Hybrid Particle Swarm - Evolutionary Algorithm for Search and Optimization

Crina Grosan[1], Ajith Abraham[2], Sangyong Han[2] and Alexander Gelbukh[3]

[1]Department of Computer Science
Babeş-Bolyai University, Cluj-Napoca, 3400, Romania
[2]School of Computer Science and Engineering
Chung-Ang University, Seoul 156-756, Korea
[3]Centro de Investigacin en Computacin (CIC)
Instituto Politcnico Nacional (IPN), Mexico
ajith.abraham@ieee.org, cgrosan@cs.ubbcluj.ro, hansy@cau.ac.kr,
gelbukh@gelbukh.com

Abstract. Particle Swarm Optimization (PSO) technique has proved its ability to deal with very complicated optimization and search problems. Several variants of the original algorithm have been proposed. This paper proposes a novel hybrid PSO - evolutionary algorithm for solving the well known geometrical place problems. Finding the geometrical place could be sometimes a hard task. In almost all situations the geometrical place consists more than one single point. The performance of the newly proposed PSO algorithm is compared with evolutionary algorithms. The main advantage of the PSO technique is its speed of convergence. Also, we propose a hybrid algorithm, combining PSO and evolutionary algorithms. The hybrid combination is able to detect the geometrical place very fast for which the evolutionary algorithms required more time and the conventional PSO approach even failed to find the real geometrical place.

1 Introduction

Evolutionary Algorithms (EA) use a population of potential solutions (points) of the search space. These solutions (initially randomly generated) are evolved using different specific operators which are inspired from biology. Through cooperation and competition among the potential solutions, these techniques often can find optima quickly when applied to complex optimization problems.

There are some similarities between PSO and Evolutionary Algorithms:

- both techniques use a population (which is called *swarm* in the PSO case) of solutions from the search space which are initially random generated;
- solutions belonging to the same population interact with each other during the search process;
- solutions are evolved (their quality is improved) using techniques inspired from the real world.

Even then, there are still many differences between these two techniques.

In what follows, we will apply both techniques for solving a well known geometrical place problems [6]. It is well known that in the case of these problems a set of points which accomplish a given condition (or a set of conditions) is explored. In many situations, the searched geometrical place consists of more than one point (solution). Evolutionary algorithms and PSO techniques are ideal candidates for this problem mainly due to their ability to deal with a population of solutions at the same time.

We propose a new particle swarm technique which is based on the basic PSO algorithm proposed by Eberhart and Kenedy in 1995. Some related work and existing variants of PSO can be found in [3], [4], [5], [8], [9], [10], [12], [13], [14].

The main scope of our paper is to perform a comparison between PSO and EA and to exploit the weakness/strength of each of them. Finally, taking into account of the results, we propose a hybrid algorithm combining PSO and EA which seems to perform better in complicated situations than each of these techniques considered separately.

The paper is structured as follows: Section 2 presents some basics of the PSO technique. Section 3 briefly describes the a new variant of the particle Swarm technique. The general evolutionary algorithm is described in Section 4. In section 5 some experiments using different test problems are performed. Conclusions and remarks are presented towards the end.

2 Particle Swarm Optimization Technique

Like other EA techniques, PSO is a population-based search algorithm and is initialized with a population of random solutions, called particles ([7]).

Unlike in the EA techniques, each particle in PSO is also associated with a velocity. Particles fly through the search space with velocities which are dynamically adjusted according to their historical behaviors. Therefore, the particles have the tendency to fly towards the better and better search area over the course of search process. The PSO was first designed to simulate birds seeking food which is defined as a 'cornfield vector' [8].

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating each generation.

Each individual is treated as a volume-less particle (a point) in the D-dimensional search space. The i^{th} particle is represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. At each generation, each particle is updated by following two 'best' values.

The first one is the best previous location (the position giving the best fitness value) a particle has achieved so far. This value is called *pBest*. The *pBest* of the i^{th}particle is represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$.

At each iteration, the P vector of the particle with the best fitness in the neighborhood, designated $l$or g, and the P vector of the current particle are combined to adjust the velocity along each dimension, and that velocity is then used to compute a new position for the particle. The portion of the adjustment

to the velocity influenced by the individual's previous best position (P) is considered the *cognition* component, and the portion influenced by the best in the neighborhood is the *social* component.

With the addition of the inertia factor, ω, by Shi and Eberhart [14] (for balancing the global and the local search), these equations are:
$v_{id} = \omega * v_{id} + \eta_x * \text{rand}() * (p_{id} - x_{id}) + \eta_2 * \text{Rand}() * (p_{gd} - x_{id})$ (1)
$x_{id} = x_{id} + v_{id}$ (2)
where rand() and Rand() are two random numbers independently generated within the range of [0,1] and $\eta 1$ and $\eta 2$ are two learning factors which control the influence of the social and cognitive components.

In (1) if the sum on the right side exceeds a constant value, then the velocity on that dimension is assigned to be $\pm V_{max}$. Thus, particles' velocities are clamped to the range of $[-V_{max}, V_{max}]$ which serves as a constraint to control the global exploration ability of the PSO algorithm. This also reduces the the likelihood of particles for leaving the search space. Note that this does not restrict the values of X_i to the range $[-V_{max}, V_{max}]$; it only limits the maximum distance that a particle will move during one iteration.

3 Proposed PSO Approach for Dealing with Geometrical Place problems

The proposed PSO algorithm is similar to the classical one which uses neighborhoods but still there are some differences which are described below. We consider the PSO algorithm with neighborhoods, but not overlapping ones as usual. Thus, the particles in the swarm 'fly' in independent sub-swarms. It is just like dividing the swarm into multiple independent 'neighborhoods'[1], [11]. The dimension of each neighborhood (sub-swarms) is the same for all considered sub-swarms. The reason for not choosing overlapping neighborhoods is that for the geometrical place problems the solution consists of a set of points and not merely a single point. In the classical PSO, each solution will follow the best solution in the swarm or the best solution located in its neighborhood. This means, finally all solutions will converge to the same point. But for the geometrical place problem we need to find a set of different solutions. By considering different sub-swarms, the number of solutions which can be obtained at the end of the search process might be at most equal to the number of sub-swarms (this in case each sub-swarm will converge to a different point). Taking into account all these considerations, we will consider small sub-swarms (having usually few particles: 4 or 5) so that we have the chances to obtain, finally, a greater number of different points (which is ideal for geometrical place problems). The algorithm proposed is called Independent Neighborhoods Particle Swarm Optimization (INPSO). The main steps of the INPSO algorithm are described below:

INPSO algorithm

while *iteration* <= *max_iterations* **do**

begin
for each particle p **do**
begin
Calculate fitness value
if the fitness value is better than the its best fitness value in history
then Update *pbest*
if the fitness value attained a minimum criteria
then Stop particle p in the current *pbest* location
end
for each particle p**do**
begin
Identify the particle in the neighborhood with the best fitness value so far as the *lbest*
Assign its index to the variable l
if particle p is not stopped
then Calculate particle velocity according equation (a)
Update particle position according equation (b)
end
end

When a particle finds a feasible solution (its fitness value attains minimum criteria) it is obvious there is no need to continue 'flying' and thus the particle can stop at that *pBest* location. But the particle will continue to share its experience with its still 'flying' neighbors (particles belonging to the same sub-swarm).

4 Experiment Results

This section illusrates the various experiments performed using geometrical place problems. Results obtained by the INPSO are compared with the results obtained by the standard EA. The EA used in the experiments uses real encoding of solutions. Mutation and convex crossover are the genetic operators used. Parameters used by INPSO are given in Table 1 and parameters used by EA are given in Table 2.

Table 1. Parameter settings for INPSO

Parameter	**Value**
η_{x}	1.49445
η_{y}	1.49445
Sub-swarm size	4
V_{max}	0.1* X_{max}
inertia weight	[0.5+(Rnd/2.0)]

Both η_{x} and η_2 are set to 1.49445 [2] to make the search cover all surrounding regions which is centered at the *pBest* and *lBest*. A randomized inertia weight

Table 2. Parameter settings for EA

Parameter	**Value**
Sigma	1
Crossover probability	0.5
Mutation probability	0.7

is used, namely it is set to [0.5+(Rnd/2.0)] ([5]). V_{max} is set to 0.1* X_{max}. The value of $V_{max}$is usually chosen to be k*X_{max}, with 0.1≤ k≤1.0 [4].

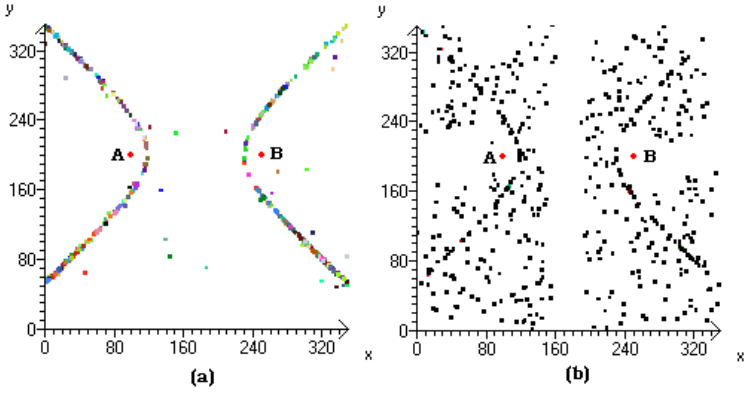
Population size is the same for both algorithms: 500 individuals and particles respectively.

4.1 Experiment 1

We consider the case in which the geometrical place of the points for which the difference (in absolute value) of the distances to two given points is equal to a given constant k. In a two dimensional space the geometrical place consists of hyperbole of focuses of the two given points. In a three dimensional space the geometrical place consists on the hyperboloid of focuses of the two given points. This problem can also be extended to higher dimensional spaces.

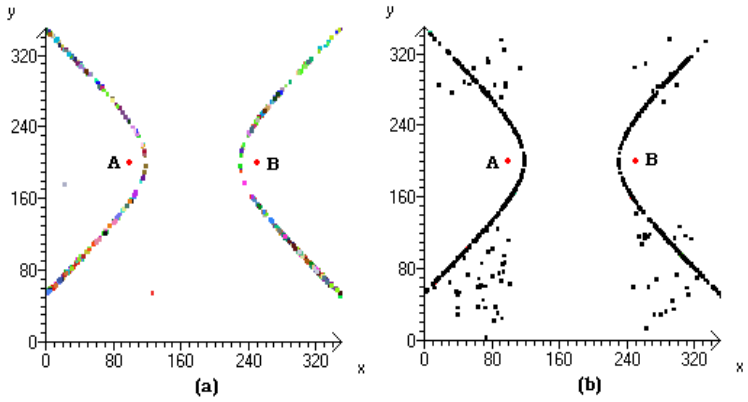
We consider the same population randomly generated for both algorithms. The populations obtained after 50 generations by INPSO and EA are depicted in Figure 1 (a) and Figure 1 (b) respectively. Population obtained after 200 generations by INPSO and EA is depicted in Figure 2 (a) and Figure 2(b) respectively.

Fig. 1. Population obtained after 50 generations. (a) population obtained by INPSO, (b) population obtained by EA.



As evident from these figures, INPSO algorithm has converged faster than EA. Even after 250 generations, there are some particles which did not con-

Fig. 2. Population obtained after 200 generations. (a) population obtained by INPSO, (b) population obtained by EA.



verge to the geometrical place. The EA individuals takes longer time (about 500 generations) but all the individuals finally converged to the geometrical place. Using the hybrid approach (INPSO for first 100 generations and EA after 100 generations) all solutions converged to the geometrical place in less than 250 generations. h

4.2 Experiment 2

In this experiment the geometrical place of the points M for which the product of the distances to two fixed points $F_1(-c, 0)$ and $F_2(c, 0)$ is equal to the constant a^2 is searched. This geometrical place is called *the oval of Cassiani*. The geometrical place depends on the values of a and c. Four cases can be envisaged and we will analyze two of the cases.

Case $a < c$.

Let us consider, for instance, a = 150 and c = 151. Population obtained after 50 generations by INPSO and EA is depicted in Figure 3 (a) and (b) respectively.

Population obtained after 250 generations by INPSO, EA and INPSO combined with EA is depicted in Figure 4.

The INPSO algorithm obtained the solutions (all particles will converge to the geometrical place) after 1000 generations while the individuals of EA converged after 800 generations. But the combined INPSO and EA obtained the solutions within 250 generations.

Case $a > c$

When a = 200 and c = 150, population obtained after 50 generations by INPSO and EA is depicted in Figure 5 (a) and (b) respectively. Population obtained after 250 generations by INPSO, EA and INPSO combined with EA is depicted in Figure 6.

As evident from these experiments, INPSO is very fast compared to EA. For more difficult problems there can be situations where some particles could never

Fig. 3. Population obtained after 50 generations by INPSO (Figure 5 (a)) and by EA (Figure 5 (b)).

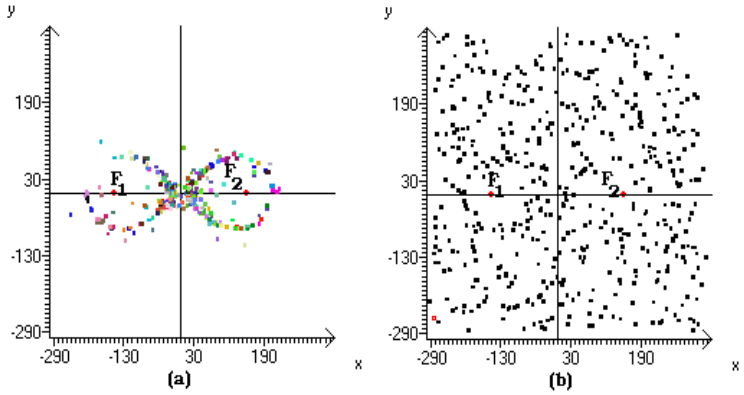


Fig. 4. Population obtained after 250 generations by INPSO (Figure 6(a)), EA (Figure 6(b) and INPSO combined to EA (Figure 6(c)).

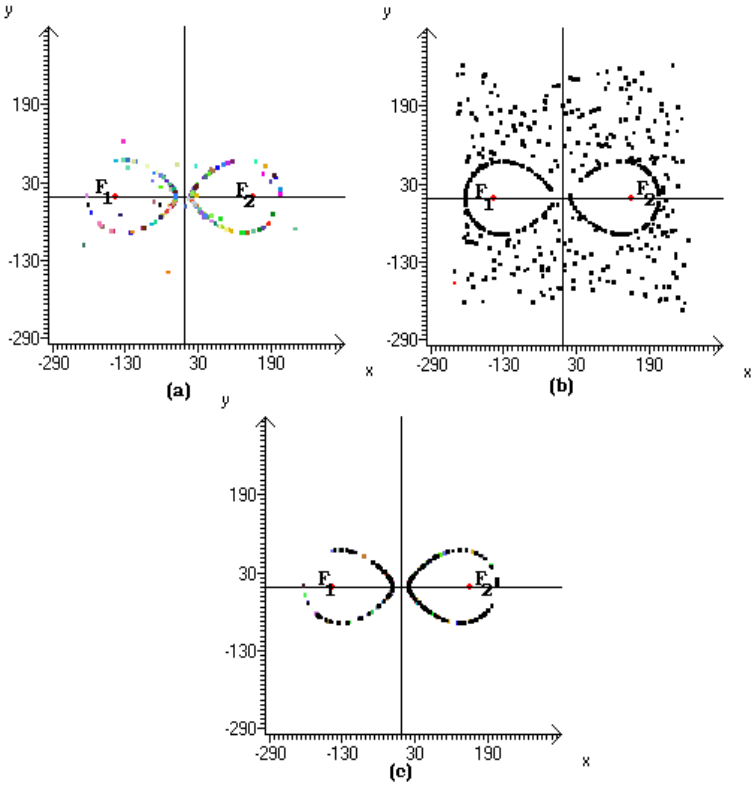


Fig. 5. Population obtained after 50 generations by INPSO (Figure 7 (a)) and by EA (Figure 7 (b)).

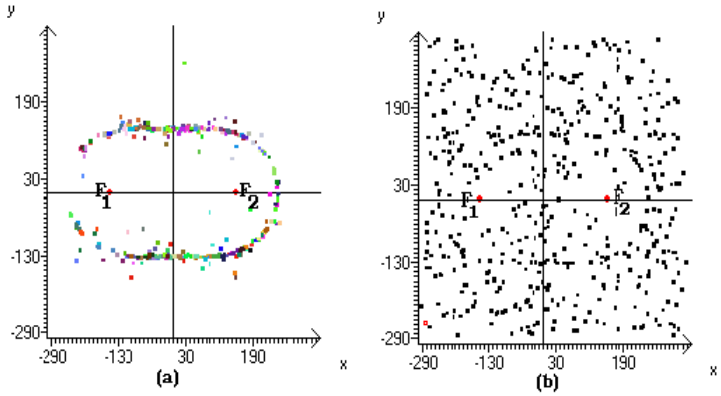
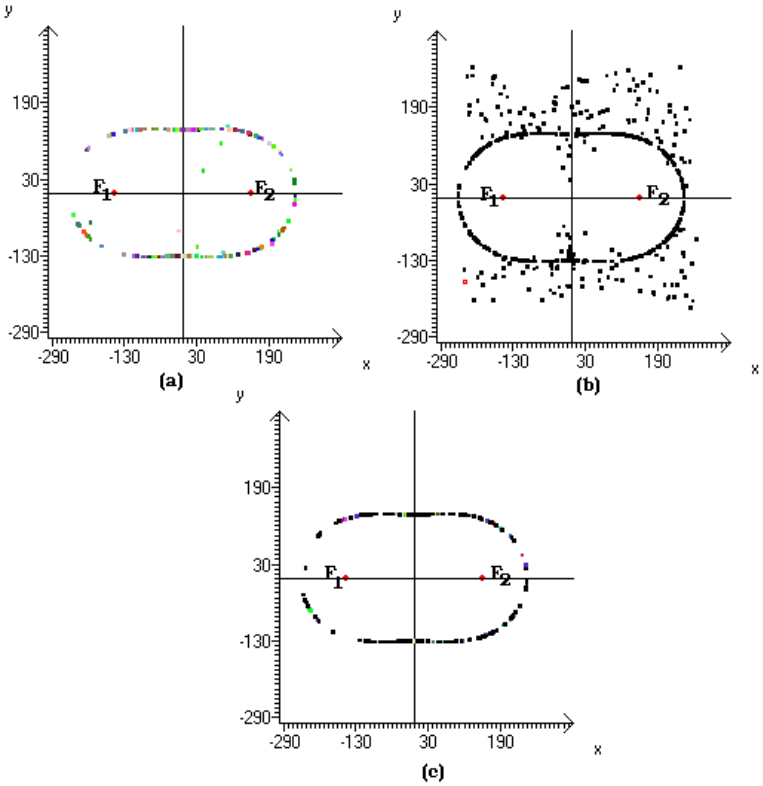


Fig. 6. Population obtained after 250 generations by INPSO (Figure 8(a)), EA (Figure 8(b) and INPSO combined to EA (Figure 8(c)).



converge to the geometrical place. EA is not very fast but with a good population size and Irsgurnumber of generations the performance could improve. Empirical results using the hybrid approach illustrate that the combination could play an important role in finding solutions with fewer number of generations when compared to the individual approaches.

5 Acknowledgements

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the Chung-Ang University HNRC-ITRC (Home Network Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

6 Conclusions

In this paper a new variant of PSO called Independent Neighborhood Particle Swarm Optimization (INPSO) is proposed and used to solve geometrical place problems. INPSO uses independent sub-swarm which evolves independently with respect to the entire population. The PSO rules are applied for each sub-swarm.

Performance of INPSO is compared with the classical Evolutionary Algorithms (EA). INPSO is very fast compared to EA. But, for difficult problems, there can be some particles (a sub-swarm for instance) which could never converge. Taking into account these issues, we proposed a hybrid approach involving INPSO and EA. The key advantages of the hybrid approach are in making use of the fast convergence property of INPSO and EA's definite convergence (guaranteed solution). After a given number of INPSO generations (in our case after 100 generations) EA method was deployed. The combination obtains the solutions very fast and all individuals converged to the geometrical place with fewer iterations.

References

1. Bergh, F.D. and Engelbrecht, A. A Cooperative Approach to Particle Swarm Optimization, IEEE Transaction on Evolutionary Computation, 8(3): pp. 225-239, 2004.
2. Clerc, M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), pp. 1951-1957, 1999.
3. Eberhart, R. C. and Kennedy, J. A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995.
4. Eberhart, R. C., Simpson, P. K., and Dobbins, R. W. Computational Intelligence PC Tools. Boston, MA: Academic Press Professional, 1996.
5. Eberhart, R. C. and Shi, Y. Particle swarm optimization: developments, applications and resources. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001), Seoul, Korea. 2001

6. Grosan, C. Solving geometrical place problems by using Evolutionary Algorithms. World Computer Congress, Student Forum, M. Kaaniche (Ed.), Toulouse, France, pp. 365-375, 2004
7. Hu, X., Shi Y., and Eberhart, R.C. Recent Advences in Particle Swarm, Congress on evolutionary Computation, Portland, Oregon, June 19-23, pp. 90-97, 2004
8. Kennedy, J. and Eberhart, R. C. Particle swarm optimization.Proceedings of IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, Vol.IV, pp.1942-1948, 1995.
9. Kennedy, J. Minds and cultures:Particle swarm implications. Socially Intelligent Agents: Papers from the 1997 AAAI Fall Symposium. Technical Report FS-97-02, Menlo Park, CA: AAAI Press, 67-72, 1997.
10. Kennedy, J. The Behavior of Particles, 7th Annual Conference on Evolutionary Programming, San Diego, USA, 1998.
11. Krohling, R.A., Hoffmann, F. and Coelho, L.S. Co-evolutionary Particle Swarm Optimization for Min-Max Problems using Gaussian Distribution, In Proceedings of the Congress on Evolutionary Computation (CEC'2004), IEEE Press, Vol. 1, pp. 959-964, 2004.
12. Shi, Y., and Eberhart, R. C. Empirical study of particle swarm optimization. Proceedings of the 1999 Congress on Evolutionary Computation, 1945-1950. Piscataway, NJ: IEEE Service Center, 1999.
13. Shi, Y., and Eberhart, R. C. Parameter selection in particle swarm optimization, Proceedings of the 1998 Annual Conference on Evolutionary Computation, 1998.
14. Shi, Y. and Eberhart, R. C. A modified particle swarm optimizer. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998), Piscataway, NJ. pp. 69-73, 1998