lpPSO - New optimization strategy inspired by PSO

Waldo Hasperué[1,2] and Leonardo Corbalan[1]

[1] III-LIDI. School of Computer Science – UNLP. Argentina.
[2] CONICET Scholarship.
{whasperue, corbalan}@lidi.info.unlp.edu.ar

Abstract. Given the large number of optimization problems that mankind faces, metaheuristics are very important strategies for the resolution of these problems. These strategies assess the suitability of the individuals, which represent solutions to the problem, a large number of times throughout the search for an optimal solution. When the assessment of an individual takes significant time or resources, the assessment of hundreds or thousands of individuals is a problem to be taken into consideration. In this paper, a strategy based on PSO that considerably reduces the number of individual assessments is presented, which is of great help for complex problems. The method proposed was compared with the classical version of PSO using classic functions in the space and a real case with a simulation model, and satisfactory results were obtained.

Keywords: metaheuristics, optimization problems, PSO.

1 Introduction

Optimization, in the sense of finding a solution that is acceptable for a given problem, is a very important task in various disciplines. From tasks such as finding the shortest path or organizing tasks to take as little time as possible, to problems such as finding the maximum of a function, the number of optimal clusters in a database or adjusting parameters in simulation models, human beings are constantly looking for ways to solve optimization problems. When the problem is very complex, this is no simple task, and tools to help solving them become necessary.

Various techniques have been developed that solve optimization problems in an exact or approximate fashion. When exact methods are difficult or non-existent, metaheuristics become the alternative to solve them by way of approximation. These are based on the integration of efficient strategies in calculation time and memory use capable of finding an acceptable solution (close to the optimum) by examining only a subset of the search space.

In particular, Particle Swarm Optimization (PSO) [1] is a metaheuristic widely used in the resolution of this type of problems [2] [3] [4] [5] [6]. This technique works with various particles, each representing a solution to the problem and travelling the space in search for the best solutions. To decide in which direction to move, they are based on the best solution found by the particle itself and on the best solution found either by the entire swarm (global PSO) or a subset of the swarm (local PSO).

One of the problems of PSO is that in multimodal functions, a large part of the swarm conglomerates in a reduced area of the search space, leaving others unexplored. In this paper, we present lpPSO (long path PSO), which is a strategy based on PSO that works with particles that, in their travel, cover a larger area of the search space by changing direction only when they reach a limit of the space. Thus, the technique proposed makes sure to cover a large part of the search space, avoiding local minima or maxima. An area with a possible optimal solution will be "visited" simultaneously from various directions by all particles.

The rest of this paper is organized as follows: Section 2 briefly describes the PSO strategy. In Section 3, the strategy proposed is presented. In Section 4, the results obtained are shown, and finally, in Section 5, some possible improvements that can be introduced as future work are discussed.

2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) [1] is a populational metaheuristic technique where each particle represents a possible solution to the problem and adapts during the algorithm taking three elements into account: its knowledge of the environment (fitness value), its previous experience (memory) and the previous experience of its neighbors. The objective of each particle is to move constantly within the search space to find a good solution for the problem. That is, trying to improve itself by constantly looking for better solutions.

There are two PSO versions that are most widely used. gBest PSO uses the criterion that the entire swarm belongs to a single neighborhood, which means that the entire swarm will go in the direction of the best particle, whereas lBest PSO uses part of the swarm as the neighborhood for any given particle, so that each particle has its own neighborhood and will move in the direction of the best particle within its own neighborhood. The size of the neighborhood directly affects populational diversity – the larger the neighborhood, the less diverse its members are, but the algorithm converges faster.

Each particle is formed by the following elements:

- A vector x that stores its current position within the search space.

- A vector *pBest* that stores the position of the best solution found by the particle itself

- A velocity vector v that stores the speed and direction in which the particle moves.

- The fitness value of the current position.

- The fitness value of the best solution found.

The position of the particle i is updated as follows

$$x_i(t+1) = x_i(t) + v_i(t) . \qquad (1)$$

The velocity vector is updated taking into account its own experience and that of the environment

$$v_i(t+1) = w.v_i(t) + \delta_1.\text{rand}_1.(pBest_i - x_i(t)) + \delta_2.\text{rand}_2.(g_i - x_i(t)) . \quad (2)$$

Where w is an inertia factor, δ_1 and δ_2 are acceleration constants, rand_1 and rand_2 are random distribution values U(0,1) and g_i represents the best solution of the individual's environment, either with gBest or lBest. w, δ_1 and δ_2 are algorithm parameters and are essential for the convergence of the algorithm [7] [8] [9].

3 lpPSO

In search metaheuristics, most of the computation requirements fall on the assessment of particle fitness. This assessment may be a single calculation of a function or, in more complex tasks such as the assessment of simulation models, testing of neural networks, assessment of graphs, etc.

In problems where assessing the fitness of a particle demands a large amount of computation, the assessment of the entire swarm of particles throughout the search process is the most expensive part and the one that most resources consumes. Thus, total algorithm time can be reduced by minimizing the number of fitness assessments.

lpPSO reduces the number of fitness assessments in relation to PSO.

3.1 Exploration Strategy

The same as PSO, this strategy uses particles that represent solutions to the base problem. Particles have vectors x, v and $pBest$ (position, velocity, and best solution found, respectively) and the fitness values for the current position and the best solution.

In the strategy proposed, a constant number of particles explore the limited d-dimensional search space, covering extensively from one end to another. The limits of the search space are usually clear in most optimization problems. Thus, the search space B is defined and limited by values a_l and b_l for each dimension

$$B = \{(x_1, x_2, \dots , x_d) \in R^d / a_l <= x_l <= b_l \;\forall\; l=1..d\} . \quad (3)$$

For lpPSO to achieve its objective, the search space B must be associated to a continuous error surface, so that the optimal solutions will be characterized by having some slope getting to them.

Each of the particles starts with a random position and direction. They travel in that direction until they reach the limit of the search space. The movement of a particle i is similar to that of PSO, and the vector x is updated as follows

$$x_i(t + 1) = x_i(t) + v_i(t)\, k_i . \quad (4)$$

Where k_i is an acceleration factor of particle i.

Let p_i be the i^{th} particle of the swarm. When p_i reaches the limit of the space, it changes its direction towards the best solution found by particle p_{i+1}. It will continue to move in that direction until a new limit is reached, changing then its direction towards the best solution of particle p_{i+2}. This process is repeated selecting the next particle in the swarm in a circular queue order. Once all particles changed their trajectory *n*-1 times, all of them have moved in the direction of the best solution found by each of the other particles.

The purpose of this series of trajectory changes is that a particle, while crossing the search space, goes in the direction of the area where another particle found a possible optimal solution; thus particle *i* can "observe" that area by approaching it from a different direction. Figure 1 shows an example of four particles that start from a random position and change their trajectories until finding an optimal solution.

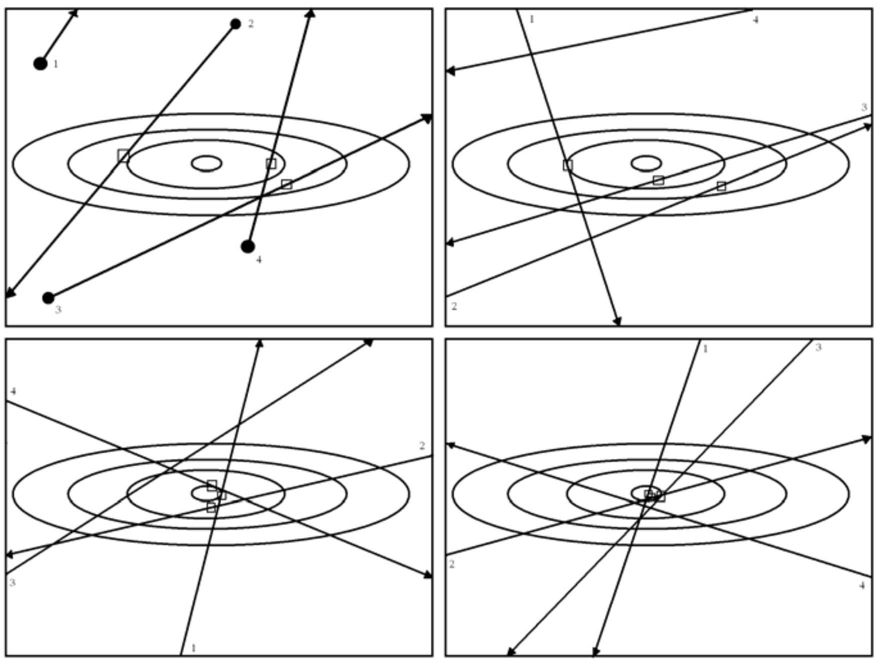

Fig. 1. Particle direction change procedure. The example shows a mostly flat surface that rises in an elliptical fashion until reaching a maximum that is at the center of the smallest ellipse. a) initial position of the particles (*circles*) and the best solution found (*square*) bye each of them in their trajectory (*arrow*). b) all particles p_i change their direction towards the best solution of particle p_{i+1}. c) the change is done towards the best solution of particle p_{i+2}. At this point, all particles but # 3 find a better solution. d) the change is done towards the best solution of particle p_{i+3}.

In lpPSO, particles not only reach the area where a possible best solution is found, but they also explore other areas of the space. Figure 2 shows an example where one particle moving towards the best solution of another particle, finds a solution that is even better. This is possible because the strategy never stops exploring new areas of the space even when all particles are intensifying their search in a promising area.

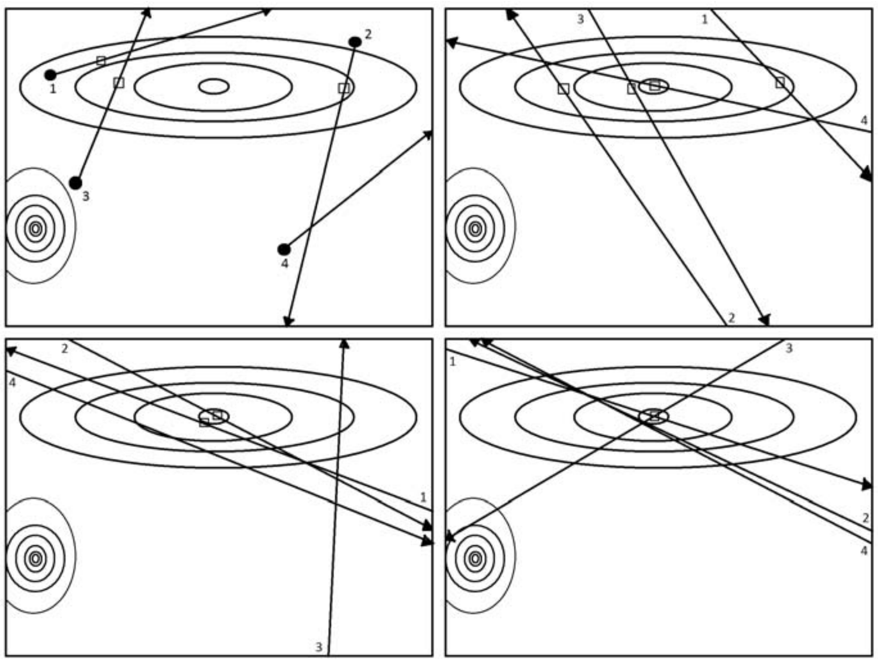

Fig. 2. The example shows the case of a particle that, when going through the area of a good solution, finds an even better one. The error surface, mostly flat, rises in an elliptical fashion until reaching two maxima that are at the centers of each group of the concentric ellipses. Group B has a higher fitness value than group A. a) initial position of the particles (*circles*) and the best solution found (*square*) bye each of them in their trajectory (*arrow*). b) and c) change in direction towards the best solution of particles p_{i+1} and p_{i+2}, respectively. d) particle 3 reaches a better solution in its trajectory beyond A. From that point on, all particles will progressively move in the direction of the solution found by particle 3, and in a few iterations they will find the highest fitness value.

3.2 Decrease in Particle Velocity

The velocity at which a particle crosses the search space is a critical parameter of the lpPSO algorithm. A particle that moves at a fast speed will have difficulty finding the good regions if these are small. On the other hand, if it moves at a very slow pace, the probability of examining the good regions increases at the cost of performing a large number of fitness assessments, which is precisely what the strategy proposed in this paper attempts to reduce.

To increase the exploratory capacity of the good regions, the particles reduce their velocity as they approach the region surrounding the best solution found by another particle, progressively until reaching a minimum velocity. As they move away from that region, the particles increase their velocity until reaching maximum speed. This allows achieving a good ratio between space exploration and promising region exploitation.

3.3 Algorithm

The lpPSO algorithm works with three parameters:

- *n*: number of swarm particles
- *velMax*: maximum velocity at which a particle can move.
- *velMin*: minimum velocity at which a particle can move.

The algorithm starts by placing *n* particles at different random positions within the search space, each with a direction that is also random.

The next step of the algorithm is an iterative process that consists in moving the particles, updating their velocities and changing their direction every time they reach a limit of the search space. This iterative process ends after a fixed number of iterations or when a satisfactory optimum is reached.

The pseudo-code of the algorithm is the following:

```
program  bounces  (n,  velMax,  velMin)
  begin
  algorithm initialization
  repeat
    for  i  =  1..n  do
      begin
        Adjust_velocity_magnitude (pi)
        Yield (pi);
        if  pi  reached_limit then
          Determine_new_direction  (pi)
        else
          assess_Fitness (pi);
      end;
  until reaching end condition;
end.
```

When the algorithm starts, initial position and velocity vectors are randomly assigned for all swarm particles with a uniform distribution in the interval defined by the limited space.

Initially, all particles move at the maximum velocity set by parameter *velMax*. The change in velocity magnitude is first applied when the particle reaches a limit of the space for the first time, and its new direction and magnitude are determined based on the best solution found by another particle. The change in magnitude of the velocity vector is established by a factor k_i that is calculated as a function of the Euclidean distance between the current position x_i and the position of the best solution towards which the particle is moving ($pBest_j$).

$$k_i(t) = \mathrm{D}(x_i(t), pBest_j(t)) + velMin. \tag{5}$$

Once ki is calculated, the particle moves as follows

$$x_i(t+1) = x_i(t) + v_i(t)\, k_i(t). \tag{6}$$

The factor ki calculated in Equation 5 determines the deceleration of the particle as it approaches $pBest_j$. The function of parameter *velMin* is to prevent the particle from stopping on $pBest_j$, allowing exploration beyond that solution. After going beyond $pBest_j$, and as the distance to it increases, the particle accelerates once again. This allows analyzing the regions that are near a good solution to be thoroughly analyzed.

When a particle reaches the limit of the space, it calculates *j* as the index of the next particle within a circular queue by means of Equation 7. This allows determining $pBest_j$, which will guide the new trajectory by calculating the velocity vector *v* as shown in Equation 8.

$$j = (j \bmod n) + 1 . \tag{7}$$

$$v_i(t + 1) = (x_i(t) - pBest_j(t))\ velMax . \tag{8}$$

4 Results

lpPSO was tested versus PSO, and satisfactory results were obtained. Two different tests were carried out. The first test consists in the assessment of five classical functions in a two-dimensional space, and the second test is resolution of a real case of parameterization of a biological simulation model.

4.1 Two-Dimensional Functions

Table 1 shows the five functions used for the first test. Each of the functions presents a minimum at some point within the space in which they are defined. Each function tries to measure a different aspect of the algorithm proposed. Function F_1 allows analyzing the accuracy of the solution found. F_2 and F_3 allow observing the behavior of the algorithm when there is a large number of local minima and maxima. F_4 and F_5 measure the exploratory capacity of the algorithm, since they represent a completely flat surface with one "hole" in the case of F_4 and three "holes" of different depths in the case of F_5.

In this test, one particle represents a point in the space, and its corresponding fitness is the value of the function assessed at that point. Since the concept of fitness in metaheuristics is directly proportional to the magnitude, i.e., the higher the value of the function, the better the fitness, and these functions attempt to reach a minimum, PSO and lpPSO algorithms were modified to consider that fitness values are better when their value is smaller.

The results obtained were averaged from the data obtained from 30 independent runs. For the PSO tests, the gBest PSO version was used, since it has shown to yield better results in this type of functions [10]. In PSO, trials were done with 5, 10, 20, 30, 40, 50, 60 and 70 particles; the best result obtained is shown in Table 1. Each trial was run with a maximum of 500 iterations, adding to the results of the run the number of fitness assessments done until finding a better solution.

In lpPSO, eight particles were used for all tests and a maximum of 200 trajectory changes per particle. Table 2 shows the results obtained in these tests.

Table 1. The five functions used for the first test.

Function	Interval
$F_1(x,y) = x^2 + y^2$	*x, and* ∈ [-1, 5]
$F_2(x,y) = 0.5 + \frac{(\sin(\sqrt{x^2+y^2+4}))^2 - 0.5}{(1+0.001(x^2+y^2))^2}$	*x, and* ∈ [-50, 50]
$F_3(x_1,x_2) = \frac{1}{0.002 + \sum_{j=1}^{25} \frac{1}{50j + \sum_{i=1}^{2}(x_i - a_{ij})^6}}$	x_1, x_2 ∈ [-50, 50]
$F_4(x,y) = \frac{1}{0.002 + \frac{1}{1+(x+1)^{12}+(y+1)^{12}}}$	x_1, x_2 ∈ [-200, 200]
$F_5(x_1,x_2) = \frac{1}{0.002 + \sum_{j=1}^{3} \frac{1}{50j^2 + \sum_{i=1}^{2}(x_i - b_{ij})^{12}}}$	x_1, x_2 ∈ [-50, 50]

Table 2. Results of the first test. For the five tested functions, the number of particles (*n*) used in each strategy is shown, as well as the best fitness found (*fb*) and the total number of assessments (*te*) that were needed to reach the best solution.

	PSO			lpPSO		
Function	*n*	*fb*	*te*	*n*	*fb*	*te*
F_1	20	4.69E-04	1032	10	4.65E-04	932
F_2	70	0.0515	5786	20	0.0487	4806
F_3	60	79.1778	5654	20	65.8941	4705
F_4	70	34.7161	11268	20	30.1567	9478
F_5	60	112.9385	3774	20	103.7195	3105

4.2 Real Case

The performance of lpPSO was tested in a real case of an optimization problem. This is a mathematical model designed to simulate the growth of a guanaco population during 30 years. Starting with an initial population of guanacos, by means of different calculations, the size of the population one year later is calculated; then, the same procedure is applied and the population for the second year is obtained, and so on until obtaining the population of guanacos during a period of 30 years.

This model has different parameters whose values result in different scenarios. The parameters include female fertility, i.e., how many calves per year a female produces, the probability of surviving until the following year of the guanacos based on their age or sex (survival probability for a calf, an adult male, an adult female, among others). There are also parameters that affect harvesting, i.e., how many guanaco specimens can be removed from the population in a year.

On the other hand, there is information of a census of the guanaco population. These data, called field data, were obtained by census (individual count) in a ranch in the province of Chubut for a period of 30 years. Harvesting data at different periods are also available for this censed population. The purpose of the simulation model is finding parameter values that reflect as best as possible the information offered by field data.

Thus, we have an optimization problem in an eight-dimensional space, since that is the number of parameters in the model. The space is limited, since for each parameter, the minimum and maximum value that can be used are known. Thus, a particle represents a combination of values for those eight parameters of the model. The sum of the square errors was used as fitness value for each particle. Since the model produces a series of values in time as output, and for that same series of values there are field data against which to compare them, the sum of the square errors (S) is as follows:

$$S = \Sigma\ (i=1..30)\ (y_i - m_i)^2\ . \qquad \textbf{(9)}$$

Where y_i is the i^{th} field data and m_i is the i^{th} value produced by the model. The same as in the previous test with the functions in the space, PSO and lpPSO algorithms were modified to consider the lowest values as having the best fitness. For PSO, tests were carried out using 50 and 100 particles, and a maximum of 500 iterations. For lpPSO, 50 and 100 particles were used and a maximum of 500 trajectory changes. Table 3 shows the results obtained.

Table 3. Results of the second test. For the two test cases, the number of particles (*n*) used in each strategy is shown, as well as the best fitness found (*fb*) and the total number of assessments (*te*) that were needed to reach the best solution.

	PSO			lpPSO		
Test	*n*	*fb*	*te*	*n*	*fb*	*te*
T_1	50	6.91E08	9756	50	6.87E08	4820
T_2	100	6.53E08	18230	100	6.38E08	12762

5 Conclusions and Future Works

A new search strategy inspired in PSO, called lpPSO, has been presented. This strategy considerably reduces the number of fitness assessments required. lpPSO was compared with the classical version of PSO by means of classical, two-dimensional function optimization problems and a real case applied to a biological simulation mathematical model. As it was observed in the results, lpPSO was better than PSO, with equal or better fitness values found as best solution and considerably reducing the total number of fitness assessments used by the algorithm.

Various swarm sizes have been tested, and no significant differences were observed in the results. As future work, the utilization of a variable-size swarm is proposed, as well as the dynamic reduction of the search space by excluding the least promising regions. This would result in an even greater reduction in the number of fitness assessments needed to find an optimal solution without degrading the current performance of the strategy.

References

1. Kenedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, Australia, vol. IV, pp. 1942–1948 (1995)
2. Cui, X., Potok, T.E., Palathingal, P.: Document Clustering Using Particle Swarm Optimization. In: IEEE Proceedings Swarm Intelligence Symposium, pp.185--191 (2005)
3. Omran, M.G.H., Salman A., Engelbrecht A.P.: Dynamic Clustering Using Particle Swarm Optimization with Application in Image Segmentation. Pattern Analysis & Applications. 8, 332--344 (2005)
4. Xiao, X., Dow, E.R., Eberhart, R., Miled, Z.B., Oppelt, R.J.: Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization. In: International Proceedings Parallel and Distributed Processing Symposium, pp. 1--10 (2003)
5. van der Merwe, D.W., Engelbrecht, A.P.: Data Clustering Using Particle Swarm Optimization. In: Congress on Evolutionary Computation, pp. 215--220 (2003)
6. Hung, C., Huang, L.: Extracting Rules from Optimal Clusters of Self-Organizing Maps. In: Second International Conference on Computer Modeling and Simulation, pp. 382--386 (2010)
7. Van den Bergh, F.: An analysis of particle swarm optimizers. Ph.D. dissertation. Department Computer Science. University Pretoria. South Africa (2002)
8. 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)
9. Shi, Y., Eberhart, R.: Parameter Selection in Particle Swarm Optimization. In: Proceedings of the 7th International Conference on Evolutionary Programming, pp. 591–600. Springer, Heidelberg (1998)
10. 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 Proceedings 9th International Conference on Artificial Intelligence and Soft Computing, pp. 438--449. Zakopane, Poland (2008)