was to perform a small change in the fitness function such that if we compare two feasible particles, the particle that has the highest fitness value wins. If one of the particles is infeasible and the other one is feasible, then the feasible particle wins. If both particles compared are infeasible, then the particle that has the lowest value in its total violation of constraints (normalized with respect to the largest violation of each constraint achieved by any particle in the current population) wins. The idea is to choose as a leader to the particle that, even when infeasible, lies closer to the feasible region.

To evaluate the performance of the proposed approach we used the 13 test functions described in [11], and we compared our results with respect to three constraint-handling

[1] This mechanism is inspired on [6].

techniques that are representative of the state-of-the-art in the area: Stochastic Ranking (SR) [11], the Homomorphous Maps (HM) [10], and the Adaptive Segregational Constraint Handling Evolutionary Algorithm (ASCHEA) [7]. Due to space limitations, in Table 1 we only present a comparison of our results against Stochastic Ranking, which is the most competitive of the approaches previously indicated. The results obtained by our approach are highly competitive, and in some cases, even improve on the results obtained by much more elaborate approaches such as the homomorphous maps [10].

		Best Result		Mean Result		Worst Result	
Problem	Optimal	PSO	SR	PSO	SR	PSO	SR
g01	-15.000000	-15.000000	-15.000000	-15.000000	-15.000000	-15.000000	-15.000000
g02	0.803619	0.803432	0.803515	0.790406	0.781975	0.750393	0.726288
g03	1.000000	1.004720	1.000000	1.003814	1.000000	1.002490	1.000000
g04	-30665.539000	-30665.500000	-30665.539000	-30665.500000	-30665.539000	-30665.500000	-30665.539000
g05	5126.498000	5126.640000	5126.497000	5461.081333	5128.881000	6104.750000	5142.472000
g06	-6961.814000	-6961.810000	-6961.814000	-6961.810000	-6875.940000	-6961.810000	-6350.262000
g07	24.306000	24.351100	24.307000	25.355771	24.374000	27.316800	24.642000
g08	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g09	680.630000	680.638000	680.630000	680.852393	680.656000	681.553000	680.763000
g10	7049.330700	7057.590000	7054.316000	7560.047857	7559.192000	8104.310000	8835.655000
g11	0.750000	0.749999	0.750000	0.750107	0.750000	0.752885	0.750000
g12	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
g13	0.053950	0.068665	0.053957	1.716426	0.057006	13.669500	0.216915

Table 1. Comparison of results of our PSO vs. Stochastic Ranking (SR) [11].

2 Multiobjective optimization

The second problem of interest to us is the general multiobjective optimization problem in which we want to find the vector $\boldsymbol{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ which will satisfy the m inequality constraints:

$$g_i(\boldsymbol{x}) \leq 0 \quad i = 1, 2, \ldots, m \tag{4}$$

the p equality constraints

$$h_i(\boldsymbol{x}) = 0 \quad i = 1, 2, \ldots, p \tag{5}$$

and will optimize the vector function

$$\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})]^T \tag{6}$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables. In other words, we wish to determine from among the set $\mathcal{F}$ of all numbers which satisfy (4) and (5) the particular set $x_1^*, x_2^*, \ldots, x_n^*$ which yields the optimum values of all the k objective functions of the problem. The following are some basic definitions related to multiobjective optimization.

Definition 1 (Pareto Dominance): *A vector $\boldsymbol{u} = (u_1, \ldots, u_k)$ is said to dominate $\boldsymbol{v} = (v_1, \ldots, v_k)$ (denoted by $\boldsymbol{u} \preceq \boldsymbol{v}$) if and only if u is partially less than v, i.e., $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$.* $\square$

Definition 2 (Pareto Optimal Set): *For a given MOP $\boldsymbol{f}(x)$, the Pareto optimal set ($\mathcal{P}^*$) is defined as:*

$$\mathcal{P}^* := \{x \in \mathcal{F} \mid \neg\exists\, x' \in \mathcal{F} \;\; \boldsymbol{f}(x') \preceq \boldsymbol{f}(x)\}. \tag{7}$$

□

Definition 3 (Pareto Front:): *For a given MOP $\boldsymbol{f}(x)$ and Pareto optimal set $\mathcal{P}^*$, the Pareto front ($\mathcal{PF}^*$) is defined as:*

$$\mathcal{PF}^* := \{\boldsymbol{u} = \boldsymbol{f} = (f_1(x), \ldots, f_k(x)) \mid x \in \mathcal{P}^*\}. \tag{8}$$

□

2.1 Our Proposed Approach

Particle swarm optimization seems particularly suitable for multiobjective optimization mainly because of the high speed of convergence that the algorithm presents for single-objective optimization [8]. The main goal of our research in this regard was to design mechanisms to extend PSO such that it could generate reasonably good Pareto fronts of difficult test functions (both constrained and unconstrained).

Our approach is based on the use of Pareto ranking and a subdivision of decision variable space into several sub-swarms (this is done using clustering techniques). Since independent PSOs are run into each swarm, our approach can be seen as a meta-MOPSO algorithm. After a certain (pre-defined) number of iterations, the leaders of each swarm are migrated to a different swarm in order to variate the selection pressure. This sort of scheme is a novel proposal to solve multiobjective optimization problems using PSO. Also, note that this algorithm does not use an external population (as other recent proposals [2, 6]), since elitism in this case is an emergent process derived from the migration of leaders.

The complete execution process of our algorithm can be divided in three stages: initialization, flight and generation of results. At the first stage, every swarm is initialized. Each swarm creates and initializes its own particles and generates the leaders set among the particle swarm set by using Pareto ranking. In the second stage is where the algorithm performs its strongest effort. First, it performs the execution of the flight of every swarm; next, it applies a clustering algorithm to group the guide particles. This is performed until reaching a total of $GMax$ iterations. The execution of the flight of each swarm can be seen as an entire PSO process (with the difference that it will only optimize an specific region of the search space). First, each particle will select a leader to which it will follow. At the same time, each particle will try to outperform its leader and to update its position. If the updated particle is not dominated by any member of the leaders set, then it will become a new leader. The execution of the swarm will start again until a total of $sgmax$ iterations is reached. Constraints are handled in this algorithm when checking Pareto dominance. When we compare two individuals, we first check their feasibility. If one is feasible and the other is infeasible, the feasible individual wins. If both are infeasible, then the individual with the lowest amount of (total) constraint violation wins. If they both have the same amount of constraint violation (or if they are both feasible), then the comparison is done using Pareto dominance. Once all the swarms have finished theirs flights, a clustering algorithm takes the control by grouping the closest particle guides into n_{swarms} swarms. These particle guides will try to outperform each swarm in the next iteration. This is mainly done by grouping the leaders of all the swarms into a single set, and then splitting this set among n_{swarms} groups (clustering is done with respect to closeness in decision variable space). Each resulting group will be assigned to a different swarm. The third and final stage will present the results, i.e. it will report all the nondominated solutions found.

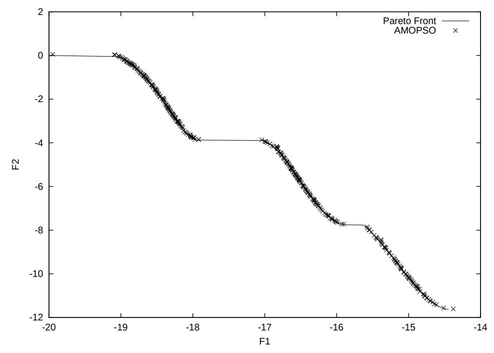

Fig. 1. Comparison of results for one example. The true Pareto front is shown as a continuous line (note that the horizontal segments are NOT part of the Pareto front and are shown only to facilitate drawing the front) and the Pareto front found by our approach is shown as crosses.

This algorithm was validated using several test functions taken from the specialized literature. Results were compared against the micro-GA for multiobjective optimization [3], the Nondominated Sorting Genetic Algorithm II (NSGA II) [5], the Pareto Archived Evolution Strategy (PAES) [9], and the Multiobjective Particle Swarm Optimization (MOPSO) [2]. Besides graphical comparisons, we adopted three metrics to compare our results [4]. Figure 1 shows a sample result for one of the problems used. In general, our algorithm had a good convergence rate to the true Pareto front, and the results indicate that our approach is a viable alternative since it outperformed some of the best multiobjective evolutionary algorithms known to date in several test functions (both constrained and unconstrained), which were not included due to space limitations.

3 Conclusions and Future Work

Our explorations of particle swarm optimization have been very fruitful so far. We have produced two highly competitive approaches: one for constrained single-objective optimization and another one for multiobjective optimization problems. Our current work focuses on performing a theoretical study about the convergence properties of the first algorithm. One aspect that we would like to explore in the future is the study of alternative mechanisms to handle constraints through the use of infeasible solutions that can act as leaders in a special swarm. We believe that this sort of mechanism could improve the performance of both approaches (i.e., the one for single-objective optimization and the one for multiobjective optimization), particularly when dealing with problems in which the global optimum lies on the boundary between the feasible and infeasible regions. We also intend to study alternative mechanisms to accelerate convergence while keeping the same quality of the results achieved in this paper. Such type of approach may be particularly useful for real-world applications. Finally, we also want to perform an analysis of the impact of the mechanism adopted to select leaders on the performance of the approach.

References

1. Carlos A. Coello Coello. Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002.
2. 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.
3. Carlos A. Coello Coello and Gregorio Toscano Pulido. Multiobjective Optimization using a Micro-Genetic Algorithm. In Lee Spector et al., editor, *Proceedings of GECCO'2001*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.
4. 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.
5. 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.
6. Jonathan E. Fieldsend and Sameer Singh. A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence. In *Proceedings of the 2002 U.K. Workshop on Computational Intelligence*, pages 37–44, Birmingham, UK, 2002.
7. Sana Ben Hamida and Marc Schoenauer. ASCHEA: New Results Using Adaptive Segregational Constraint Handling. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 1, pages 884–889, Piscataway, New Jersey, May 2002. IEEE Service Center.
8. James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.
9. Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
10. Slawomir Koziel and Zbigniew Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.
11. Thomas P. Runarsson and Xin Yao. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.
12. Yuhui Shi and Russell C. Eberhart. Parameter Selection in Particle Swarm Optimization. In V. W. Porto et al., editor, *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pages 591–600. Springer-Verlag, March 1998.