A New Binary PSO with Velocity Control

Laura Lanzarini, Javier López,
Juan Andrés Maulini, and Armando De Giusti

III-LIDI (Institute of Research in Computer Science LIDI)
Faculty of Computer Science, National University of La Plata
La Plata, Buenos Aires, Argentina

Abstract. Particle Swarm Optimization (PSO) is a metaheuristic that is highly used to solve mono- and multi-objective optimization problems. Two well-differentiated PSO versions have been defined – one that operates in a continuous solution space and one for binary spaces. In this paper, a new version of the Binary PSO algorithm is presented. This version improves its operation by a suitable positioning of the velocity vector. To achieve this, a new modified version of the continuous gBest PSO algorithm is used. The method proposed has been compared with two alternative methods to solve four known test functions. The results obtained have been satisfactory.

Keywords: Swarm Intelligence, PSO, Binary PSO, Velocity Control.

1 Introduction

Particle Swarm Optimization (PSO) is a metaheuristic proposed by Kennedy and Eberhart [1]. Its operation is based on the simulation of simple social models, and it has been successfully used in function optimization, as well as in neural network training [2].

There are different versions that were developed from the original idea, most of them related to the variation of various algorithm parameters or to the combination and variation of various topologies, sizes, and number of populations [3][4][5][6]. Variations of the algorithm with changes in the way particle velocity is updated have also been proposed, aimed at achieving diversity and avoiding stagnation in the search process [7] [8] [9].

In [10], Kennedy and Eberhart introduced a discrete binary version of PSO for discrete optimization problems. With binary PSO, each particle is represented as a string of zeros and ones. Unlike the continuous version, the discrete version uses the velocity vector as a probability function that allows deciding the value that should take each binary digit that determines the position of the individual.

This paper is organized as follows: In Section 2, the basic components of the PSO algorithm, both in its continuous and binary versions, are described. In Section 3, the method proposed is described, as well as the differences with its original version. In Section 4, the results obtained when comparing the performance of the new algorithm with other binary versions are presented. Finally, Section 5 includes a summary of this paper highlighting the most relevant aspects.

2 Particle Swarm Optimization

2.1 Continuous Particle Swarm Optimization

In PSO, each individual represents a possible solution to the problem and adapts following three factors: its knowledge of the environment (its fitness value), its previous experiences (its memory), and the previous experiences of the individuals in its neighborhood [1]. In this type of technique, each individual is in continuous movement within the search space and never dies.

Each particle is composed by three vectors and two fitness values:

– Vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ stores the current position of the particle
– Vector $pBest_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ stores the best solution found for the particle
– Velocity vector $v_i = (v_{i1}, v_{i2}, \ldots, v_{in})$ stores the gradient (direction) based on which the particle will move.
– The fitness value $fitness\_x_i$ stores the suitability value of the current solution.
– The fitness value $fitness\_pBest_i$ stores the suitability value of the best local solution found so far (vector $pBest_i$)

The position of a particle is updated as follows:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{1}$$

As explained above, the velocity vector is modified taking into account its experience and environment. The expression is:

$$v_i(t+1) = w.v_i(t) + \varphi_1.rand_1.(p_i{-}x_i(t)) + \varphi_2.rand_2.(g_i{-}x_i(t)) \tag{2}$$

where w represents the inertia factor [12], φ_1 and φ_2 are acceleration constants, $rand_1$ and $rand_2$ are random values belonging to the (0,1) interval, and g_i represents the position of the particle with the best *pBest* fitness in the environment of x_i (*lBest* or *localbest*) or the entire swarm (*gBest* or *globalbest*). The values of w, φ_1 and φ_2 are important to ensure the convergence of the algorithm. For detailed information regarding the selection of these values, please see [13] and [14].

2.2 Binary Particle Swarm Optimization

PSO was originally developed for a space of continuous values and it therefore poses several problems for spaces of discrete values where the variable domain is finite. Kennedy and Eberhart [10] presented a discrete binary version of PSO for these discrete optimization problems.

In binary PSO, each particle uses binary values to represent its current position and the position of the best solution found. The velocity vector is updated as in the continuous version, but determining the probability that each bit of the position vector becomes 1. Since this is a probability, the velocity vector should be mapped in such a way that it only contains values within the [0,1] range. To this end, the sigmoid function indicated in (3) is applied to each of its values.

$$v'_{ij}(t) = sig(v_{ij}(t)) = \frac{1}{1 + e^{-v_{ij}(t)}} \tag{3}$$

Then, the particle position vector is updated as follows

$$x_{ij}(t+1) = \begin{cases} 1 \; if \; rand_{ij} < sig(v_{ij}(t+1)) \\ 0 \; if \; not \end{cases} \tag{4}$$

where $rand_{ij}$ is a number ramdomly generated by an uniform pdf in [0,1].

It should be mentioned that the incorporation of the sigmoid function radically changes the way in which the velocity vector is used to update the position of the particle. In continuous PSO, the velocity vector takes on higher values first to facilitate the exploration of the solution space, and then reduces them to allow the particle to stabilize. In binary PSO, the opposite procedure is applied. Each particle increases its exploratory ability as the velocity vector reduces its value; that is, when v_{ij} tends to zero, $\lim\limits_{t \to \infty} sig(v_{ij}(t)) = 0.5$, thus allowing each binary digit to take a value of 1 with a probability of 0.5. This means that it could take on either value. On the contrary, when the velocity vector value increases, $\lim\limits_{t \to \infty} sig(v_{ij}(t)) = 1$, and therefore all bits will change to 1, whereas when the velocity vector value decreases, taking negative values, $\lim\limits_{t \to \infty} sig(v_{ij}(t)) = 0$ and all bits will change to 0. It should be noted that, by limiting the velocity vector values between -3 and 3, $sig(v_{ij}) \in [0.0474, 0.9526]$, whereas for values above 5, $sig(v_{ij}) \simeq 1$ and for values below -5, $sig(v_{ij}) \simeq 0$.

3 Binary PSO with Velocity Control. Method Proposed.

Based on the observations of the behavior of the velocity vector in the binary PSO algorithm defined in [10], and on the importance of correctly calculating the probabilities that allow changing each binary digit, a modified version of the original PSO algorithm to modify the velocity vector is proposed.

Under this new scheme, each particle will have two velocity vectors, $v1$ and $v2$. The first one is updated according to (5).

$$v1_i(t+1) = w.v1_i(t) + \varphi_1.rand_1.(2 * p_i - 1) + \varphi_2.rand_2.(2 * g_i - 1) \tag{5}$$

where the variables $rand_1$, $rand_2$, φ_1 and φ_2 operate in the same way as in (3). The values p_i and g_i correspond to the i^{th} binary digit of the $pBest_i$ and $gBest$ vectors, respectively.

The most significant difference between (3) and (5) is that in the latter, the shift of vector $v1$ in the directions corresponding to the best solution found by the particle and the best global solution does not depend on the current position of the particle. Then, each element of the velocity vector $v1$ is controlled by applying (6)

$$v1_{ij}(t) = \begin{cases} \delta 1_j & if \; v1_{ij}(t) > \delta 1_j \\ -\delta 1_j & if \; v1_{ij}(t) \leq -\delta 1_j \\ v1_{ij}(t) & if \; not \end{cases} \tag{6}$$

where

$$\delta 1_j = \frac{limit1_{upper_j} - limit1_{lower_j}}{2} \tag{7}$$

That is, velocity vector $v1$ is calculated with (5) and controlled with (6). Its value is used to update velocity vector $v2$, as shown in (8).

$$v2(t+1) = v2(t) + v1(t+1) \tag{8}$$

Vector $v2$ is also controlled as vector $v1$ by changing $limit1_{upper_j}$ and $limit1_{lower_j}$ by $limit2_{upper_j}$ and $limit2_{lower_j}$, respectively. This will yield $\delta 2_j$, which will be used as in (6) to limit the values of $v2$. Then, the new position of the particle is calculated with (4) using the values of $v2$ as arguments of the sigmoid function.

4 Results Obtained

In this section, the performance of the proposed binary PSO variation will be compared to the performance of the method proposed by Kennedy and Eberhart in [10] and the binary PSO defined in [11]. A known set of N-dimensional test functions was minimized.

Forty independent runs were performed for each of the methods using 2,000 iterations. N=3, 5,10 and 20 variables were used. Population size was always 20 particles. The values of $limit1$ and $limit2$ are the same for all variables; $[0;1]$ and $[0;6]$, respectively. Thus, probabilities within the interval $[0.0474, 0.9526]$ can be obtained. The values for φ_1 and φ_2 were set at 0.25 in all cases. As regards [10] and [11] methods, velicty limits were set within $[-3, 3]$ in order to keep the same range of probabilities.

The test functions used were Sphere, Rosenbrock, Griewangk, and Rastrigin, which were assigned numbers 1 to 4, respectively.

In table 1, the fitness value for the best solution found by each method is shown, as well as the average best fitness value for all 40 runs.

As it can be observed, the method proposed finds the best solutions and has the lowest average fitness values.

Table 2 indicates whether the results obtained are significant. The symbol ▲ was used to represent that $p - value < 0.05$, indicating that the null hypothesis must be rejected. The test carried out is of the lower tail type, whose null hypothesis states that the mean of the method indicated on the corresponding row is not lower than the mean of the method indicated on the corresponding column. The symbol $\triangledown$ indicates that the null hypothesis is not rejected.

Figure 1 shows the plot box diagrams calculated with the best results obtained in each of the 40 runs. Each column corresponds to a different function. Diagrams on the same row correspond to the same number of variables. From top to bottom, rows 1, 2, 3 and 4 correspond to the results obtained when assessing the functions with 3, 5, 10 and 20 variables, respectively. In each figure, methods are numbered from 1 to 3 and correspond to: the method proposed, Binary PSO from [10], and Binary PSO from [11], respectively. As it can be seen, the method proposed offers better solutions than the other two PSO alternatives.

Table 1. Results obtained

		Proposed method		Binary PSO [10]		Binary PSO [11]	
Nro. Var	Nro. Funct	Best Fitness	Average best Fitness	Best Fitness	Average best Fitness	Best Fitness	Average best Fitness
3	1	0	**0**	0	1,2e-09	1,8e-08	6,3e-07
3	2	7,0e-04	**2,8**	1,1e-05	3,0	2,0e-03	5,6
3	3	2,1e-09	**3,3e-03**	2,1e-09	6,8e-03	4,2e-06	8,8e-03
3	4	5,4e-08	**5,4e-08**	5,4e-08	**5,4e-08**	8,9e-06	7,6e-04
5	1	0,0	**3,1e-09**	1,4e-07	1,3e-05	1,7e-04	1,2e-03
5	2	2,2	**28,7**	2,1	111,5	7,2	278,3
5	3	2,6e-09	**8,2e-03**	1,6e-03	2,0e-02	1,9e-02	6,6e-02
5	4	9,0e-08	**1,3e-07**	2,3e-04	5,1e-01	5,0e-01	3,7
10	1	8,2e-05	**9,8e-04**	1,3e-02	7,1e-02	9,7e-02	6,2e-01
10	2	7,3	**141,0**	334,0	2812,8	92013,0	613510,0
10	3	1,3e-02	**7,6e-02**	7,7e-02	1,9e-01	5,2e-01	7,3e-01
10	4	0,8	**4,3**	7,5	15,3	13,7	44,0
20	1	3,3e-01	**8,1e-01**	1,7e+00	3,9e+00	1,2e+01	1,9e+01
20	2	1865,9	**10105**	2,8e+05	1,2e+07	1,2e+08	5,0e+08
20	3	1,4e-01	**2,7e-01**	9,3e-01	1,0e+00	1,3e+00	1,4e+00
20	4	27,7	**43,0**	52,7	88,9	169,8	215,2

Table 2. Results of hypothesis tests

nro Var.	Método	Proposed Binary PSO	Binary PSO [10]	Binary PSO [11]
3	Proposed binary PSO		▽ ▽ ▽ ▽	▲ ▽ ▲ ▽
3	Binary PSO [10]	▽ ▽ ▽ ▽		▲ ▽ ▽ ▽
3	Binary PSO [11]	▽ ▽ ▽ ▽	▽ ▽ ▽ ▽	
5	Proposed binary PSO		▽ ▽ ▲ ▽	▲ ▲ ▲ ▲
5	Binary PSO [10]	▽ ▽ ▽ ▽		▲ ▽ ▲ ▲
5	Binary PSO[11]	▽ ▽ ▽ ▽	▽ ▽ ▽ ▽	
10	Proposed binary PSO		▲ ▲ ▲ ▲	▲ ▲ ▲ ▲
10	Binary PSO [10]	▽ ▽ ▽ ▽		▲ ▲ ▲ ▲
10	Binary PSO [11]	▽ ▽ ▽ ▽	▽ ▽ ▽ ▽	
20	Proposed binary PSO		▲ ▲ ▲ ▲	▲ ▲ ▲ ▲
20	Binary PSO [10]	▽ ▽ ▽ ▽		▲ ▲ ▲ ▲
20	Binary PSO [11]	▽ ▽ ▽ ▽	▽ ▽ ▽ ▽	

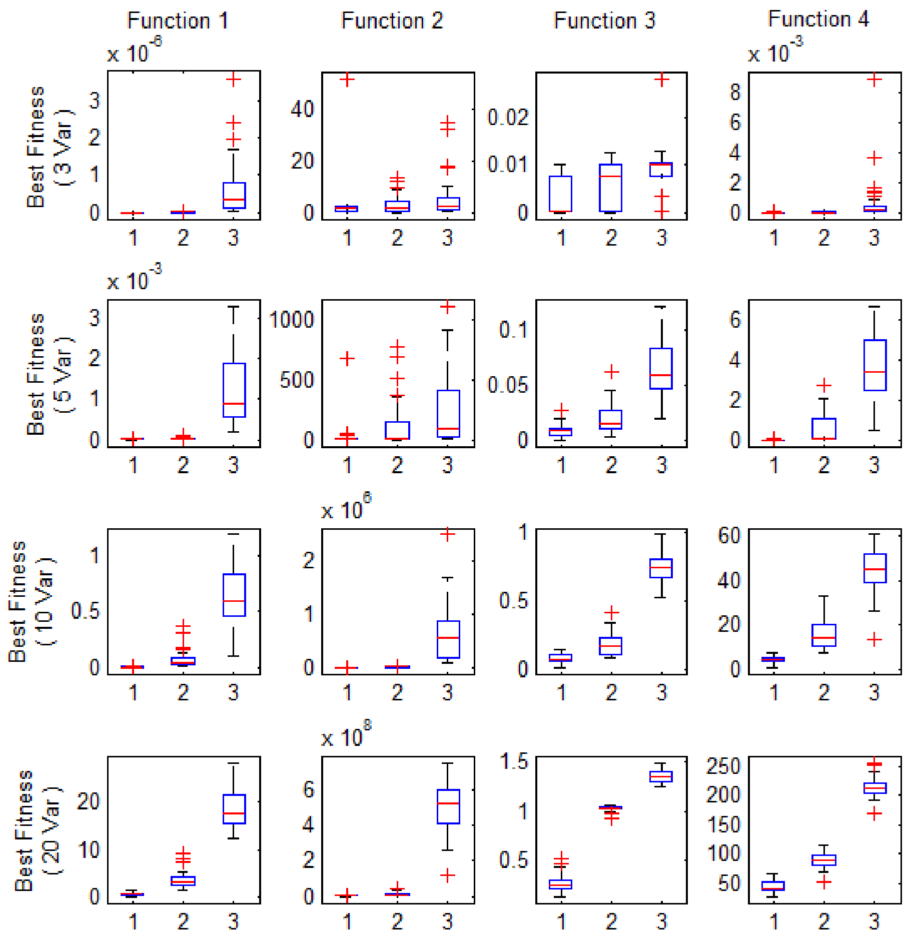

Fig. 1. Boxplots corresponding to the best solutions obtained in each of the 40 independent runs. The method is indicated on the x-axis: 1 = Method proposed, 2 = Binary PSO from [10] and 3 = Binary PSO from [11]. Each row indicates the results obtained with 3, 5, 10 and 20 variables.

Figure 2 is organized in a similar fashion. It shows the boxplots corresponding to the average fitness of each of the 40 runs performed for each function and for each number of variables considered. The populational diversity of each method can be observed there. In general, based on box heights, it can be said that, even though the method from [11] presents the greater inter-quartile ranges, its solutions are the worst. As for the method proposed and the Binary PSO from [10], ranges are equivalent.

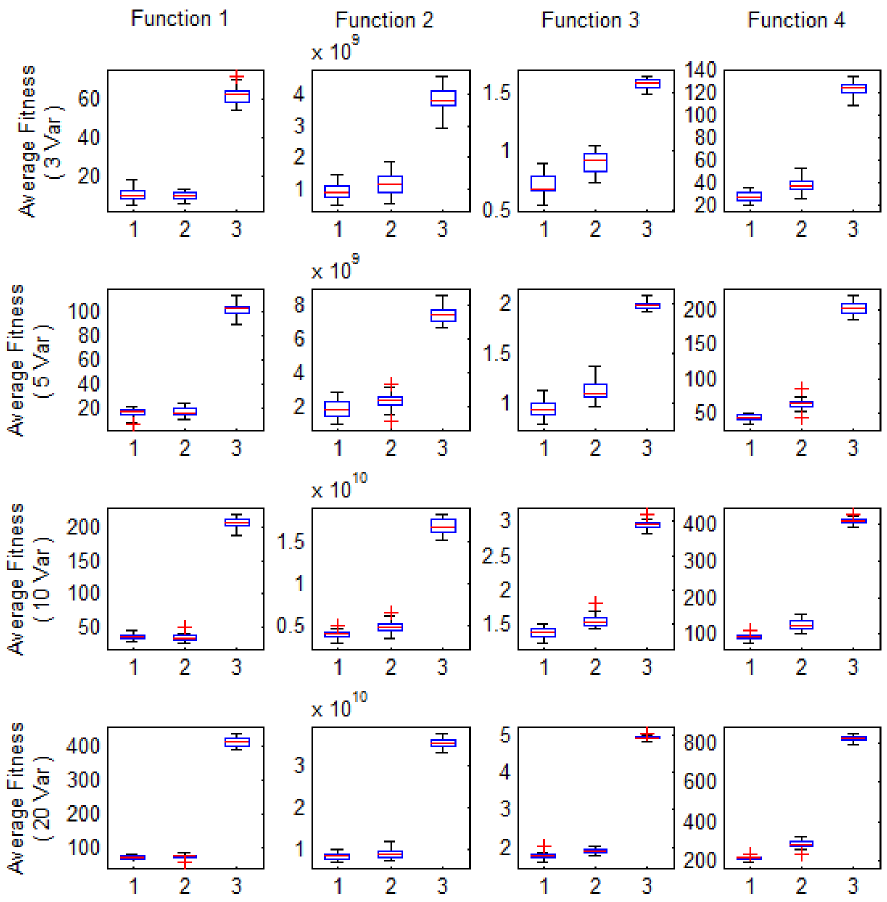

Fig. 2. Boxplots corresponding to the average fitness of each of the 40 independent runs. The method is indicated on the x-axis: 1 = Method proposed, 2 = Binary PSO from [10] and 3 = Binary PSO from [11]. Each row indicates the results obtained with 3, 5, 10 and 20 variables.

5 Conclusions

A variation of the binary PSO method originally proposed in [10] that controls velocity vector changes by using a variation of the continuous PSO method has been presented.

The results obtained by minimizing a set of test functions are better than those obtained with the methods defined in [10] and [11] for all the cases assessed.

Table 2 shows that, as the number of variables used increases, the difference in means becomes more significant.

Figure 1 shows that, in most of the runs, the results obtained with the method proposed have been better than those obtained with the other two methods.

Similarly, Figure 2 shows that the method proposed generates the best population average fitness results, at least for the test functions assessed. If we consider the average fitness inter-quartile range (height of the boxplots in Figure 2), it can be stated that the method proposed in this paper and the Binary PSO from [10] are equivalent, the best solutions being offered by the former.

Currently, our research is focused on measuring the performance of the algorithm proposed with an increasing number of dimensions in the problem, in order to apply the algorithm to real-world problem resolution.

It would also be interesting to analyze its performance using variable-size PSO [6]. This would allow adapting the swarm size based on the complexity of the problem to solve.

References

1. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, Perth, Australia, vol. IV, pp. 1942–1948. IEEE Service Center, Piscataway (1995)
2. Hu, X., Shi, Y., Eberhart, R.: Recent Advances in Particle Swarm. In: Congress on Evolutionary Computation, CEC 2004, vol. 1, pp. 90–97 (2004)
3. Cagnina, L., Esquivel, S., Coello Coello, C.: A bi-population PSO with a shake-mechanism for solving constrained numerical optimization. In: IEEE Congress on Evolutionary Computation, CEC 2007, pp. 670–676 (2007)
4. Mohais, A., Mendes, R., Ward, C., Postho, C.: Neighborhood Re-Structuring in Particle Swarm Optimization. In: Zhang, S., Jarvis, R.A. (eds.) AI 2005. LNCS (LNAI), vol. 3809, pp. 776–785. Springer, Heidelberg (2005)
5. Atyabi, A., et al.: Particle Swarm Optimizations: A Critical Review. In: 5th International Conference on Information and Knowledge Technology, Mashad, Iran (2007)
6. Lanzarini, L., Leza, V., De Giusti, A.: Particle Swarm Optimization with Variable Population Size. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 438–449. Springer, Heidelberg (2008)
7. Riget, J., Vesterstrom, J.: A Diversity-Guided Particle Swarm Optimizer – the ARPSO. EVALife Project Group Department of Computer Science (2002)
8. Clerc, M.: Stagnation Analysis in Particle Swarm Optimisation or What Happens When Nothing Happens.Department of Computer Science; University of Essex. Technical Report CSM-460 (2006)
9. Peer, E., Van den Bergh, F., Engelbrecht, A.: Using Neighbourhoods with the Guaranteed Convergence PSO. In: Swarm Intelligence Symposium, SIS 2003, pp. 235–242 (2003)
10. Kennedy, J., Eberhart, R.: A discrete binary version of the particle swarm algorithm. In: Proc. of the World Multiconference on Systemics, Cybernetics and Informatics (WMSCI), pp. 4104–4109 (1997)
11. Khanesar, M., Teshnehlab, M., Shoorehdeli, M.: A novel Binary Particle Swarm Optimization. In: 18th Mediterranean Conference on Control and Automation, Athens, pp. 1–6 (June 2007)

12. Shi, Y., Eberhart, R.: Parameter Selection in Particle Swarm Optimization. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 591–600. Springer, Heidelberg (1998) ISBN 3-540-64891-7
13. Clerc, M., Kennedy, J.: The particle swarm – explosion, stability and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
14. Van den Bergh, F.: An analysis of particle swarm optimizers. Ph.D. dissertation. Department Computer Science. University Pretoria. South Africa (2002)