Global Numerical Optimization with a bi-population Particle Swarm Optimizer

Susana C. Esquivel and Leticia C. Cagnina
LIDIC (Research Group) - Universidad Nacional de San Luis
Ej. de Los Andes 950, (D5700HHW) San Luis, ARGENTINA
E-mails: {esquivel,lcagnina}@unsl.edu.ar

Abstract

This paper presents an enhanced Particle Swarm Optimizer approach, which is designed to solve numerical unconstrained optimization problems. The approach incorporates a dual population in an attempt to overcome the problem of premature convergence to local optima. The proposed algorithm is validated using standard test functions (unimodal, multi-modal, separable and nonseparable) taken from the specialized literature. The results are compared with values obtained by an algorithm representative of the state-of-the-art in the area. Our preliminary results indicate that our proposed approach is a competitive alternative to solve global optimization problems.

Keywords: Particle Swarm Optimizer, Optimization, Unconstrained Functions

Resumen

Este artículo presenta un nuevo algoritmo Particle Swarm Optimizer, diseñado para resolver problemas de optimización numéricos sin restricciones, que incorpora una población dual para intentar solucionar el problema de convergencia prematura en óptimos locales. El algoritmo propuesto es validado usando funciones de prueba estandard (unimodales, multi-modales, separables y no separables) tomadas de la literatura especializada. Los resultados son comparados con los valores obtenidos por un algoritmo representativo del estado del arte en el área. Los resultados preliminares indican que la propuesta es una alternativa competitiva para resolver problemas de optimización global.

Palabras claves: Particle Swarm Optimizer, Optimización, Funciones sin Restricciones

1 INTRODUCTION

In the last years metaheuristics (particularly, evolutionary algorithms) has been used to solve real-world applications. One of that metaheuristics that has been adopted to solve such problems is Particle Swarm Optimizer (PSO) [14].
PSO was conceived as a simulation of individual and social behavior [13] such as the one observed in flocks of birds and fish. PSO explores the search space using a population of individuals, and the best performers (either within a group or with respect to the entire population) affect the performance of the others. Each individual is named *particle* and represents a possible solution within a multidimensional search space. The particles have their own position and velocity, which are constantly updated. They record their past behavior and use it to move towards promising regions of the search space.
PSO has been found to be highly competitive for solving unconstrained real-world optimization problems [15, 16, 9, 3, 4]. However, on strongly multi-modal test functions, PSO tends to suffer premature convergence. This happens because the diversity decreases, leading to the stagnation of the swarm.

In this paper, we present a PSO algorithm which is designed to solve unconstrained optimization problems. Our approach contains a different mechanism to update the velocity and position of the particles [2], which is extended by adding to it a bi-population as a way to avoid premature convergence.

The remainder of this paper is organized as follows. Section 2 provides the statement of general unconstrained optimization problems. Section 3 briefly discusses the previous related work. In Section 4, we describe in detail our proposed approach. Section 5 describes the experimental setup and provides an analysis of the results obtained from our empirical study. Section 6 shows the statistical analysis of results between the algorithms presented. The conclusions and some directions for future research are stated in Section 7.

2 STATEMENT OF THE PROBLEM

The unconstrained optimization involves maximization and minimization problems, although maximization ones can be transformed into minimization ones as:

$$f'(\vec{x}) = -f(\vec{x}) \tag{1}$$

So, we treated just with minimization class of problem in this paper.
Without loss of generality, we can consider the general nonlinear optimization problem as a minimization problem, which can be formally stated as the problem of finding $\vec{x}$ which:

$$minf(\vec{x}) \text{ with } \vec{x} = (x_1, x_2, \ldots, x_D) \in \mathcal{S} \subseteq \mathcal{R}^D \tag{2}$$

Each $x_d \in [l_d, u_d]$ with $d \in [1..D]$. The l_d and u_d are the lower and upper bounds imposed on the decision variables. $\mathcal{S}$ (the search space) is a $D-$dimensional rectangle defined by the lower and upper bounds of each variable x_d.

3 PREVIOUS RELATED WORK

As indicated before, despite its success in a variety of optimization problems, PSO has been applied to solve just some unconstraint multi-modal function, but no much comparative studies with another

methods are presented. Next, we will review the most representative work done in this area.

Riget et al. [19] presented an *attractive* and *repulsive* PSO (ARPSO) specially designed to overcome the problem of premature convergence. The algorithm uses a measure to control the diversity in the swarm. The performance of ARPSO was compared with a basic PSO and a genetic algorithm using 4 multi-modal functions. The conclusions are that their algorithm performs extremely well for the problems tested.

In [10], Jian discusses the parameters on PSO. A velocity and position disturbance mechanism is introduced to prevent premature convergence. The algorithm is tested using 2 unimodal function and 2 multi-modal. The simulation experiments showed that the improved PSO has good performance.

Voss [20] introduced a PCPSO procedure to create a *flying and dynamic* coordinate system with the particles. The author compared the methodology of the algorithm as a symbiotic relationship between swarm movement and the rotation and dimension of the life space. The proposed PCPSO is validated with 4 function: 2 unimodal and 2 multi-modal. The algorithm performed well for the 30-dimensional test functions tested.

Oltean et al. [5] evolved the structure of a Particle Swarm Optimization with a new model. The model is a hybrid method that combines a basic PSO and a genetic algorithm. The genetic algorithm chromosomes are vectors with the indexes of the particles. Those chromosomes are used to select which particles will be update. The authors make some experiments with 6 unimodal and 4 multi-modal functions. The results are compared with the previous version PSO standard and the evolved PSO. The conclusions are that sometimes the new version performs similarly and sometimes better.

Liang et al. [17] showed a variant of PSO which uses a novel learning strategy whereby all other particles' best is used to update the velocity of particles in the swarm. The strategy helps to preserve diversity and in that way avoid premature convergence. The paper studies the algorithm performance using 2 unimodal functions, 6 multi-modal unrotated, 2 noncontinuos Rastrigin's function and 6 rotated multi-modal problems. The results showed that the proposed algorithm improves significantly the performance of PSO algorithms.

In [7] the authors proposed the use of two simple PSO. Both (global and local models) were hybridized with a nonuniform mutation operator taken from the evolutionary algorithms literature. The algorithms were tested with 4 multi-modal functions and compared the results with those obtained with 6 PSO models studied by Peer et al. The performance of their algorithms indicate that the proposed PSO models are highly competitive.

4 BI-PSO ALGORITHM

In this section, we describe in detail our proposed approach, which we call the Bi-PSO algorithm.

4.1 General Model

As stated before, a PSO algorithm operates on a population of particles. Due to the type of problem to optimize (with n decision variables), the particles are n-dimensional real number vectors. The best position found so far for the particles (for the *gbest* model) or in the neighborhood (*lbest* model) is

recorded. The best value reached by each particle (*pbest*) is stored, too. As in the basic model, the particles evolve using two update formulas, one for position and another one for velocity.

4.2 Our Approach

As it was stated in some of our previous work [1], the *gbest* model tends to converge to a local optimum although works well in many problems. Motivated by this, we proposed a formula to update the velocity, using a combination of both the *gbest* and the *lbest* models [2], it is shown in equation (3).

$$v_{id} = w(v_{id} + \gamma_1(p_{id} - par_{id}) + \gamma_2(p_{ld} - par_{id}) + \gamma_3(p_{gd} - par_{id})) \tag{3}$$

where v_{id} is the velocity of particle i at the dimension d; w is the inertia factor [6] whose goal is to balance global exploration and local exploitation. γ_1 is the personal learning factor, and γ_2 and γ_3 are the social learning factors. These 3 values are multiplied by 3 different random numbers within the range [0..1], p_{id} is the best position reached by the particle i; p_{ld} is the best position reached by any particle in the neighborhood, p_{gd} is the best position reached by any particle in the swarm. par_{id} is the value of the particle i at the dimension d.

The equation for updating the particles also was modified as we proposed in a previous work [2]. In that paper, during 10% of the iterations, we applied the normal formula (depicted in equation (4)) as suggested in [14].

$$par_{id} = par_{id} + v_{id} \tag{4}$$

And in the remainder 90% of cases, we used equation (5) proposed by Kennedy [12].

$$par_i = N\left(\frac{p_i + p_l}{2}, |p_i - p_l|\right) \tag{5}$$

where p_i is the position of the particle to be updated, N is the Gaussian random generator, p_i and p_l are the best position reached by the particle par_i and the best position reached by any particle in the neighborhood of par_i, respectively. That probability was empirically found to be the best after performing a series of experiments with all the test functions evaluated.

We use a circle topology [11] to compute the p_{ld} value, in which each particle is connected to k neighbors. The neighbors are determined by the position of the particles in the storage structure. Figure 1 illustrates this concept.

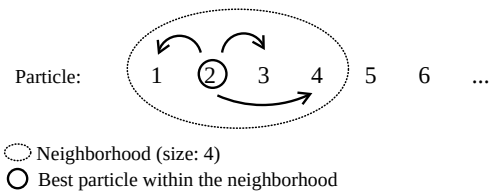


Figure 1: Circle topology adopted in the population of our PSO approach

To help avoiding convergence to a local optimum, we used a dynamic mutation operator [1] which is applied to each individual with a pm-probability. This value is calculated considering the total number of iterations in the algorithm (*cycles*) and the current cycle number as the following equation indicates:

$$pm = max\_pm - \frac{max\_pm - min\_pm}{max\_cycle} * current\_cycle \quad (6)$$

where max_pm and min_pm are the maximum and minimum values that pm can take, max_cycle is the total number of cycles that the algorithm will iterate, and $current\_cycle$ is the current cycle in the iterative process. The probability pm is high in the early cycles and we decrease that value on every iteration. That helps to explore at the starts in the search process and to explode at the ends.

4.3 Bi-population

The Bi-PSO algorithm splits the entire population into two subpopulations each of which is independently evolved. The idea is to maintain more than one group of particles exploring the search space (at the same time). In that way the possibility of falling into local optima is reduced.

One may then wonder why to adopt only two subpopulations and not more. The reason is that it does not make any sense to adopt more than two subpopulations, considering the small number of particles that we use in our original population (only 10). In fact, we believe that our neighborhood topology would not work properly if we adopt less than 5 particles and therefore our choice of adopting only two subpopulations.

All the features stated before for the entire population (neighborhoods, *lbest* and *gbest* approaches, equations for updating the velocity and the positions) still apply, but in this case, they are applied not to a single population, but to each subpopulation. When the iterative process finishes, the best particle from both subpopulations is reported as the final output.

4.4 Bi-PSO Pseudocode

Figure 2 shows the pseudocode of our proposed Bi-PSO algorithm. At the beginning of the search, we initialize the vectors of position and velocity of each particle in both subpopulations (lines 2 to 5). After evaluating the particles and obtaining the best values: *pbest*, *lbest* and *gbest* (lines 6 and 7), the subpopulations begin to evolve. During the evolutionary process, new values of *pbest*, *lbest* and *gbest* are chosen and both, the velocity and the position of each particle are updated (lines 8 to 24). At line 25, a *keeping* mechanism is applied to control that all the dimensions in all particles are within the allowable bounds. The mutation probability is updated and the particles are mutated, if applicable (lines 26 and 27). After that, the particles are evaluated and new "best" values are recorded (lines 28 to 30). Finally, the best value reached by any subpopulation is taken and compared. The best of them is returned (lines 31 and 32).

5 EXPERIMENTAL STUDY

For validating our proposed approach, we adopted 13 minimization test problems with 30 variables taken from [21]:

- f_1: **Sphere Model**. Unimodal and separable function.
- f_2: **Schwefel's Problem 2.22**. Unimodal and separable function.
- f_3: **Schwefel's Problem 1.2**. Unimodal and nonseparable.
- f_4: **Schwefel's Problem 2.21**. Unimodal and separable function.

```
0.  Bi-PSO:
1.  Swarm Initialization
2.    Initialize subpop1
3.    Initialize velocity for subpop1
4.    Initialize subpop2
5.    Initialize velocity for subpop2
6.    Evaluate fitness for each subpop
7.    Record pbest and gbest for each subpop
8.  Swarm flights through the search space
9.    DO
10.     FOR each subpop DO
11.         FOR i=1 TO numberOfparticles DO
12.             Search the best leader in the
13.                neighborhood of part_i
14.                and record in lbest_i
15.             FOR j=1 TO numberOfdimensions DO
16.                 Update vel_ij
17.                 IF flip(0.1)
18.                    Update part_ij with eq.(3)
19.                 ELSE
20.                    Gaussian update with eq.(5)
21.                 END
22.             END
23.         END
24.     END
25.     Keeping particles
26.     Update pm
27.     Mutate every particle depending on pm
28.     Evaluate fitness(part_i)
29.     Record pbest and gbest
30.   WHILE(current_cycle < max_cycle)
31.   result=BEST(best_subpop1,best_subpop2)
32.   RETURN(result)
```

Figure 2: Pseudo-code of Bi-PSO

- f_5: **Generalized Rosenbrock's Function**. Multi-modal and nonseparable problem.
- f_6: **Step Function**. Unimodal and separable problem.
- f_7: **Quartic Function with Noise**. Unimodal and separable problem.
- f_8: **Generalized Schwefel's Problem 2.26**. Multi-modal and separable function, with many local minima. The number of local minima increases exponentially as the function dimension increases.
- f_9: **Generalized Rastrigin's Function**. Multi-modal and separable problem, with many local minima. The number of local minima increases exponentially as the function dimension increases.
- f_{10}: **Ackley's Function**. Multi-modal and nonseparable problem, with many local minima. The number of local minima increases exponentially as the function dimension increases.
- f_{11}: **Generalized Griewank Function**. Multi-modal and nonseparable problem, with many local minima. The number of local minima increases exponentially as the function dimension increases.

- f_{12} and f_{13}: **Generalized Penalized Functions**. Multi-modal and nonseparable problems, with many local minima. The number of local minima increases exponentially as the function dimension increases.

The detailed description of the test problems may be consulted in [21]. We performed 50 independent runs per problem, with a total of 120,000 evaluations of objective function per run. Our proposed Bi-PSO used the following parameters: swarm size = 10 particles, $pm\_min = 0.1$, $pm\_max = 0.4$, neighborhood size = 3, inertia factor w = 0.8, personal learning factor and social learning factors for γ_1, γ_2 and γ_3 was set to 1.8. The parameter settings such as swarm size, mutation probability, neighborhood size and learning factors were empirically derived after numerous experiments.

Our results were compared with respect to those obtained by a Differential Evolution (DE) algorithm proposed by Velázquez [18]. DE is an Evolutionary Algorithm created to solve optimization problems, mainly in continuous search spaces. It is well known the high performance of DE in the most problems tested, compared with other optimization techniques. DE performs mutation based on the distribution of solutions in the population of individuals, so search directions depend on individuals' location selected to calculate the mutation values.
In [18] a new approach is presented to increase the probability of each parent to generate a better offspring. For that the author allows each solution to generate more than one offspring using a different operator which combines the information of the best solution and of the current parent to find new search directions. The new model uses a binomial discrete recombination operator, too. The parameter setting may be consulted at [18], but it is important remark that the DE performed 50 independent runs with a total of 120.000 evaluations (the same that with Bi-PSO) as the author suggest in his work. DE approach is highly competitive and is representative of the state-of-the-art in the area, for that we use it to make the comparison of performance.
Table 1 shows the best values found by Bi-PSO and DE.

Table 1: Comparison of the **best** values obtained by our Bi-PSO and the Differential Evolution (DE).

Funct.	Benchmark	Bi-PSO	DE
f1	0.00000	0.00000	0.00000
f2	0.00000	0.00000	0.00000
f3	0.00000	3.50228	**0.00000**
f4	0.00000	0.11688	**0.00000**
f5	0.00000	0.**00010**	0.04031
f6	0.00000	0.00000	0.00000
f7	0.00000	0.00000	0.00000
f8	-12569.48661	-12569.48661	-12569.48661
f9	0.00000	0.00000	0.00000
f10	0.00000	0.00000	0.00000
f11	0.00000	0.00000	0.00000
f12	0.00000	0.00000	0.00000
f13	0.00000	0.00000	0.00000

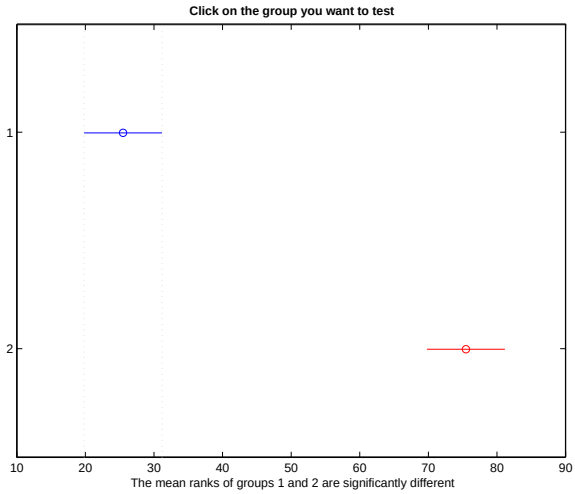
Table 2 shows the Mean values obtained by Bi-PSO and DE for the 13 unconstrained test functions adopted in our empirical study.

Table 2: Best Values obtained with **Bi-PSO** and **DE**.

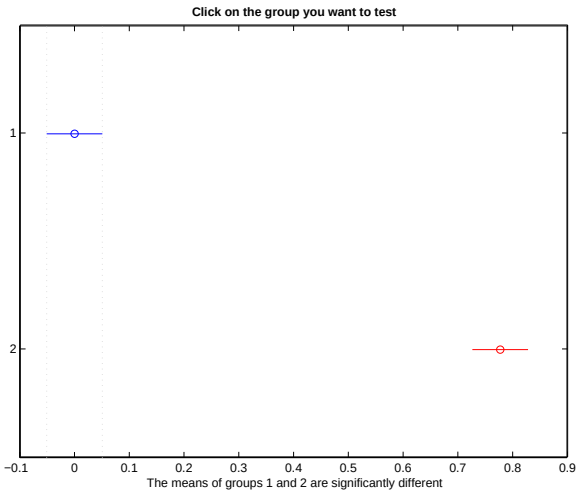
Funct.	Benchmark	Mean DE	Mean Bi-PSO
f1	0.00000	0.00000	0.00000
f2	0.00000	0.00000	0.00000
f3	0.00000	**0.00000**	178.54809
f4	0.00000	**0.00000**	0.77809
f5	0.00000	**2.71842**	28.28478
f6	0.00000	0.00000	0.00000
f7	0.00000	0.00000	0.00000
f8	-12569.48661	-12550.53648	**-12569.48632**
f9	0.00000	**0.23879**	0.84162
f10	0.00000	0.00000	0.00000
f11	0.00000	0.00000	0.00000
f12	0.00000	0.00000	0.00000
f13	0.00000	0.00000	0.00000

6 STATISTICAL ANALYSIS

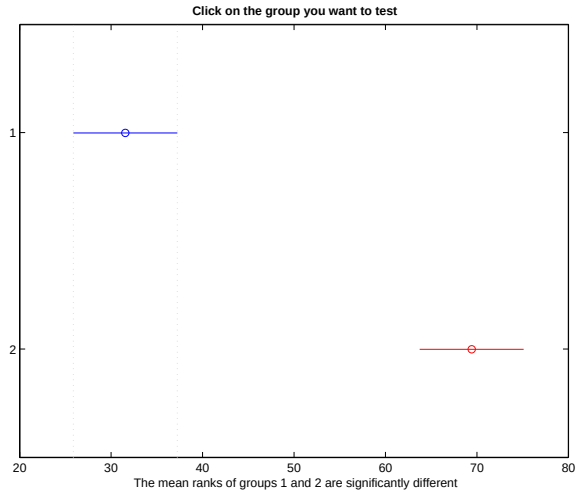
To analize the performance of our algorithm we used a statistical test. We do an analysis of variance between DE and Bi-PSO using the best values of the 50 independent runs we did with each one. We apply the Kruskal-Wallis [8] nonparametric one-way analysis because the values (the sample) do not have a normal distribution (determined with the Kolmogorov-Smirnov test).
The Kruskal-Wallis test returns the *p-value* for the null hypothesis for all samples. If the *p-value* is zero or near, that suggests that at least one sample is significantly different (or *statistically significant*) than the other samples. Usually, if *p-value* is less than 0.05, we declare the results are significant.
Table 3 shows the *p-value* for each function. The results indicate the values reached with Bi-PSO (for all function except: f3, f4, f5, f8 and f9), are not statistically significant from those of DE. That indicates DE and Bi-PSO are comparable with respect a performance. As we know, DE is a higher performance approach even its complex process to compute the solutions. Bi-PSO is simpler and has the similar performance in 8 functions.
For f3, f4, f5, f8 and f9, the values reached by DE are statistically different (table 3) from those obtained with our algorithm. For functions f3 and f4 DE obtains the best results as we can observe in table 1. However, for function f5 the best value is reached by our algorithm (table 1). For functions f8 and f9, we do not observe differences between the bests obtained by every algorithm (table 1), but we observe the differences in table 2 in which the best mean value for f8 is obtained for Bi-PSO and for function f9 is obtained by DE.
For those functions significantly different we apply the Tukey test to determine between which experimental conditions the differences are significant. The test returns: an estimate value (EV) into a range ([LI,LS]). If the range does not contain the zero-value, then the results are confirmed (significantly different). As we observe in table 3, for all function statistically significant the ranges do not have the zero-value. The subfigures in figure 3 confirm the last (algorithm 1 is DE and algorithm 2 is Bi-PSO, on the vertical axis).



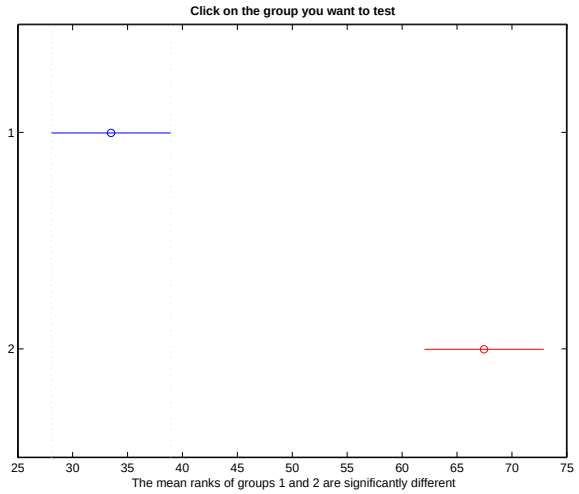
(a) f3



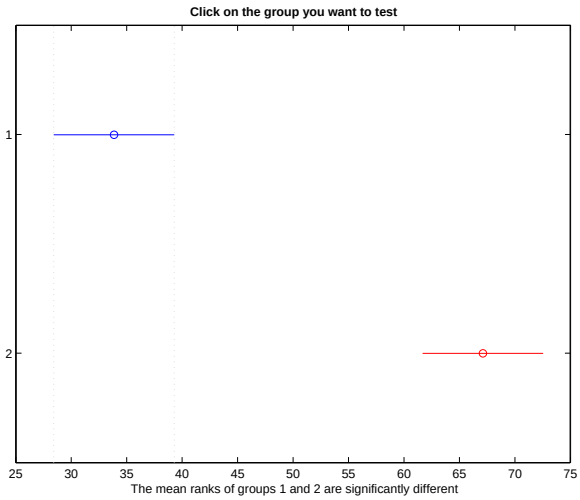
(b) f4



(c) f5



(d) f8



(e) f9

Figure 3: DE vs Bi-PSO Tukey's test

Table 3: Kruskal-Wallis' p-values y Tukey's results for DE vs Bi-PSO

Funct.	*p-value*	LI	EV	LS
f1	1.0000	-	-	-
f2	1.0000	-	-	-
f3	0.0000	-61.3721	-50.0000	-38.6279
f4	0.0000	-0.8797	-0.7780	-0.6763
f5	6.6437e-011	-49.2521	-37.8800	-26.5079
f6	1.0000	-	-	-
f7	1.0000	-	-	-
f8	8.9338e-010	-44.8755	-34.0000	-23.1245
f9	2.0054e-009	-44.1559	-33.2800	-22.4041
f10	1.0000	-	-	-
f11	0.3162	-	-	-
f12	1.0000	-	-	-
f13	1.0000	-	-	-

7 CONCLUSIONS

We have introduced a bi-population PSO algorithm, which is proposed to solve unconstrained numerical optimization. The results reached by Bi-PSO are good and competitive with respect to Differential Evolution (DE) algorithm (which is one of the best evolutionary algorithms to treat constrained and unconstrained problems). Our results indicate that Bi-PSO, which is simpler than DE obtains comparable performance for the best values found although it presented more variability in the mean values. As part of our future work, we aim to study alternative schemes to maintain diversity. Another goal is to improve the robustness of our approach, so that the variability of results significantly decreases, without degrading the quality of the best solutions found.

8 ACKNOWLEDGEMENTS

The authors would like to thank Mg. Jesús Velázquez Reyes and PhD. Carlos Coello Coello to provide us the Differential Evolution algorithm source code in order to make the proofs. Also we gratefully acknowledge constant support from the University and ANPCYT.

REFERENCES

[1] L. Cagnina, S. Esquivel, and R. Gallard. Particle swarm optimization for sequencing problems: a case study. In *Congress on Evolutionary Computation*, pages 536–541, Portland, Oregon, USA, 2004. http://www.lidic.unsl.edu.ar/publicaciones/in- fo_publicacion.php?id_publicacion=200.

[2] L. C. Cagnina, S. C. Esquivel, and C. A. Coello Coello. A particle swarm optimizer for constrained numerical optimization. In *9th International Conference - Parallel problem Solving from Nature - PPSN IX*, pages 910–919, Reykjavik, Island, 2006.

[3] W. Cedeno and D. Agrafiotis. Particle swarms for drug design. In *Proceeding of the IEEE Congress on Evolutionary Computation 2005*, pages 479–486, Edinburgh, Scotland, 2005.

[4] E. Correa, A. Freitas, and C. Johnson. A new discrete particle swarm algorithm applied to attribute selection in a bioinformatics data set. In *Proceeding of the GECCO 2006*, pages 35–42, Seattle, Washington,USA, 2006.

[5] L. Diosan and M. Oltean. Evolving the structure of the particle swarm optimization algorithms. In J. Gottlieb and G. R. Raidl, editors, *EvoCOP 2006, LNCS 3906*, pages 25–36. Springer-Verlag Berlin Heidelberg 2006, 2006.

[6] R. Eberhart and Y. Shi. A modified particle swarm optimizer. In *International Conference on Evolutionary Computation, IEEE Service Center*, Anchorage, AK, Piscataway, NJ, 1998.

[7] S. Esquivel and C. Coello Coello. On the use of particle swarm optimization with multimodal functions. In IEEE Press, editor, *Proceedings of the Congress on Evolutionary Computation (CEC 2003)*, Canberra, Australia, December 2003.

[8] M. Hollander and D. A. Wolfe. *Nonparametric Statistical Methods*. Wiley, 1973.

[9] X. Hu, R. Eberhart, and Y. Shi. Swarm intelligence for permutation optimization: a case study on n-queens problem. In *Proceeding of the IEEE Swarm Intelligence Symposium*, pages 243–246, Indianapolis, Indiana, USA, 2003.

[10] W. Jian, Y. Xue, and J. Qian. An improved particle swarm optimization algorithm with disturbance. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, pages 5900–5904, 2004.

[11] J. Kennedy. Small world and mega-minds: effects of neighborhood topologies on particle swarm performance. In *1999 Congress on Evolutionary Computation*, pages 1931–1938, Piscataway, NJ, 1999. IEEE Service Center.

[12] J. Kennedy. Bores bones particle swarm. In *IEEE Swarm Intelligence Symposium*, pages 80–87, 2003.

[13] J. Kennedy and R. Eberhart. The particle swarm: social adaptation in information-processing systems. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Organization*, 1999.

[14] J. Kennedy and R. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, California, USA, 2001.

[15] J. Kennedy and W. Spears. Matching algorithm to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In *IEEE Conference on Evolutionary Computation*, Anchorage, Alaska, 1998.

[16] R. Krohling, H. Knidel, and Y. Shi. Solving numerical equations of hydraulic problems using particle swarm optimization. In *Congress on Computational Intelligence*, pages 1968–1961, Honolulu, Haway, 2002.

[17] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive learning particle sawrm optimizer for global optimization of multimodal functions. In *IEEE Transactions on Evolutionary Computation*, volume 10, pages 281–295, June 2006.

[18] Jesús Velázquez Reyes. Propuesta de evolución diferencial para optimización de espacios restringidos. Master's thesis, Departamento de Ingeniería Eléctrica, Sección de Computación, Centro de Investigación y de Estudios Avanzados del IPN, México, 2006.

[19] J. Riget and S. Vesterstrom. A diversity-guided particle swarm optimizer - the arpso. Technical report, EVALife, 2002. http://www.evalife.dk.

[20] M. Voss. Principal component particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 2005.

[21] X. Yao and Y. Liu. Fast evolution strategies. In T. Bäck G. Rozenberg and A. E. Eiben, editors, *Advances in Evolutionary Computing: theory and applications*, pages 45–94, New York, USA, 2003. Springer-Verlag New York.