SEARCH OPTIMIZATION USING HYBRID PARTICLE SUB-SWARMS AND EVOLUTIONARY ALGORITHMS

CRINA GROSAN[1], AJITH ABRAHAM[2], MONICA NICOARA[1]

[1]*Department of Computer Science*
Babes-Bolyai University, Cluj-Napoca, 3400, Romania
[2]*IITA Professorship Program, School of Computer Science and Engineering*
Chung-Ang University, Seoul 156-756, Korea

Abstract: Particle Swarm Optimization (PSO) technique 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 variant of the PSO technique named Independent Neighborhoods Particle Swarm Optimization (INPSO) dealing with sub-swarms for solving the well known geometrical place problems. Finding the geometrical place can be sometimes a hard task and in almost all situations the geometrical place consists of more than one single point. Taking all these into account, the INPSO algorithm is very appealing for solving this class of problems. The performance of the INPSO approach is compared with Geometrical Place Evolutionary Algorithms (GPEA). The main advantage of the INPSO technique is its speed of convergence (finding quick solutions). To enhance the performance of the INPSO approach, a hybrid algorithm combining INPSO and GPEA is also proposed in this paper. The developed hybrid combination is able to detect the geometrical place much faster even for difficult problems for which the direct GPEA approach required more time and the INPSO (even with few sub-swarms) approach failed in finding all the geometrical place points (solutions).

1. INTRODUCTION

Many group-living vertebrates exhibit complex, and coordinated, spatio-temporal patterns, from the motion of fish and birds, to migrating herds of social ungulates and patterns of traffic flow in human crowds. The common property of these apparently unrelated biological phenomena, is that of inter-individual interaction, by which individuals can influence the behavior of other group members. Self-organization theory suggests that much of complex group behavior may be coordinated by relatively simple interactions among the members of the group. Following this theory, in 1995, Kennedy and Eberhart developed some algorithms that modeled the "flocking behavior" seen in many species of birds (Eberhart and Kennedy, 1995). Different from the evolution-motivated computation techniques, a relatively new evolutionary paradigm, called Particle Swarm Optimization (PSO) had been discovered through simplified social model simulation.

Evolutionary Computation (EC) techniques 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. Evolutionary computation includes Genetic algorithms (GA) [Goldberg, 1989], Evolution Strategies (ES) [Rechenberg, 1994], Genetic Programming (GP) [Koza, 1992] and Evolutionary Programming (EP) [Fogel, 1994].

PSO shares many similarities with evolutionary computation techniques such as genetic algorithms. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. There are some similarities between PSO and Evolutionary Algorithms [Angeline, 1998]:

- both techniques use a population (which is called *swarm* in the case of PSO) of solutions from the search space which are initially random generated;
- do not require auxiliary knowledge of the problem;
- 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);
- can provide more than one solution at the end of search process and the final choice is left to the user.

Even then, there are still many differences between these two techniques. In what follow, we will apply

both techniques for solving a well known class of search problems: 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). That's the main reason we think the evolutionary techniques (particularly Evolutionary Algorithms) and Particle Swarm 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 PSO technique (INPSO) which is based on the basic PSO algorithm proposed by Eberhart and Kennedy in 1995 (Eberhart and Kennedy, 1995). Some related work with the existing PSO variants can be found in [Eberhart and Kennedy, 1995; Eberhart et al., 1996; Eberhart and Shi, 2001; Kennedy and Eberhart, 1995; Kennedy, 1997a; Kennedy, 1997b; Kennedy, 1998a; Kennedy, 1998b; Shi and Eberhart, 1998a; Shi and Eberhart, 1998b; Shi and Eberhart, 1999].

Geometrical Place problems have already been investigated using Evolutionary Algorithms (GPEA) (Grosan, 2004, Grosan et al., 2005a). The main scope of our paper is to perform a comparison between INPSO and GPEA and to exploit the weakness/strength of each of them. Finally, taking into account the results, we propose a hybrid algorithm combining INPSO and GPEA which seems to perform better in complicated situations than each of these techniques applied separately (Grosan et al., 2005b). The paper is structured as follows: Section 2 presents the fundamentals of PSO technique. Section 3 briefly describes the INPSO technique proposed in this paper. The general evolutionary algorithm is described in Section 4 and the proposed hybrid INPSO-GPEA technique is presented 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. PARTICLE SWARM OPTIMIZATION TECHNIQUE

Like other evolutionary computation techniques, PSO is a population-based search algorithm and is initialized with a population of random solutions, called particles [Hu et al., 2004]. 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. The PSO was first designed to simulate birds seeking food which is defined as a 'cornfield vector' [Kennedy and Eberhart, 1995].

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 the 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. (The particles fly through the problem space by following the particles with the best solutions so far). 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 $X_i = (x_{i1}, x_{i2},..., x_{iD})$. At each generation, each particle is updated by the 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 $P_i=(p_{i1}, p_{i2},..., p_{iD})$. 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 (Shi and Eberhart, 1998) (brought in for balancing the global and the local search), these formulas are:

$$v_{id} = \omega * v_{id} + \eta_1 * rand() * (p_{id} - x_{id}) + \eta_2 * Rand() * (p_{gd} - x_{id}) \quad (a)$$

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

where *rand()* and *Rand()* are two random numbers independently generated in the range [0,1] and η_1 and η_2 are two learning factors which control the influence of the social and cognitive components. In (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, the likelihood of particles

leaving the search space is reduced. 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. The main PSO algorithm as described by Pomeroy [Pomeroy, 2003] is as follows:

```
/* set up particles' next location */
 for each particle p do {
  for d = 1 to dimensions do {
    p.next[d] = random(...)
    p.velocity[d] = random(deltaMin, deltaMax)
  }
  p.bestSoFar = initialFitness
}

/* set particles' neighbors */
for each particle p do {
  for n = 1 to numberOfNeighbors do {
    p.neighbor[n] = getNeighbor(p, n)
  }
}

/* run Particle Swarm Optimizer */
while iterations <= maxIterations do {
  /* Make the "next locations" current and then */
  /* test their fitness. */
  for each particle p do {
    for d = 1 to dimensions do {
      p.current[d] = p.next[d]
    }
    fitness = test(p)
    if fitness > p.bestSoFar then do {
      p.bestSoFar = fitness
      for d = 1 to dimensions do {
        p.best[d] = p.current[d]
      }
    }

    if fitness = targetFitness then do {
      ... /* e.g., write out solution and quit */
    }
  } /* end of: for each particle p */

  for each particle p do {
    n = getNeighborWithBestFitness(p)
    for d = 1 to dimensions do {
      iFactor = iWeight * random(iMin, iMax)
      sFactor = sWeight * random(sMin, sMax)
      pDelta[d] = p.best[d] - p.current[d]
      nDelta[d] = n.best[d] - p.current[d]
      delta = (iFactor * pDelta[d]) + (sFactor *
nDelta[d])
      delta = p.velocity[d] + delta
      p.velocity[d] = constrict(delta)
      p.next[d] = p.current[d] + p.velocity[d]
    }
  } /* end of: for each particle p */
} /* end of: while iterations <= maxIterations */
end /* end of main program */
```

```
/* Return neighbor n of particle p */
function getNeighbor(p, n) {
  ...
  return neighborParticle
}

/* Return particle in p's neighborhood */
/* with the best fitness */
function getNeighborWithBestFitness(p) {
  ...
  return neighborParticle
}

/* Limit the change in a particle's */
/* dimension value */
function constrict(delta) {
  if delta < deltaMin then
    return deltaMin
  else
  if delta > deltaMax then
    return deltaMax
  else
    return delta
}
```

3. PSO ADAPTED FOR GEOMETRICAL PLACE PROBLEMS

Since geometrical place problems usually suppose to find a set of points which accomplish a given condition, any evolutionary technique which deals with a population of solutions (a set of individuals) would be an ideal technique. The standard PSO algorithm uses a population of particles. In the search process, particles are supposed to follow the best particle from the population. This way, there is a probability that all particles will converge in the same point of the search space. But in the case of the geometrical place problems, the search place consists of more than one point. Consequently, we need to adapt the PSO technique so as to obtain multiple different points at the end of the search process. The idea of using sub-swarms or neighborhoods seems to be ideal for our problem. This way, we can obtain the maximum number of solutions which is equal to the number of sub-swarms assuming that all sub-swarms will converge to different points in the search space. The proposed PSO algorithm is similar to the classical ones which use neighborhoods [Brits et al., 2002; Peer et al., 2003] 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 is not only a single point. Instead of searching for one single point as the final solution it is required to search for a set of points having the same property but independent from one 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 the chances to obtain, finally, a greater number of different points. The way in which these sub-swarms work is depicted in Figure 1. We consider 5 sub-swarms. Each sub-swarm contains the same number of particles (6 nos). The geometrical place is supposed to be the eclipse in the middle. Particles in each sub-swarm will follow the best particle found so far in their own sub-swarm only.

The algorithm proposed is called Independent Neighborhoods Particle Swarm Optimization (INPSO). The main steps of the INPSO algorithm are described below:

INPSO algorithm

begin
for each particle *p* **do**
begin
Initialize with random position in the problem space
Initialize with random velocity in the problem space
endfor
while iteration ≤ max_iterations **do**
begin
for each particle *p* **do**
begin
Calculate fitness value
if the fitness value is better than its best fitness value in history
then
begin
Update *pbest*
if the fitness value attained a minimum criteria
then
Stop particle p in the current *pbest* location
endif
end
endif
endfor
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
begin
Calculate particle velocity according to equation (a)
Update particle position according to equation (b)
end
endfor
end
end.

When a particle finds a feasible solution (its fitness value attains minimum) 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. EVOLUTIONARY ALGORITHMS FOR GEOMETRICAL PLACE PROBEMS

Evolutionary Algorithms (EA) are stochastic search methods inspired from the metaphor of natural biological evolution. Evolutionary Algorithms were introduced in 1965 by John Holland [Holland, 1975]. These algorithms operate on a population of potential solutions applying the principle of survival of the fittest to produce better approximations to a solution. This population is initially randomly generated over the search space which is the definition domain. At each generation, operators borrowed from natural genetics such as selection, recombination, mutation, migration, inversion, reinsertion, etc. are applied to the individuals from the population. By applying genetic operators these individuals are evolved. Each individual from population is evaluated by using a quality (fitness) function. Using this quality the best individuals are selected at each generation. Many selection mechanisms have been implemented [Baker, 1985; Box, 1993; Goldberg 1989]. The selected

individuals are modified by applying crossover and/or mutation operator. Various forms of these operators can be found [Eshelman et al., 1989; Schaffer and Morishima, 1987; Spears and De Jong, 1991; Syswerda 1989]. In this way new solutions are obtained. Some of these new solutions can be better than the existing solutions. There are many modalities to accept the new solutions (also called offspring) in population. Some algorithms accept the new solution only if this solution is better than his parent (or parents). The elitist algorithms accept the new obtained solution in population. Some surveys in Evolutionary Algorithms and their applications can be found in [Box, 1993; Davis, 1991; Goldberg, 1989].

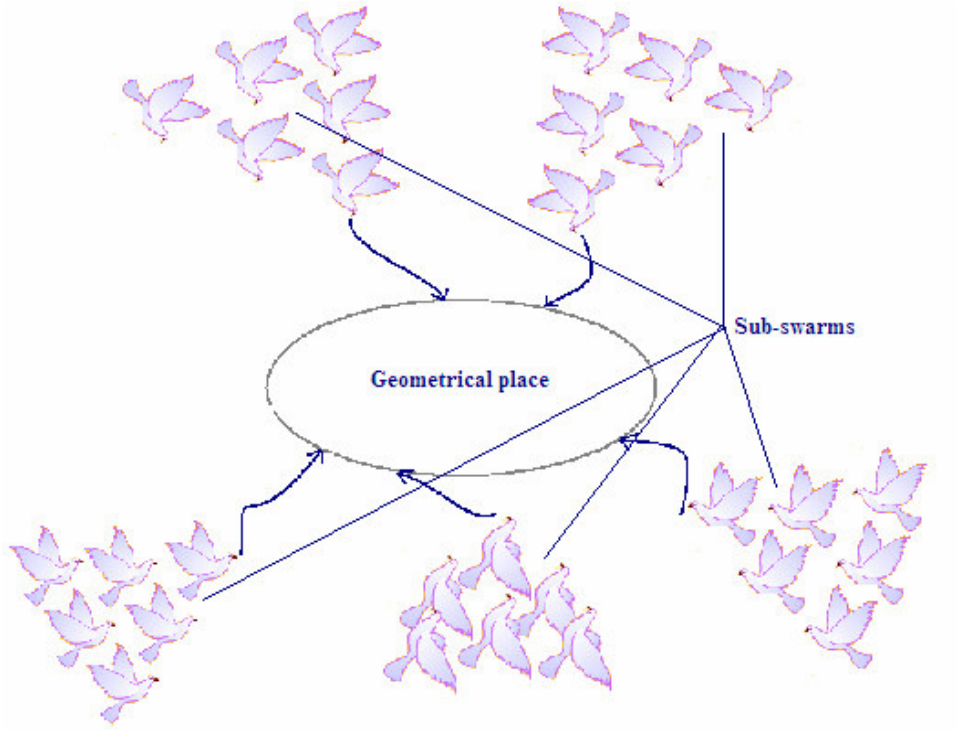

FIGURE 1. An example of 5 independent sub-swarms and each sub-swarm containing 6 particles.

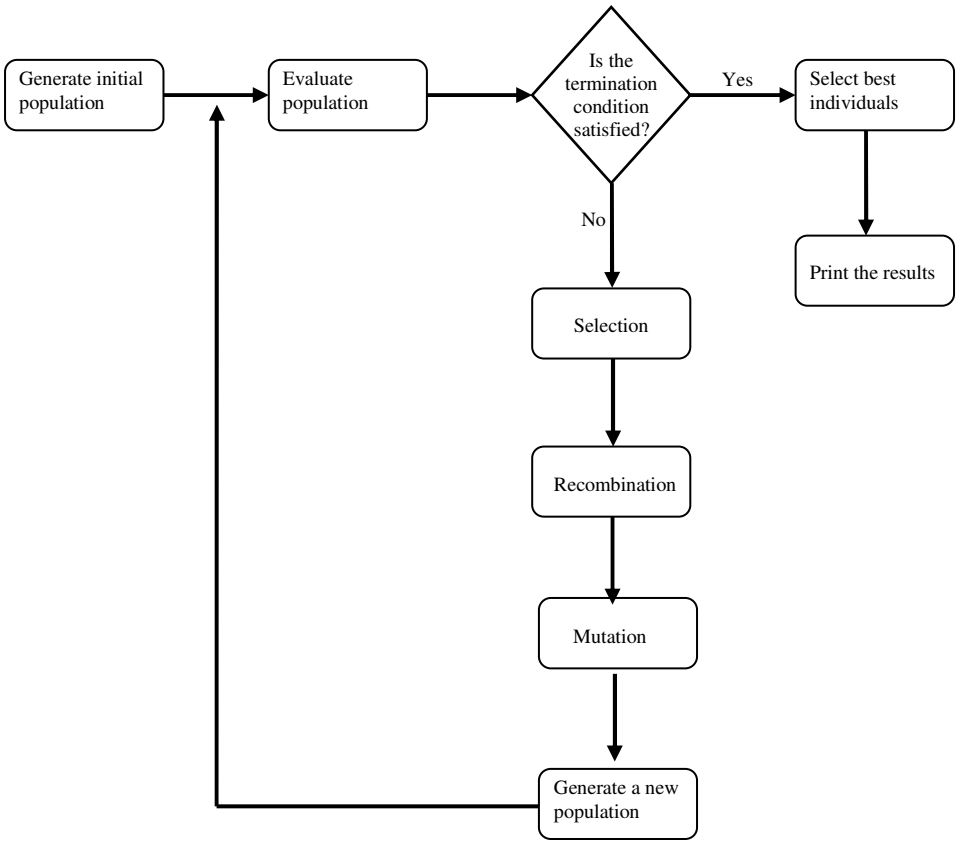

FIGURE 2. Flow chart of an evolutionary algorithm

Grosan [Grosan, 2004] proposed a simple evolutionary algorithm for dealing with geometrical place problems. This algorithm is called Geometrical Place Evolutionary Algorithms (GPEA). A real encoding of solutions is considered. Each individual is evaluated using an adequate fitness function. Binary selection is applied. A convex crossover operator [Goldberg, 1989] is applied over selected individuals. The best between parents and offspring enters the new population. Over this new obtained population mutation operator is applied. Individuals obtained after these steps will constitute the population of the next generation. These steps are repeated for a specified number of generations. GPEA is presented below.

GPEA description

begin
Set $t = 0$;
Randomly initialize the population $P(t)$ of size *popsize*;
Repeat
Evaluate population $P(t)$;
for $i := 0$ **to** *popsize* **do**
begin
Select two parents from $P(t)$;
Apply crossover over selected parents and obtain an offspring;
Mutate the best between parents and offspring and add it to population $P'(t)$.
end;
$t = t + 1$;
$P(t) = P'(t - 1)$;
Until t = *Number of generations*.
end.

5. HYBRID INPSO – GPEA APPROACH

Our preliminary experiments indicated that the 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). GPEA always converges to the solution, but compared to INPSO it is time consuming. Taking these into account, we propose a hybrid of 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. GPEA uses real representation of solutions. Gaussian

mutation and convex crossover are the only genetic operators used.

6. EXPERIMENT SETUP AND RESULTS

We performed 6 different experiments comparing INPSO performances with GPEA approach. For each problem considered in experiments we run each algorithm 20 times and we considered the worst results. INPSO and GPEA are applied for the same initial population. Experiment results are graphically illustrated in Figures 3-11. As we can see from the experiments results, INPSO converges faster when compared to GPEA (less number of iterations). But, for some difficult problems (Figures 8-11), it could be found that some sub-swarms could not converge (even if the number of generations are increased drastically). GPEA always converges to the solution, but compared to INPSO it is time consuming.

The values of the main parameters used by INPSO and GPEA in experiments are presented in Table1. For all the considered problems, the performance of the hybrid INPSO-GPEA technique is presented in Figures 3-11.

Parameter	Value	
	INPSO	**GPEA**
Population size	500	500
Sub-swarms size	4	-
η_x, η_y	1.49445	-
V_{max}	0.1 * X_{max}	-
Inertia weight	0.5 + Rnd/2.0	-
Sigma	-	1
Mutation probability	-	0.5
Crossover probability	-	0.7

Table 1. Parameters of INPSO and GPEA

Both η_x, η_y are set to 1.49445 according to the work by Clerc [Clerc, 1999]. 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 [Eberhart and Shi, 2001]. V_{max} is set to 0.1* X_{max}. The value of V_{max} is usually chosen to be k * X_{max}, with $0.1 \leq k \leq 1.0$ [Eberhart et al., 1996]. Each of the considered problems is described in the following sub-sections.

6.1 Problem 1: Circle

Geometrical place of the points M for which the distance (in absolute value) to a given point C is constant and equal to k. Geometrical place is the circle having as center the given point *C* and ray equal to *k*. Results obtained by INPSO, GPEA and the hybrid algorithms are depicted in Figure 3. Figure 3 (a) refers to INPSO, Figure 3 (b) refers to GPEA and Figure 3 (c) to the hybrid INPSO – GPEA approach. We denoted by *N* the number of generations.

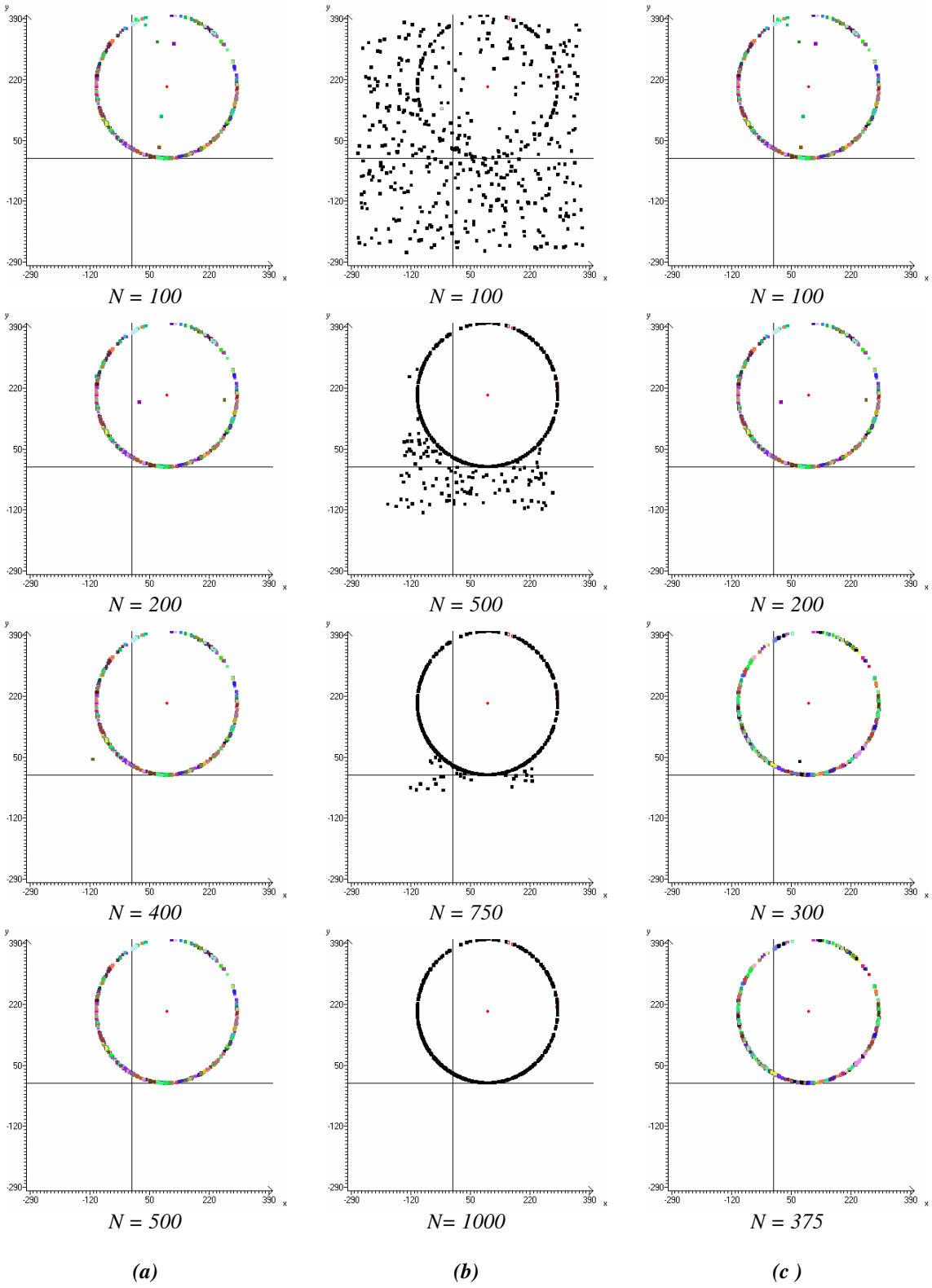

FIGURE 3. Circle (a) INPSO, (b) GPEA (c) Hybrid INPSO – GPEA

As we can see from Figure 3 (a), INPSO particles converge (almost all of them) in 200 generations. Finally, after 500 generations all particles converged while GPEA needs 1000 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 375 generations. Number of particles that did not converge in 100 generations by applying INPSO is 8 (from a population). If the swarm size is 100 particles, then all of them converge in 100 generations. From a swarm of 200

individuals, 2 particles will fail to find the geometrical place. The number of particles which will not converge increases with the population size. This will be 4 for a swarm of 300 particles and 5 for 400 particles respectively. We use a greater number of particles so that the geometrical place could be clearly visualized.

6.2. Problem 2: Ellipse

Geometrical place of the points M for which sum of distances to two given point F_1 and F_2 is constant and equal to a given number k. In this case, the geometrical place consists of the ellipse having as focuses points F_1 and F_2. Results obtained by INPSO, GPEA and hybrid INPSO – GPEA are depicted in Figure 4.

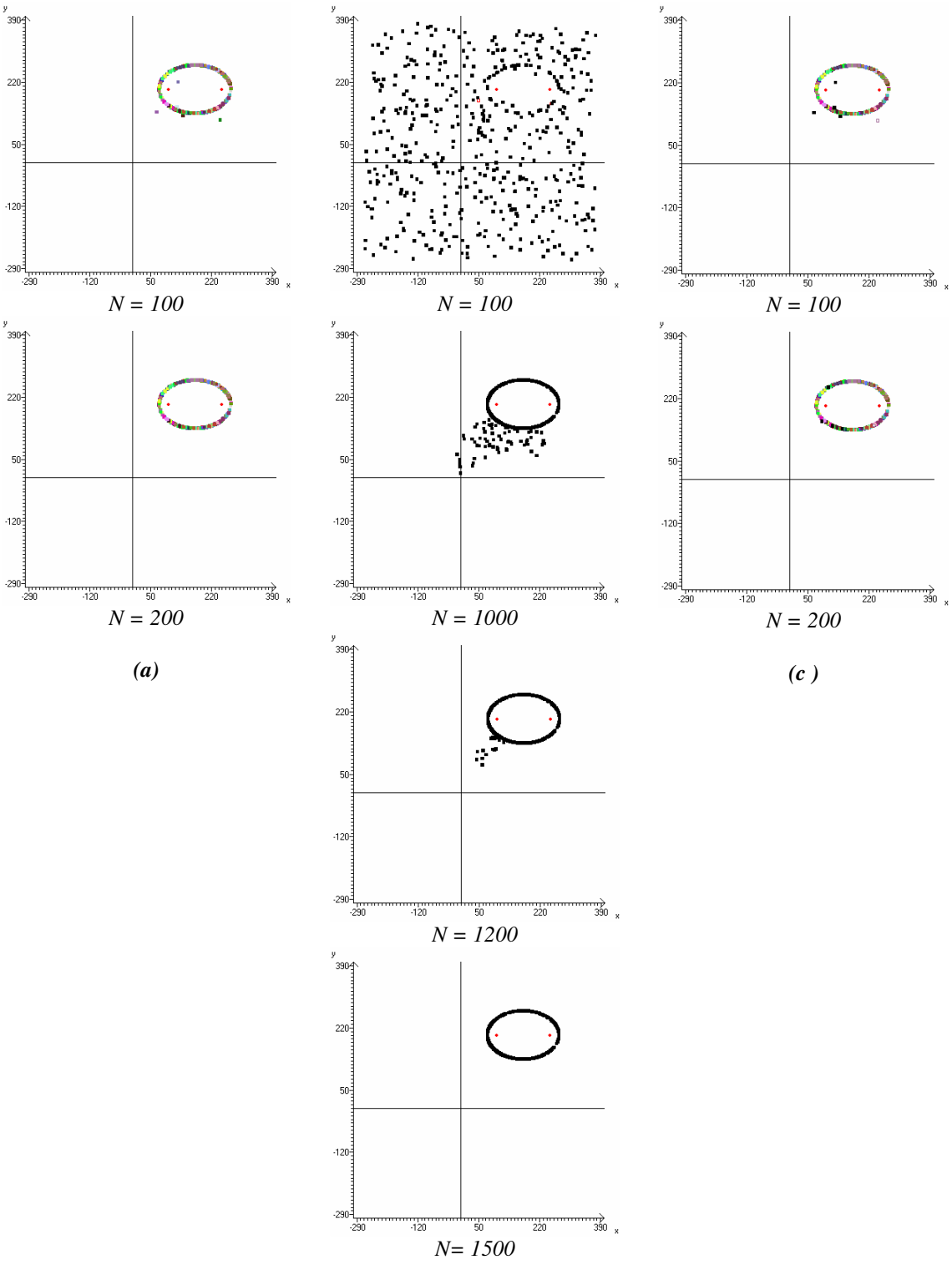

Figure 4. Ellipse (a) INPSO, (b) GPEA, (c) Hybrid INPSO – GPEA

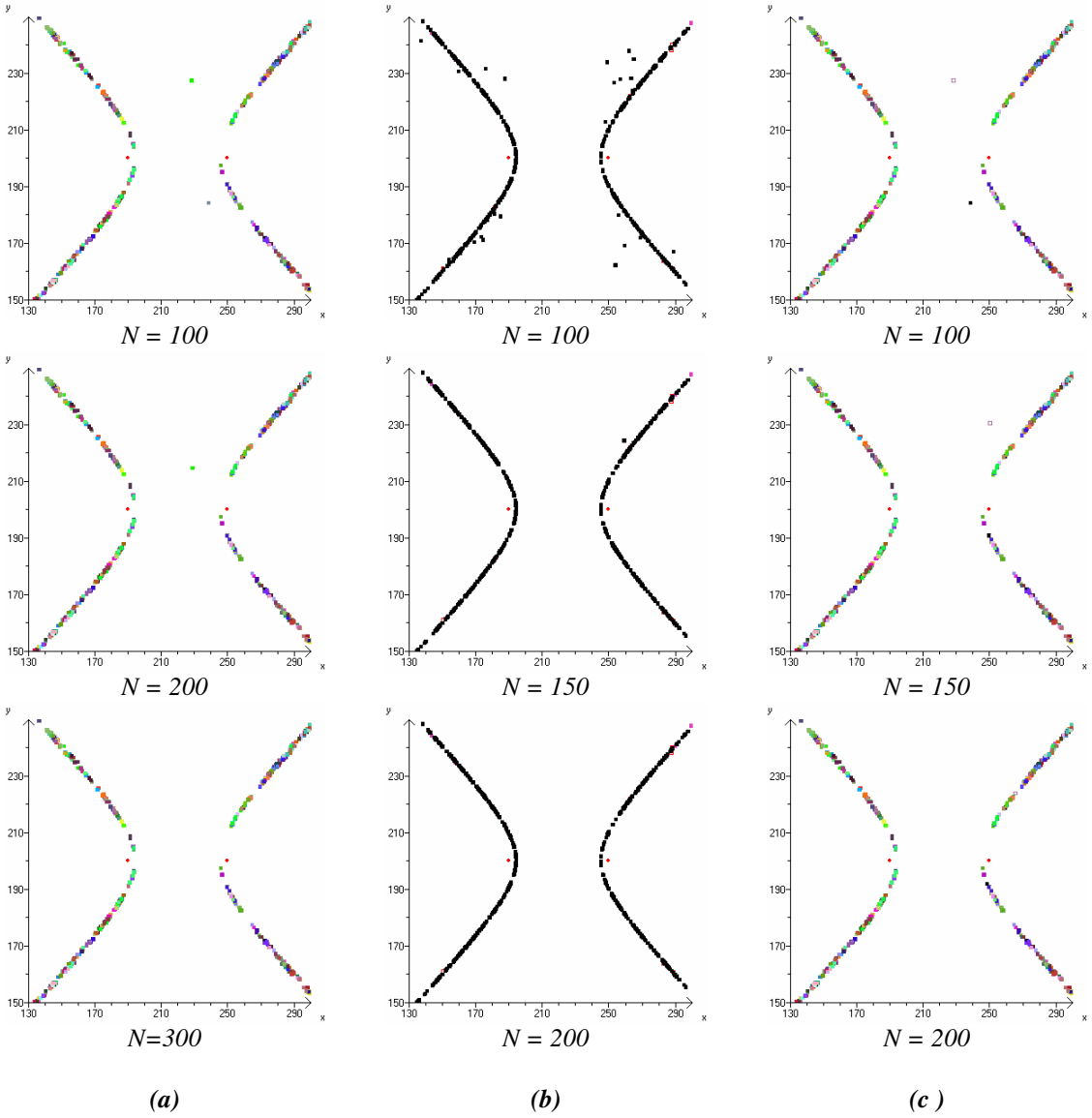

FIGURE 5. Hyperbola (a) INPSO, (b) GPEA, (c) Hybrid INPSO – GPEA

As depicted in Figure 4 (a) 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 1200 generations for all the particles to converge. The hybrid INPSO–GPEA approach converged in a similar way as INPSO. Number of particles that did not converge within 100 generations by applying INPSO is 7 (out from a population). If swarm size is 100 particles, then all of them converged in 100 generations. From a swarm of 200 individuals, only a single particle failed in finding the geometrical place. The number of particles that will not converge is 4 for a swarm of 300 particles and 6 for 400 particles respectively.

6.3. Problem 3: Hyperbola

Geometrical place of the points M for which the difference (in absolute value) to two given points F_1 and F_2 is constant and equal to a given number k. The geometrical place is the hyperbola having the foci with points F_1 and F_2 respectively. Results obtained by INPSO, GPEA and hybrid INPSO-GPEA are presented in Figure 5. From Figure 5, it is evident that the GPEA converges faster than INPSO. It needs only 200 generations for GPEA to converge while for INPSO it required 300 generations. The hybrid INPSO–GPEA approach took 200 generations to converge. Number of solutions that did not converge after 100 generations by applying INPSO is equal to 2.

6.4. Problem 4: Parabola

Geometrical place of the points M those are equidistant from a point F (the focus) and a line (the directrix). Results obtained by INPSO, GPEA and hybrid INPSO – GPEA) are presented in Figure 6.

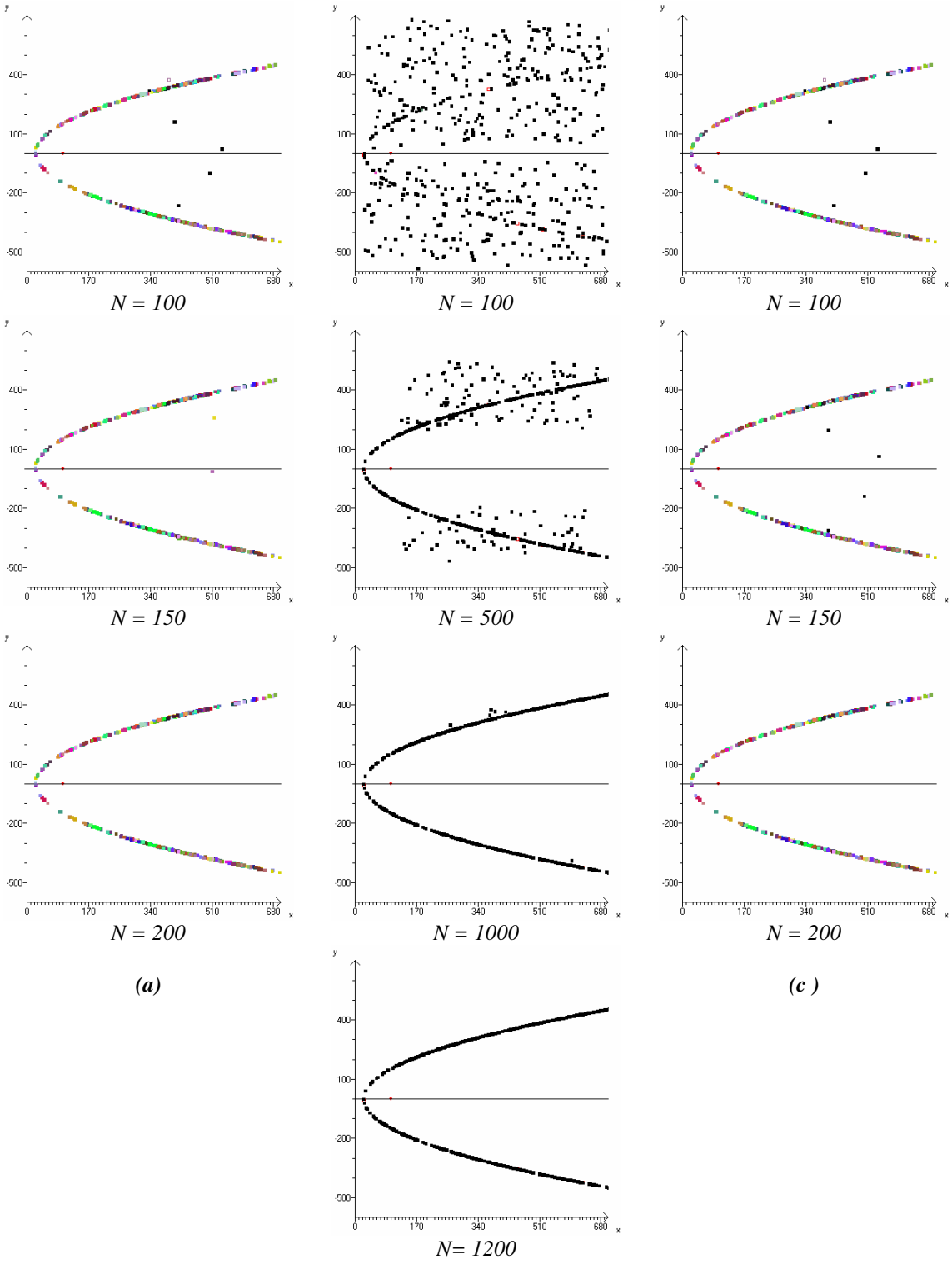

FIGURE 6. Parabola (a) INPSO, (b) GPEA, (c) Hybrid INPSO – GPEA

As illustrated in Figure 6, GPEA converges very slowly. It needs about 1200 generations until all solutions are converge to the geometrical place. INPSO and the hybrid INPSO – GPEA algorithms converged within 200 generations. Number of particles that failed to converge in 100 generations by applying INPSO is 28 (from a population of 500 individuals). For INPSO of size 100 particles, all the particles converged within 100 generations.

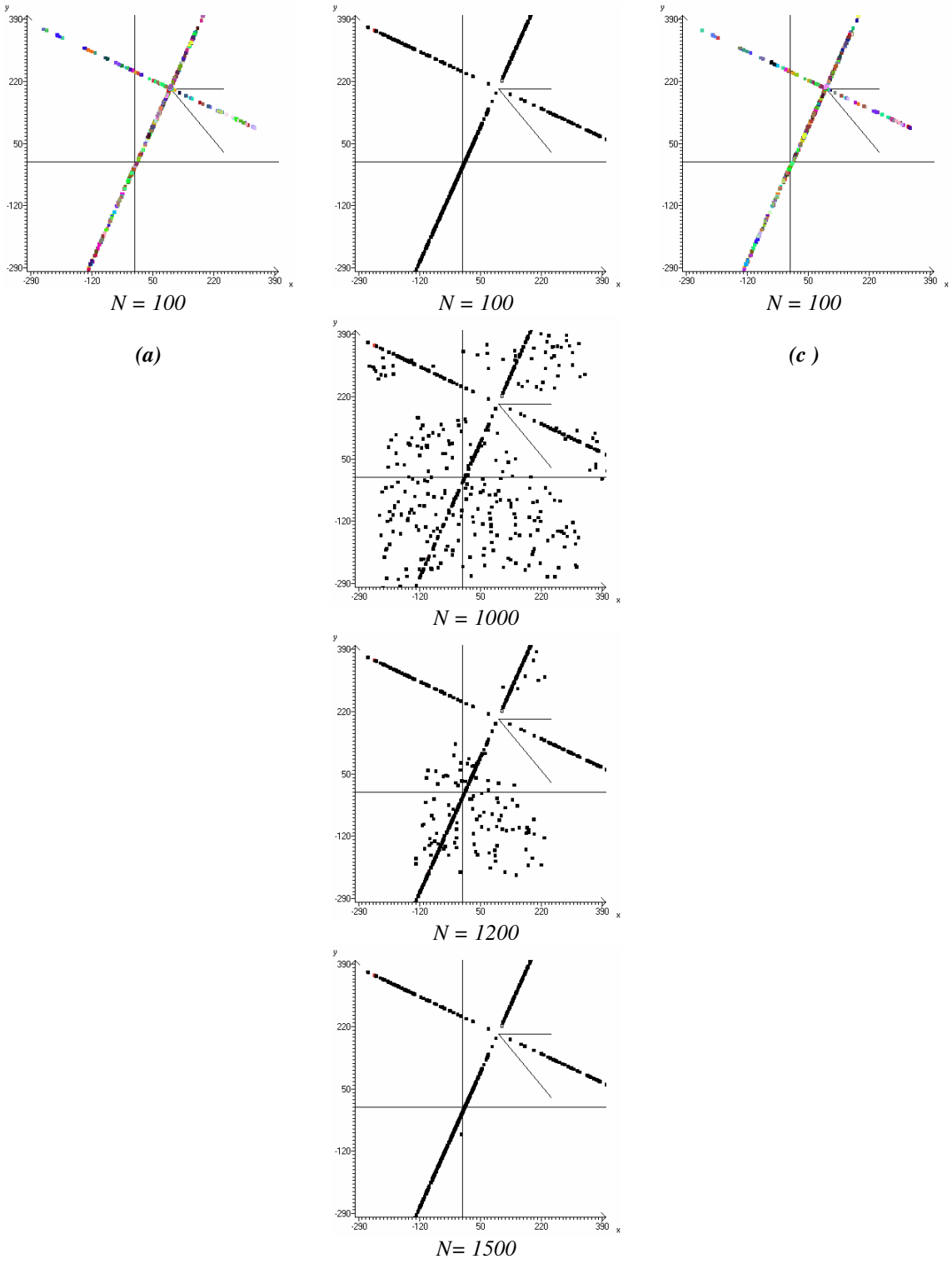

FIGURE 7. Lines (a) INPSO, (b) GPEA, (c) Hybrid INPSO – GPEA

For a swarm of 200 individuals, 8 particles failed to find the geometrical place. The number of particles that did not converge is 12 for a swarm of 300 particles and 24 for a swarm of 400 particles respectively.

6.5. Problem 5: Lines

Geometrical place of the points M which are equidistant from two given convergent straight lines.

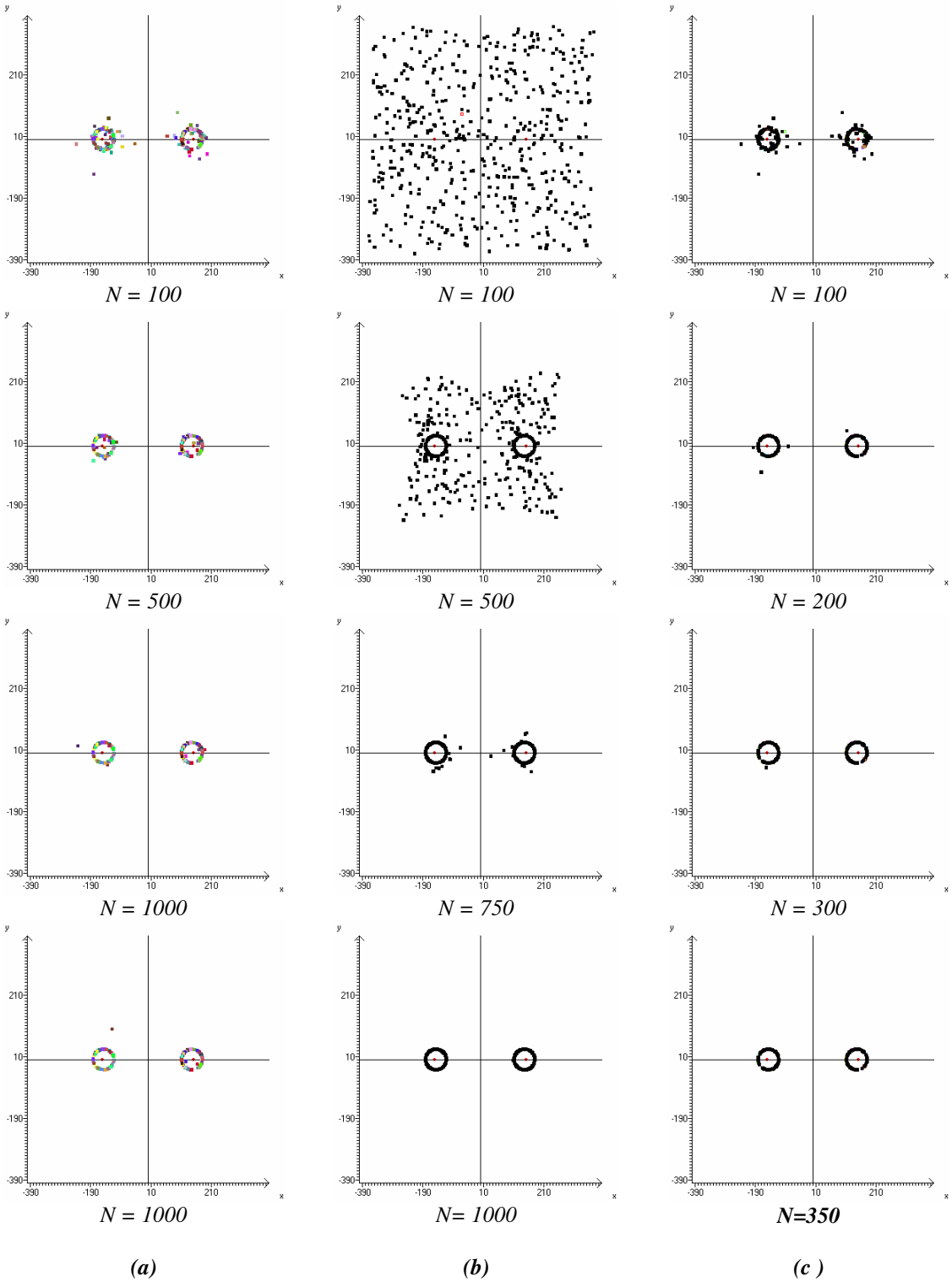

FIGURE 8. Oval of Cassiani (*a = 100, c* = 150) (a) INPSO, (b) GPEA, (c) Hybrid INPSO–GPEA

The geometrical place consists of two straight lines which are the bisecting lines for all angles which these two lines (and their extensions) can form. Results obtained by INPSO, GPEA and hybrid INPSO – GPEA are depicted in Figure 7. As evident from Figure 7, INPSO has converged very fast (in about 100 generations) while GPEA took nearly 1500 generations. In this case, there is no need to hybridize INPSO and GPEA.

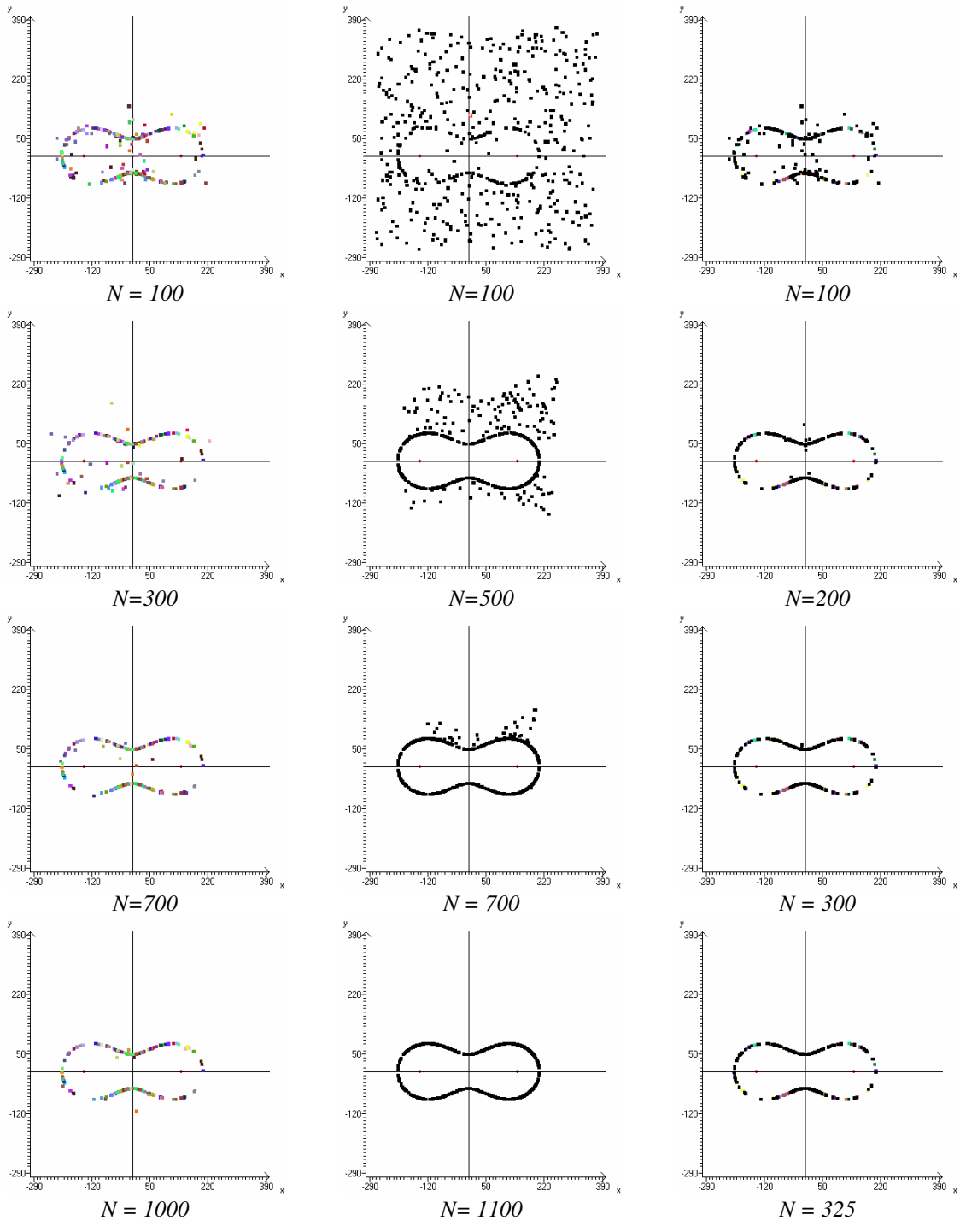

FIGURE 9. Oval of Cassiani (a = *150, c* = 142) (a) INPSO, (b) GPEA, (c) Hybrid INPSO–GPEA

6.6. Problem 6: The Oval of Cassiani

Geometrical place of the points M for which the product of distances to two given points F_1 (-c, 0) and F_2(c, 0) is constant and equal to a constant a^2. The geometrical place for this problem is called the *oval of Cassiani.* We can identify four different situations for different values of a and c. These four situations are:

1) $a < c$;
2) $c < a < c\sqrt{2}$;
3) $a > c\sqrt{2}$;
4) $a = c$.

Each of these situations is analyzed in the following sub-sections.

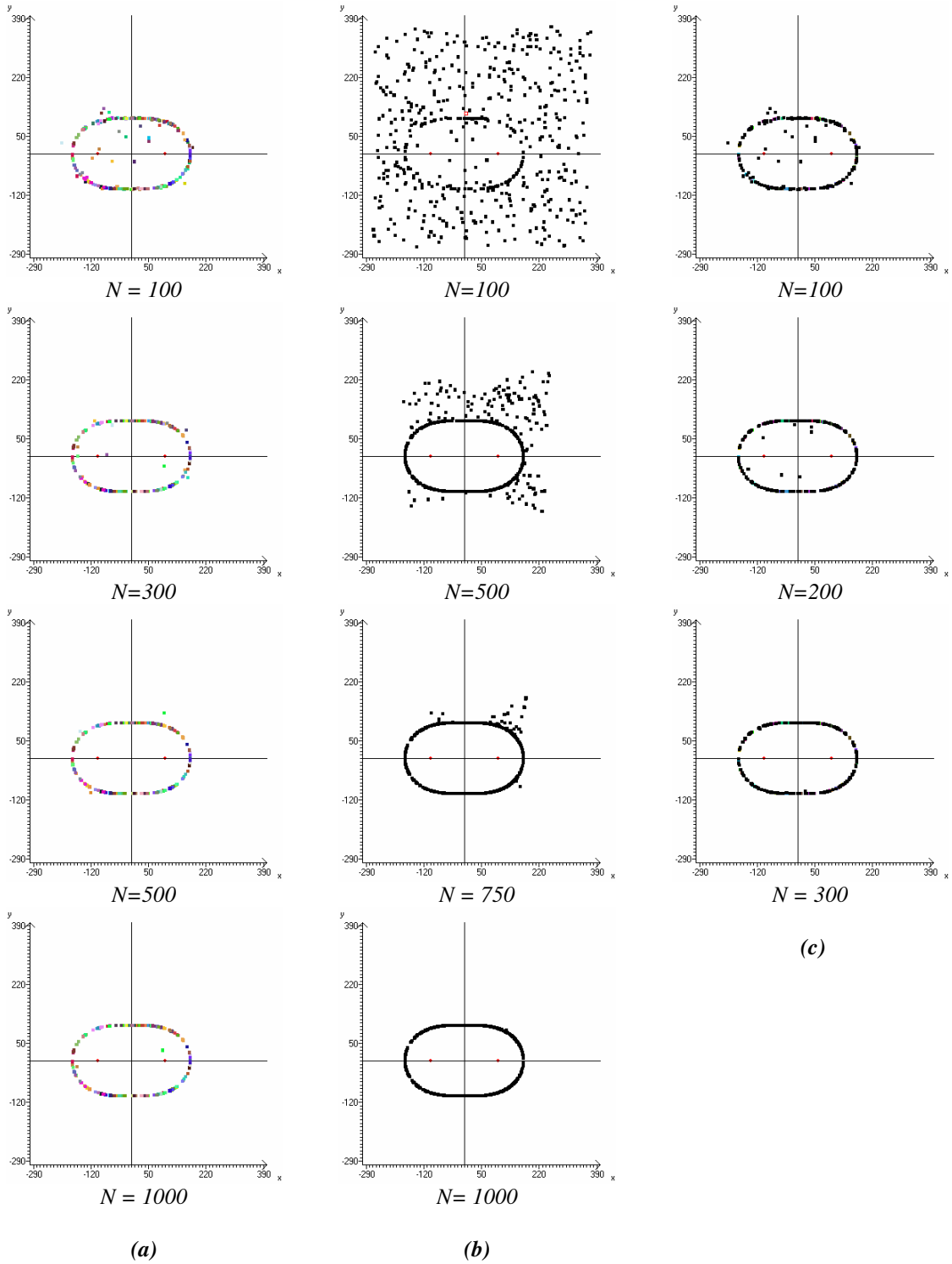

FIGURE 10. Oval of Cassiani (a = *143, c* = 100) (a) INPSO (b) GPEA (c) Hybrid INPSO–GPEA

5.6.1 Case a < c

We consider the following values: a = 100, c = 150. Results obtained by INPSO, GPEA and hybrid INPSO – GPEA are depicted in Figure 8. Figure 8 illustrates the effectiveness of using the hybrid approach. Both INPSO and GPEA require more than 1000 generations to converge to the geometrical place. The hybrid approach will converge in 350 generations. Number of particles that failed to converge in 100 generations by applying INPSO is 190 (from a population of 500 individuals). For a swarm of size 100 particles, 40 particles did not converge in 100 generations. For a swarm of 200 individuals, 72 particles failed to find the geometrical place. The number of particles that did not converge is 121 for a swarm of 300 particles and 161 for a swarm of 400 particles respectively.

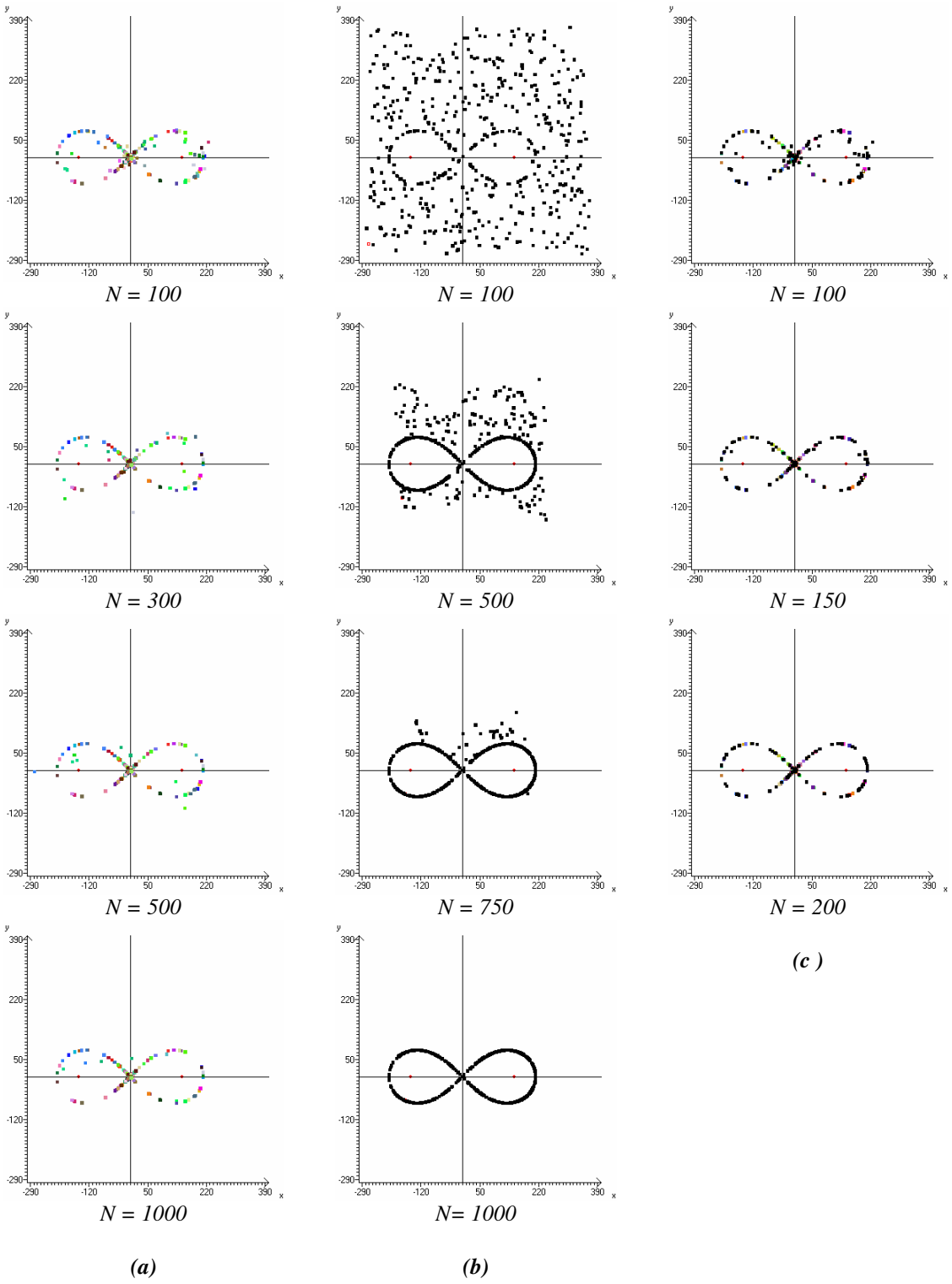

FIGURE 11. Oval of Cassiani ($a = c = 150$) (a) INPSO, (b) GPEA (c) Hybrid INPSO – GPEA

5.6.2 Case $c < a < c\sqrt{2}$

We considered the values $a = 150$, $c = 142$. Results obtained by INPSO, GPEA and hybrid INPSO – GPEA are depicted in Figure 9. As evident from Figure 9, the hybrid INPSO – GPEA approach obtained the best results compared to INPSO (which converged in 1000 generations) and GPEA (which required 1100 generations to converge).

Number of particles that failed to converge in 100 generations by applying INPSO is 157 (out of a population of 500 individuals). For a swarm of size 100 particles, 25 particles did not converge in 100 generations. For a swarm of 200 individuals, 90 particles failed to find the geometrical place. The number of particles that will not converge is 125 for

For this case, the hybrid approach converged much faster (only 300 generations) compared to INPSO and GPEA (both converged in about 1000 generations). Number of particles that failed to converge in 100 generations by applying INPSO is 191 (out of a population of 500 individuals). For a swarm of size 100 particles, 32 did not converge in 100 generations. From a swarm of 200 individuals, 70 particles failed to find the geometrical place. The number of particles that will not converge is 125 for a swarm of 300 particles and 166 for a swarm of 400 particles respectively.

5.6.4 Case a = c

We considered the value of 150 for both *a* and *c* in this situation. The results obtained by INPSO, GPEA and hybrid INPSO – GPEA are depicted in Figure 11. Compared to INPSO and GPEA, the hybrid approach is much faster. As evident from Figure 11, INPSO and GPEA required 1000 generations to converge. The hybrid approach converged in about 200 generations. Number of particles that failed to converge in 100 generations by applying INPSO is 82 (out of a population of 500 individuals). For a swarm of size 100 particles, 18 particles did not converge in 100 generations. For a swarm of 200 individuals, 32 particles failed to find the geometrical place. The number of particles that did not converge is 46 for a swarm of 300 particles and 67 for a swarm of 400 particles respectively.

6. CONCLUSIONS

Geometrical place problems are a well known class of mathematical problems. Even if there are situations when these problems are trivial, there are also situations when mathematicians spend several hours for detecting a real geometrical place for a given problem. We approached these problems by using two well known intelligent techniques namely evolutionary algorithms (GPEA) and a variant of particle swarm optimization which uses sub-swarms (INPSO).

We performed several experiments and comparisons between these two techniques. As evident from comparisons results, evolutionary algorithms work very well for all situations. They are less susceptible to getting 'stuck' at local optima than particle swarm optimization techniques. But

a swarm of 300 particles and 114 for a swarm of 400 particles respectively.

5.6.3 Case Case $a > c\sqrt{2}$

We used the values 143 for *a* and 100 for *c*. Results obtained by all three techniques are presented in Figure 10.
they tend to be computationally expensive and needs longer time to find all the solutions. For most of the considered problems (except Oval of Cassiani), the proposed INPSO technique could obtain better results in a faster, cheaper way when compared with other methods. For the Oval of Cassiani problem, INPSO failed to converge as expected.

Further, we exploited the advantages of these two methods: INPSO is very fast but sometimes there can be particles which will never converge; GPEA is computationally slow, but always could assure convergence. Consequently, we combined these two techniques: by using the fast convergence property of INPSO and the capabilities of GPEA to locate all the solutions. In the hybrid approach, INPSO is run for 100 iterations and then GPEA is applied to the population obtained by INPSO. The proposed hybrid approach seems to work very well for all the problems considered especially for the Oval of Cassiani problem where the GPEA and INPSO approaches required more than 65% iterations when compared to the hybrid approach (INPSO-GPEA).

REFERENCES

Angeline, P. 1998, “Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Difference”, “In Proceedings of the 7th Annual Conference on Evolutionary Programming”, San Diego,USA.

Baker, J. E. 1985, “Adaptive selection methods for genetic algorithms”. In *Proceedings of First International Conference on Genetic Algorithms*, 101-111.

Box, G. E. P. 1993,“Evolutionary operators: a method for increasing industrial productivity”. *Journal Royal Statistical Society*, 6: 81-101.

Brits, R., Engelbrecht, A. P., van den Bergh, F. 2002, “A Niching Particle Swarm Optimizer”. In *Proceedings of the Conference on Simulated Evolution And Learning*, Singapore.

Clerc, M. 1999, “The swarm and the queen: towards a deterministic and adaptive particle swarm optimization”. In *Proceedings of the IEEE*

Congress on Evolutionary Computation (CEC), pp. 1951-1957.

Davis, L. 1991, "Handbook of Genetic Algorithms". Van Nostrand Reinhold, New York.

Eberhart, R. C. and Kennedy, J. 1995, "A new optimizer using particle swarm theory". In *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, Nagoya, Japan. pp. 39-43.

Eberhart, R. C., Simpson, P. K., and Dobbins, R. W. 1996, "Computational Intelligence PC Tools". Boston, MA: Academic Press Professional.

Eberhart, R. C. and Shi, Y. 2001, "Particle swarm optimization: developments, applications and resources". In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, Seoul, Korea.

Eshelman, L. J., Caruna, R.A., Schaffer, J. D. 1989, "Biases in the crossover landscape". In *Proceeding of the Third International Conference on Genetic Algorithms*, J. Schaffer (ed.), Morgan Kaufmann Publisher, Los Altos, CA, 10-19.

Fogel, L. J. 1994, "Evolutionary Programming in Perspective: the Top-down View". In *Computational Intelligence: Imitating Life*, J.M. Zurada, R. J. Marks II, and C. J. Robinson, Eds., IEEE Press, Piscataway, NJ.

Goldberg, D. E. 1989, "Genetic Algorithms in Search, Optimization, and Machine Learning". Reading MA: Addison-Welsey.

Grosan C., 2004, "Solving geometrical place problems by using Evolutionary Algorithms". In *Proceedings of World Computer Congress*, M. Kaaniche (Ed.), Toulouse, France, pp. 365-375.

Grosan, C, Abraham A., and Nicoara, M. 2005, "Geometrical Place Problems Using Particle Swarm and Evolutionary Algorithms", In *Proceedings of the 7th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05)*, Timisoara, Romania, IEEE CS Press.

Grosan, C, Abraham A., Han, S.Y, and Gelbukh, A. 2005, "Hybrid Particle Swarm - Evolutionary Algorithm for Search and Optimization", *In Proceedings of the 4th Mexican International Conference on Artificial Intelligence, Mexico, Lecture Notes in Computer Science,* Springer Verlag, Germany.

Holland, J. 1975, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

Hu, X., Shi Y., and Eberhart, R.C. 2004, "Recent Advences in Particle Swarm", In *Proceedings of Congress on evolutionary Computation (CEC)*, Portland, Oregon, pp. 90-97.

Kennedy, J. and Eberhart, R. C. 1995, "Particle Swarm Optimization". In *Proceedings of IEEE International Conference on Neural Networks*, Perth, Australia, IEEE Service Center, Piscataway, NJ, Vol.IV, pp.1942-1948.

Kennedy, J. 1997, "The Particle Swarm: Social Adaptation of Knowledge", In *Proceedings of IEEE International Conference on Evolutionary Computation*, Indianapolis, Indiana, IEEE Service Center, Piscataway, NJ, 303-308.

Kennedy, J. 1997, "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.

Kennedy, J. 1998, "The Behavior of Particles", In *Proceedings of 7th Annual Conference on Evolutionary Programming*, San Diego, USA.

Kennedy, J. 1998, "Thinking is social: Experiments with the adaptive culture model". Journal of Conflict Resolution, 42,56-76.

Koza, J. R. 1992, Genetic Programming: "On the Programming of Computers by Means of Natural Selection", MIT Press, Cambridge, MA.

Peer, E.S., van den Bergh, F., Engelbrecht, A. P. 2003, "Using Neighborhoods with Guaranteed Convergence PSO". In *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 235-242, USA.

Pomeroy, P., 2003, "An Introduction to Particle Swarm Optimization", http://www.adaptiveview.com/articles/ipsop1.html.

Rechenberg, I. 1994, "Evolution Strategy", In *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J. Marks II, and C. Robinson, Eds., IEEE Press, Piscataway, NJ.

Schaffer, J. D., Morishima, A. 1987, "An adaptive crossover distribution mechanism for genetic algorithms". In *Proceeding of the Second International Conference on Genetic Algorithms*, J. J. Grefenstette (Ed.),Lawrance Erlbaum Associates, Hillsdale, NY, 36-40.

Shi, Y., and Eberhart, R. C. 1999, "Empirical study of particle swarm optimization". In *Proceedings of Congress on Evolutionary Computation*, 1945-1950. Piscataway, NJ: IEEE Service Center.

Shi, Y., and Eberhart, R. C. 1998, "Parameter selection in particle swarm optimization", In *Proceedings of the Annual Conference on Evolutionary Computation*.

Shi, Y. and Eberhart, R. C. 1998, "A modified particle swarm optimizer". In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, Piscataway, NJ. pp. 69-73

Spears, W. M., De Jong, K. A. 1991, "On the virtues of uniform crossover". In *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publisher, 230-236.

Syswerda, G. 1989, "Uniform crossover in genetic algorithms". In *Proceedings of the third Conference in Genetic Algorithms*, J. Schaffer (ed.), Morgan Kaufmann Publisher, Los Altos, CA, 2-9.

Author Biographies

Crina Grosan currently works as an Assistant Professor in the Computer Science Department of Babes-Bolyai University, Cluj-Napoca, Romania. Her main research area is in Evolutionary Computation, with a focus on Evolutionary Multiobjective Optimization and applications and Genetic Programming. Crina Grosan authored/co-authored over 50 papers in peer reviewed international journals, proceedings of the international conferences and book chapters. She is co-author of two books in the field of computer science. She proposed few Evolutionary techniques for single and multiobjective optimization, a genetic programming technique for solving symbolic regression problems and so on. Dr. Grosan is the co-editor for a book on Swarm Intelligence for Data Mining, which will be published by Springer Verlag, Germany. She is member of the IEEE (CS), IEEE (NN) and ISGEG. She received her PhD degree from Babes-Bolyai University, Romania.

Ajith Abraham currently works as a Distinguished Visiting Professor under the South Korean Government's Institute of Information Technology Assessment (IITA) Professorship programme at Chung-Ang University, Korea. He is also a visiting researcher of Rovira i Virgili University, Spain and an Adjunct Professor of Jinan University, China and Dalian Maritime University, China. His primary research interests are in computational intelligence with a focus on using evolutionary computation techniques for designing intelligent paradigms. Application areas include several real world knowledge-mining applications like information security, bioinformatics, Web intelligence, energy management, financial modelling, weather analysis, fault monitoring, multi criteria decision-making etc. He has authored/co-authored over 200 research publications in peer reviewed reputed journals, book chapters and conference proceedings of which three have won 'best paper' awards.

He is the Editor of The International Journal of Hybrid Intelligent Systems (IJHIS), IOS Press, Netherlands; Journal of Information Assurance and Security (JIAS), USA; International Journal of Computational Intelligence Research (IJCIR), Neurocomputing Journal, Elsevier Science, The Netherlands; International Journal of Systems Science (IJSS), Taylor & Francis, UK; Journal of Universal Computer Science (J.UCS), Austria; Journal of Information and Knowledge Management, World Scientific, Singapore; Journal of Digital and Information Management (JDIM), Digital Information Research Foundation, India and International Journal of Neural Parallel and Scientific Computations (NPSC), USA. Since 2001, he is actively involved in the Hybrid Intelligent Systems (HIS) and the Intelligent Systems Design and Applications (ISDA) series of annual International conferences. He was also the General Co-Chair of The Fourth IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST05), Japan and the Program Co-Chair of the Inaugural IEEE Conference on Next Generation Web Services Practices, Seoul, Korea. He received PhD degree from Monash University, Australia. More information at: http://ajith.softcomputing.net

Monica Nicoara is student of Faculty of Mathematics and Computer Sciences, Babes-Bolyai University, Cluj-Napoca, Romania. Her main research areas are Particle Swarm Optimization, Evolutionary Computation and Combinatorial Optimization.