Performance Tuning of Evolutionary Algorithms Using Particle Sub Swarms

Crina Grosan
Department of Computer Science
Faculty of Mathematics and Computer Science
Babes-Bolyai University, Kogalniceanu 1, Cluj-Napoca 3400, Romania
cgrosan@cs.ubbcluj.ro

Ajith Abraham
School of Computer Science and Engineering
Chung-Ang University, Seoul 156-756, Korea
ajith.abraham@ieee.org

Monica Nicoara
Department of Computer Science
Faculty of Mathematics and Computer Science
Babes-Bolyai University, Kogalniceanu 1, Cluj-Napoca 3400, Romania
monica.nicoara@gmail.com

Abstract

Particle Swarm Optimization (PSO) technique proved its ability to deal with very complicated optimization and search problems. This paper proposes a new particle swarm variant which deals with sub-populations. This algorithm is applied for solving the well known class of mathematical problems: geometrical place problems (also known as locus problems). Finding the geometrical place can be sometimes a hard task and in almost all situations the geometrical place consists in more than one single point. The performance of the sub-swarm based PSO method is compared with Evolutionary Algorithms. The main advantage of the PSO technique is its speed of convergence. Also, we propose a hybrid algorithm by combining PSO and EA. This combination is able to detect the geometrical place very fast for difficult problems for which EA's need more time and PSO technique even with sub-populations could not find the geometrical place.

1 Introduction

Evolutionary Algorithms (EA) use a population of potential solutions (points) of the search space. These solutions (initially random 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.

A relatively new evolutionary paradigm, called Particle Swarm Optimization (PSO) had been discovered through simplified social model simulation (is motivated from the simulation of social behavior). There are some similarities between PSO and Evolutionary Algorithms ([1]):

- both techniques use a population (which is called *swarm* in 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 (swarm behavior in the case of Particle Swarm technique and ideas from human genetics in the case of Evolutionary Algorithms).

In what follows, we illustrate both techniques for solving the geometrical place problems. 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 searched. In many situations, the searched geometrical place consists in more than one point (solution). PSO and EA are fit for this problem, mainly due to their ability to deal with a population of solutions in 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 of the existing PSO variants can be found in [5], [6], [7].

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

The paper is structured as follows: Section 2 presents some general notions about the locus problems. Section 3 presents PSO technique. Section 4 briefly describes the new proposed sub-swarm PSO technique. The hybrid EA and PSO approach is described in Section 5. In Section 6 some experiments considering different test problems are performed. A set of conclusions and remarks are presented towards the end.

2 Geometrical place problem

The *geometrical place* (or *locus*) can be defined as the set of the points which have the same (or a common) property. By following a traditional mathematical rule, the geometrical place problems are solved in two steps:

- First, we are trying to determine (using intuition) this set
- Second, we have to prove that this set is the geometrical place we are looking for.

Example:

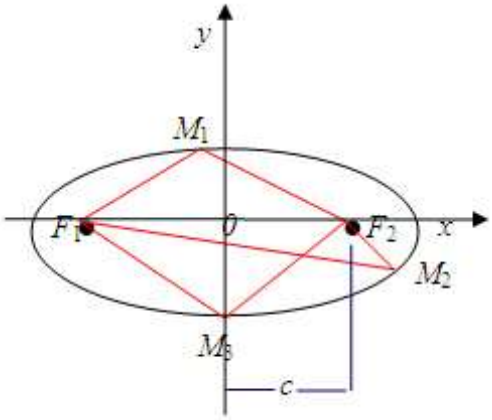
An ellipse is a curve that has the locus (or geometrical place) of all points in the plane, the sum of whose distances r_1 and r_2 from two fixed points F_1 and F_2 (the foci) separated by a distance equal to $2c$ is a given positive constant (Figure 1). The points M_1,M_2, M_3 in Figure 1 are points from the geometrical place.

3 Particle Swarm Optimization

Like other evolutionary computation techniques, PSO is a population-based search algorithm and is initialized with a population of random solutions, called particles ([5]).

Unlike in the other evolutionary computation 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.

Figure 1. Example of an ellipse locus



The PSO was first designed to simulate birds seeking food which is defined as a 'cornfield vector' [6]. Assume the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. The birds do not know where the food is. But they know how far the food is and their peers' positions. So what's the best strategy to find the food? An effective strategy is to follow the bird which is nearest to the food.

PSO learns from the scenario and uses it to solve optimization problems. In PSO, each single solution is like a 'bird' in the search space, which is called 'particle'. All particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles.

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 [8] (brought in for balancing the global and the local search), these formulas are:

$v_{id} = \omega * v_{id} + \eta_{\text{x}} * rand() * (p_{id} - x_{id}) + \eta_2 * Rand() * (p_{gd} - x_{id})$ (a)

$x_{id} = x_{id} + v_{id}$ (b)

where rand() and Rand() are two random numbers independently generated in the range [0,1] and $\eta 1$ and $\eta 2$ are two learning factors which control the influence of the social and cognitive components.

In equation (a) 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 $[-V_{max,}\ V_{max}]$which serves as a constraint to control the global exploration ability of particle swarm. Thus, is reduced the likelihood of particles leaving the search space). Note that this 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.

4 Independent Neighborhoods Particle Swarm Optimization

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'. The dimension of each neighborhood (sub-swarms) is the same for all considered sub-swarms.

The reason for not choosing overlapping neighborhoods is that in the case of the studied problem (solving geometrical place problems) the solution consists of a set of points and not a single point only. Instead of searching for one single point as the solution it is need of searching for a set of points having the same property but independent one from another.

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 (which is given by the position of the best particle in the swarm). 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 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**

{ **for** each particle p **do**

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

for each particle p **do**

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)

}

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).

5 Hybrid PSO and EA approach

In [3], a simple Evolutionary Algorithm called Geometrical Place Evolutionary Algorithm (GPEA) for dealing with loci problems is proposed. Analyzing results obtained for some test problems by INPSO and GPEA independently considered we concluded that GPEA always converges to the solution, but compared to INPSO it is time consuming. INPSO approach converges faster when compared to GPEA (less number of iterations). But, for some difficult problems we found that some sub-swarms of INPSO could not converge (even if the number of generations is increased drastically)([4]). Taking these into account, we propose a hybrid approach between INPSO and GPEA. First, we will exploit INPSO's ability to converge very fast and then, we will use GPEA's advantage to always converge to the solution. After 100 generations of INPSO approach we will switch to GPEA (the solutions which failed earlier using INPSO is expected to converge using additional iterations of GPEA). GPEA chromosomes consist of the existing INPSO particles, with their current location.

6 Experiment Results

We performed some numerical experiments by considering two geometrical place problems. GPEA 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 GPEA are given in Table 2.

Table 1. Parameters used in experiments by INPSO

Parameter	Value
Swarm size	500
sub-swarm size	4
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 according to the work by Clerc ([2]). The obvious reason is it will make the search cover all surrounding regions which is centered at the *pBest* and *lBest*.

A randomized inertia weight is used, namely it is set to [0.5+(Rnd/2.0)], which is selected in the spirit of Clerc's constriction factor.

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$\leq$ k$\leq$1.0.

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

Table 2. Parameters used in experiments by GPEA

Parameter	Value
Population size	500
Sigma	1
Crossover probability	0.5
Mutation probability	0.9

6.1 Experiment 1

Geometrical place of the points M for which the fraction to two given points A and B is constant and equal to k, $k \neq 1$. Geometrical place consists of a circle (having different size for different values of k). The problem is known as *"Apollonius's circle"*. We will analyze the case in which k is = 2.

Results obtained by INPSO, GPEA and the hybrid algorithms are depicted in Figure 2. Figure 2 (left) refers to INPSO, Figure 2 (middle) refers to GPEA and Figure 2 (right) to the hybrid INPSO - GPEA approach. N denotes the number of generations.

As we can see from Figure 2(right), INPSO particles converged (almost all of them) in 100 generations. Finally, after 200 generations all particles converged while GPEA needs 1500 generations for all the individuals to converge. For the hybrid INPSO-GPEA (first INPSO for 100 iterations and thereafter GPEA) approach, all the particles (individuals) converged after 170 generations. Number of particles that did not converge in 100 generations by applying INPSO is 5 (from a population of 500 individuals). If the swarm size is 100 particles, two of them will not converge in 100 generations. From a swarm of 200 individuals, 4 particles will fail to find the geometrical place. Same number of particles (4) will not converge from a swarm of 300 particles and 8 from a swarm of 400 particles respectively. We use a greater number of particles so that the geometrical place could be clearly visualized.

6.2 Experiment 2

Geometrical place of the points M for which the sum of the distances to two fixed points representing the centers of two circles having the rays R_1 and R_2 respectively is equal to the product of the rays ($R_1 * R_2$).

The geometrical place is an ellipse having the foci centers of the two circles and consisting on those points for which the sum of distances to the foci is equal to $R_1 * R_2$.

Depending on the circles position, the shape of the ellipse is different. Two situations are analyzed. When the circles centers are same, then the geometrical place is a circle itself (see Figure 3).

Figure 3. Geometrical place when circle's centers are identical

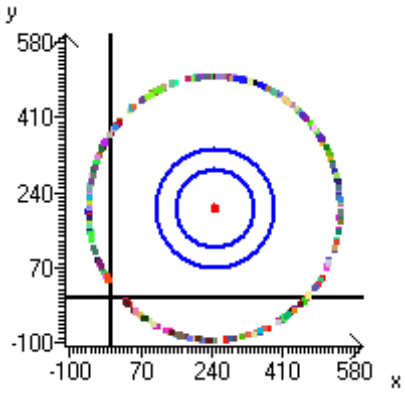
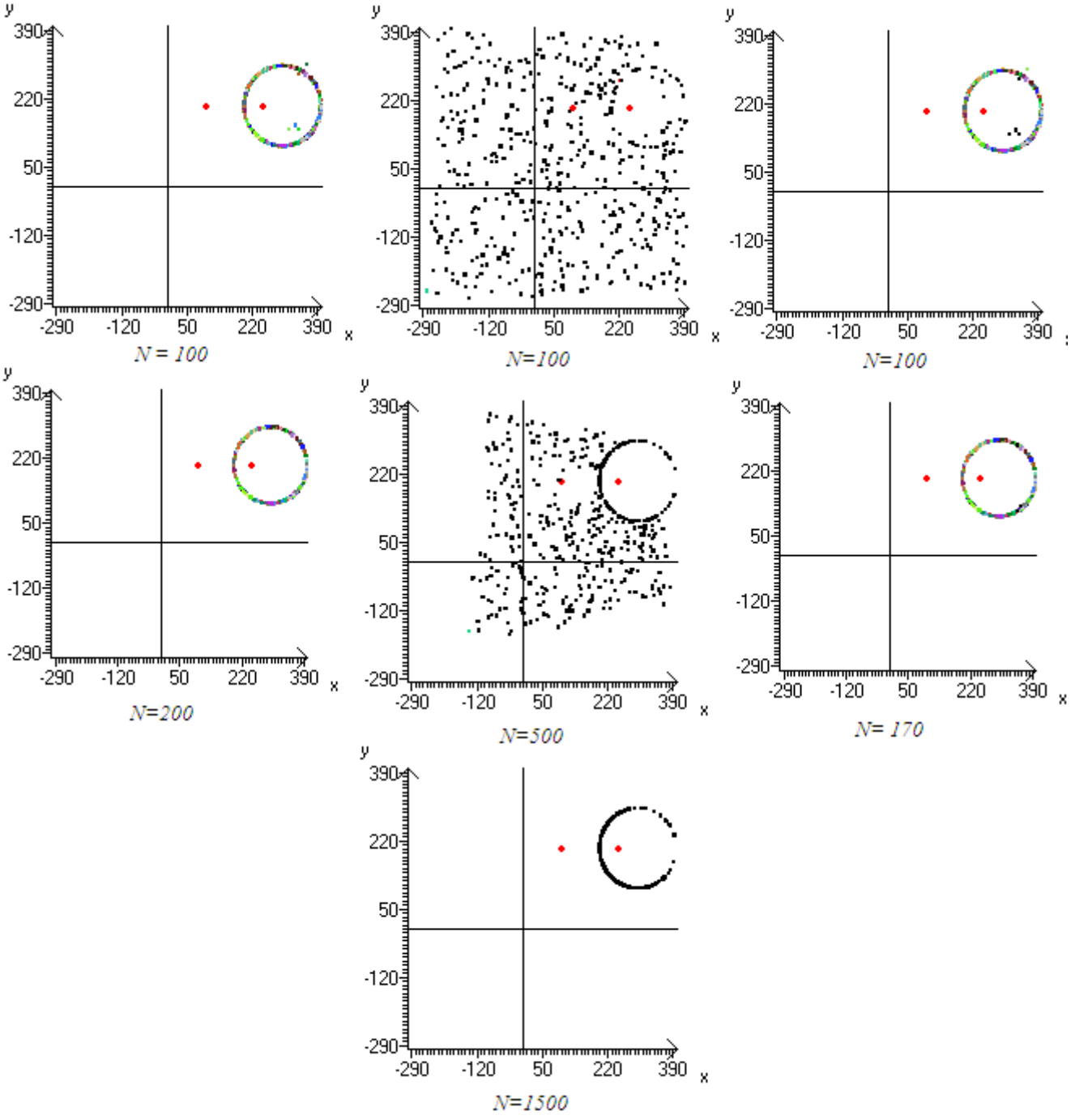


Figure 2. Apollonius's circle: INPSO (left), GPEA (middle), Hybrid INPSO-GPEA (right)



Case 1.

In this situation we consider the circles as having the same value corresponding to the Ox axis.

Results obtained by INPSO, GPEA and the hybrid algorithms are depicted in Figure 4. Figure 4 (left) refers to INPSO, Figure 4 (middle) refers to GPEA and Figure 4 (right) to the hybrid INPSO - GPEA approach.

As we can see from Figure 4 (right), INPSO particles converged (almost all of them) in 200 generations. Finally, after 300 generations all particles converged while GPEA needs 900 generations for all individuals to converge. For the hybrid INPSO-GPEA (first INPSO for 100 iterations and thereafter GPEA) approach, all the particles (individuals) converged after 250 generations. Number of particles that did not converge in 100 generations by applying INPSO is 7 (from the entire population of 500 particles). If the swarm size is 100 particles, then one particle will not converge in 100 generations. From a swarm of 200 individuals, 4 particles will fail to find the geometrical place. The number of particles which will not converge from a swarm of 300 particles is 7 and 4 for 400 particles respectively.

Case 2.

In this case we consider circles as having centers situated on different levels.

Results obtained by INPSO, GPEA and the hybrid algorithms are depicted in Figure 5. Figure 5 (right) refers to INPSO, Figure 5 (middle) refers to GPEA and Figure 5 (right) to the hybrid INPSO - GPEA approach. We denoted by N the number of generations.

As depicted in Figure 5 (right) INPSO converged much faster when compared to GPEA. In about 100 generations almost all particles converged. After 200 generations all particles converged while GPEA required about 800 generations for all individuals to converge. The hybrid INPSO-GPEA approach converged in a similar way as INPSO (in 170 generations). Number of particles that did not converge within 100 generations by applying INPSO is 9 (out from a population of 500 particles). If swarm size is 100 particles, then all of them converged in 100 generations. Same situation for a swarm of 200 particles. From a swarm of 300 individuals, only a single particle failed in finding the geometrical place. The number of particles that will not converge is 8 for a swarm of 300 particles and 6 for 400 particles respectively.

7 Conclusions

In this paper a new variant of PSO is proposed. The algorithm (called Independent Neighborhood Particle Swarm Optimization (INPSO)) seems to work well for the considered geometrical place problems (and, generally, with problems for which the searched solution consists in more than one point). INPSO uses independent sub-swarms which evolves independently of the entire population. The PSO rules are applied for each sub-swarm.

Some numerical experiments are performed. INPSO is compared with 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 will never converge to the required solution. Taking into account of these problems, INPSO and EA were hybridized. PSO is very fast and EA's always converges to the solution. First, we applied INPSO and after a given number of generations (in our case after 100 generations) we applied EA. The combination obtains the solution very fast and all individuals converged to the geometrical place.

References

[1] Angeline, P. Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Difference, The 7th Annual Conference on Evolutionary Programming, San Diego,USA, 1998.

[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] Grosan C., Solving geometrical place problems by using Evolutionary Algorithms. World Computer Congress, M. Kaaniche (Ed.), Toulouse, France, pp. 365-375, 2004

[4] Crina Grosan, Ajith Abraham and Monica Nicoara, Search Optimization Using Hybrid Particle Sub-Swarms and Evolutionary Algorithms, International Journal of Simulation Systems, Science & Technology, UK, Volume 6, Nos. 10 and 11, pp. 60-79, 2005.

[5] 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

[6] 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.

[7] 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.

[8] 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

Figure 4. Experiment 2 with Case 1: INPSO (left), GPEA (middle), Hybrid INPSO-GPEA (right)

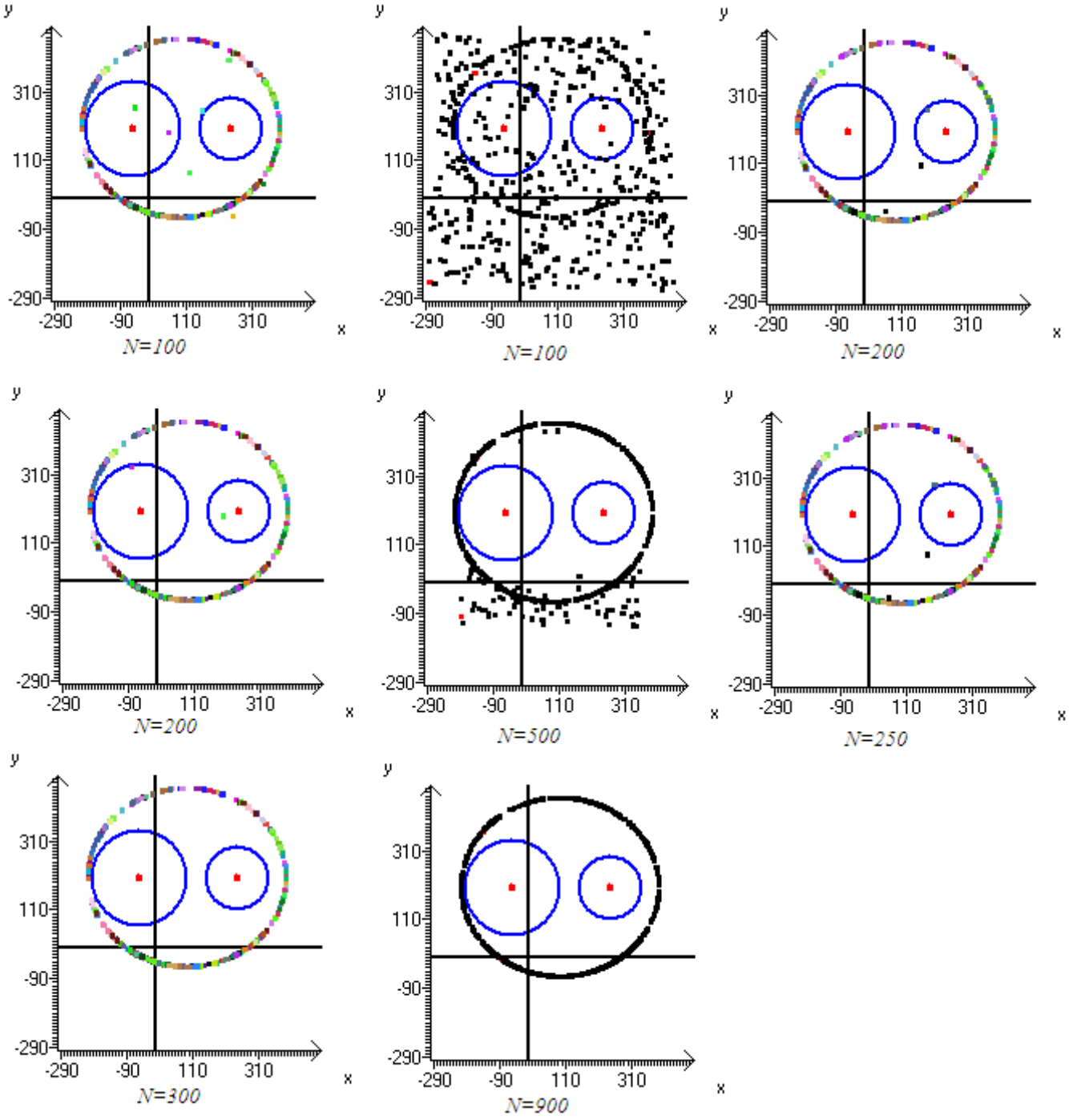


Figure 5. Experiment 2 with Case 2: INPSO (left), GPEA (middle), Hybrid INPSO-GPEA (right)