A Performance Comparison of PSO and GA Applied to TSP

Abdelhakim Gharib
Laboratoire LISER, ENSEM
UH2C
KM7, BP 8118 Route El Jadida
Casablanca, Morocco

Jamal Benhra
Laboratoire LISER, ENSEM
UH2C
KM7, BP 8118 Route El Jadida
Casablanca, Morocco

Mohsine Chaouqi
Laboratoire LISER, ENSEM
UH2C
KM7, BP 8118 Route El Jadida
Casablanca, Morocco

ABSTRACT
The aim of this article is to present a collective intelligence approach to help solving optimization problems and apply it in particular to the Travelling Salesman Problem. The approach used is the particle swarm optimization (PSO) whose main idea is to simulate the collective behavior of a cloud. This article also compares the results obtained using PSO algorithm with those obtained by using another famous metaheuristic wich is the Genetic Algorithm.

Keywords
Traveling salesman problem; Particle Swarm Optimization; Genetic algorithm

1. INTRODUCTION
In many applications in real world, there's always a need to find optimal configurations from a discrete set of objects. This is known as a combinatorial optimization problem. While many of these combinatorial optimization problems can be solved in polynomial time, a majority belongs to the class of NP-hard problems [1]. To face these hard combinatorial optimization problems, approximations and heuristics algorithms were used as a compromise between the quality of the solution and the computation time [2]. A class of heuristic algorithms: meta-heuristic algorithms, was developed showing, at the same time, promising results in the field of combinatorial optimization. This class includes: Particle Swarm Optimization (PSO), Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithms (GA) [3], Ant Colony Optimization (ACO) [4]…etc

One of the benchmark problems used to test these heuristics algorithms is the traveling salesman problem (TSP). In this problem, the traveling salesman must visit n cities and get to each city only once. The goal is to minimize the total distance traveled. One of the first researchers who dealt with the TSP and study it in detail was K. Menger [5]. There are two types of the TSP problem, the asymmetric case and the symmetric case. The symmetric TSP is a special case of the problem in which locations have coordinates in a Euclidian plane [6] and it was proven that this problem is NP-hard [7]. This means that the time to find an optimal solution increases exponentially depending on the size of the problem. Solving this problem would require the use of very efficient algorithms, therefore it was decided to opt for using two famous algorithmic approaches: Particle Swarm Optimization and Genetic Algorithms.

The goal of this work is to compare the PSO algorithm, in its standard case, with another heuristic algorithm wich is the GA at solving instances of the TSP [8]. For this reason, a simple instance that we created, and three benchmark Euclidian symmetric TSPs are used with increasing complexity.

The rest of the paper is organized as follows: the second chapter dresses a state of art of the methods implemented (PSO & GA) and the problem studied (TSP), the third and the fourth chapter present the results of the different simulations done and a comparison between the two heuristics, and the final chapter concludes this work.

2. STATE OF THE ART
2.1 Particle Swarm Optimization
The particle swarm optimization (PSO) is a stochastic optimization method developed by EBERHART and KENNEDY in 1995 [9]. It draws the origin of the ecosystem, specifically the social behavior of animals living in swarms, such as schools of fish and grouped flights of birds.
In its application to optimization problems, this method relies on a set of individuals, originally arranged randomly, called particles. This particles move in the search space. Each one is considered as a solution of the problem, since they have a position X_{id} and a speed V_{id}. In addition, each particle has a memory about his best position visited P_{id} and the neighborhood's one Pgd.

The evolution of the algorithm equations is given as following:

$$\begin{cases} V_{id}^{t+1} = \omega\, V_{id}^{t} + C_1R_1\,(P_{id} - X_{id}) + C_2R_2\,(P_{gd} - X_{id}) \\ X_{id}^{t+1} = X_{id}^{t} + V_{id}^{t+1} \end{cases}$$

ω denotes the coefficient of inertia , the coefficients C_1 and C_2 are constants determined empirically according to the relationship $C_1 + C_2 \leq 4$, and finally, R_1 and R_2 are random positive numbers following a uniform distribution on [0,1]

[9].

The displacement strategy of a particle, as shown in Fig1, is influenced by the following three components:

1. A component of inertia ($\omega\, V_{id}^{t}$): the particle tends to follow its current direction of travel;
2. A cognitive component (C1R1 (Pid - Xid)): the particle tends to move towards the best site for which it has already crossed over;
3. A social component (C2R2 (Pgd - Xid)): the particle tends to rely on the experience of its congeners and thus to head for the best site already achieved by its neighbors.

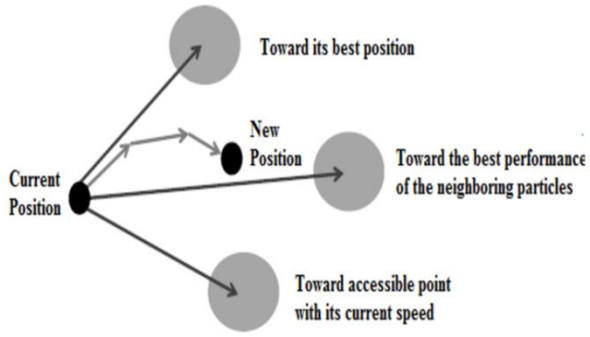

Fig 1: Particles movement

The operations made in the PSO algorithm are explained in Fig 2 as follows:

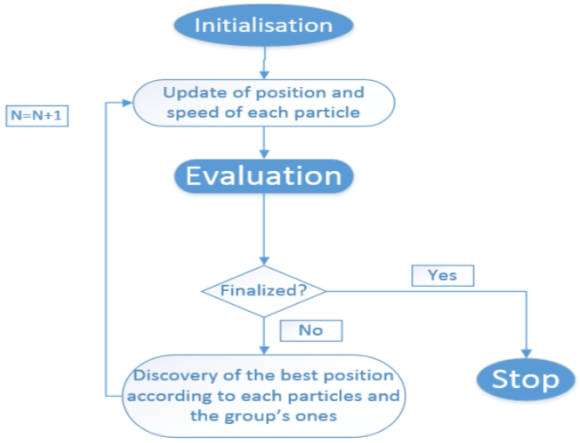

Fig 2: Launch chart of the PSO algorithm

2.2 The Genetic Algorithms

Genetic Algorithm was introduced by Holland et al. [10]. It is inspired by Darwin's theory about evolution and based on mimicking the survival of the fittest among the species generated by random changes in the gene-structure of the chromosomes in the evolutionary biology [11].

In the Genetic Algorithm logic, a solution vector is called individual or chromosome. Each chromosome is made of discrete units called genes and each gene controls one or more elements of the chromosome. Normally, a chromosome is a unique solution in the solution space. GA operates with a set of chromosomes, called population. The population is normally initialized randomly [11].

The basic process for a genetic algorithm is:

1. Initialization: Create an initial population. This population is usually randomly generated and can be any desired size, from only a few individuals to thousands.
2. Evaluation: Each member of the population is then evaluated and a 'fitness' value for that individual is calculated.
3. Selection: Discard the bad designs and keep only the best individuals in the population.
4. Crossover: Create new individuals by combining aspects of the selected individuals. This is like mimicking reproduction in nature. The hope is that by combining certain traits from two or more individuals, an even 'fitter' offspring which will inherit the best traits from each of its parents, will be created.
5. Mutation: Maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next by making small changes at random to an individual's genome.
6. Repeat: After obtaining the next generation, start again from step two until reaching a termination condition.

2.3 Traveling Salesman Problem

The traveling salesman problem (TSP) is a well-known and important combinatorial optimization problem.
Generally, the process is as follows: start by choosing any city in a given list and end up by returning to the departure one. The traveled distance must be minimized knowing that the distances between cities are known. The notion of distance may be replaced by the time or the cost.

According to Kemighan [12], the definition of the traveling salesman problem is as following:

Given a graph G = (N, A), where N = $\{v_0, v_1 \ldots . v_n\}$ is the set of nodes (cities). And let A = $\{(v_i, v_j) / v_i, v_j \in N, i < j\}$ be the set of stops that connect the nodes if the distances are symmetrical and A = $\{(v_i, v_j) / v_i, v_j \in N\ i \neq j\}$ represents the arcs if the distances are asymmetric. In other words, the problem is symmetrical if the distance for the trip from town A to town B is the same as the reverse one.

Despite the simple problem statement, solving the TSP is difficult since it belongs to the class of NP-Hard problems [7]. In the symmetric case, according to this study, a calculation of the complexity of the problem shows that the number of feasible solutions is given by the formula:

(n-1)! /2; where n is the number of cities.

Considering the computation time for a trip as equal to 1 µs, the calculation time found is big enough as shown in Table 1.

Table 1. Calculation time and number of possibilities

Number of cities	Number of possibilities	Time calculation
4	3	3 µs
14	3113510400	52 min
20	60E+15	1928 year
52	7.8E+65	2.5E+43 millard years
280	4.66E+157	2.5E+43 milliard years

To test the algorithmic approaches, it was decided to run several experiments, varying the complexity of the TSP. The two algorithms are tested on three benchmark instances and one personal instance wich coordinates are:

X {30, 50, 43, 40, 40, 36, 21, 8, 16, 7, 3, 80, 55, 60, 45, 1, 30, 45, 25, 3};

Y {5, 9, 10, 30, 20, 43, 48, 40, 36, 21, 18, 16, 27, 17, 80, 1, 12, 21, 65, 4};

TSPLIB: http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html, provides most of the benchmark problems. It offers problem coordinates, best tour and the optimum solutions for the three test problems

The benchmark TSPs used are Berlin52, Eil101, and A280.

3. RESULTS AND DISCUSSIONS

All the simulations were completed on a Windows 64 bits personal computer with an i7-4710 processor clocked at 2.50GHz, and 8 GB of Ram. The Genetic algorithm was developed in MATLAB. The Particle Swarm optimization algorithm was written in JAVA.

3.1 Particle Swarm Optimization

3.1.1 Personal Instance

The first instance to test the PSO is the personal instance with 20 locations and coordinates as follows:
X {30, 50, 43, 40, 40, 36, 21, 8, 16, 7, 3, 80, 55, 60, 45, 1, 30, 45, 25, 3};
Y {5, 9, 10, 30, 20, 43, 48, 40, 36, 21, 18, 16, 27, 17, 80, 1, 12, 21, 65, 4};

The PSO finds the best tour in 3.987 seconds. Fig 3 presents the evolution of the objective function wich is the distance over time.

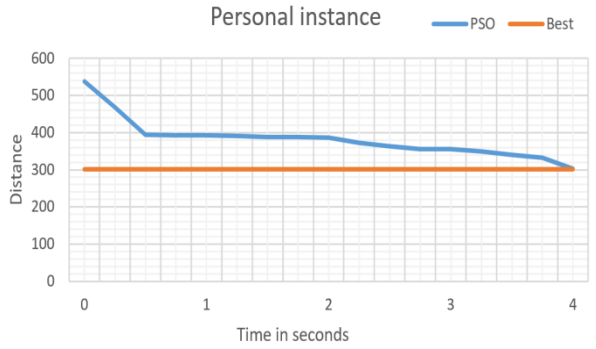

Fig 3: Evolution of PSO results over time in the personal instance

3.1.2 Berlin 52

The second instance is Berlin 52 with 52 locations in Berlin (Groetschel) [13]. The PSO gives average results but doesn't find the optimal solution. The convergence time is considerably short: 19.23 seconds. The PSO returns an optimal tour equals to 12568.

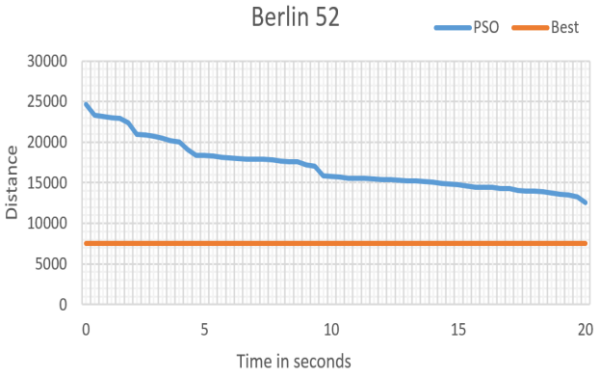

Fig 4: Evolution of PSO results over time in Berlin 52

3.1.3 Eil 101

The third instance is Eil101 with 101 cities. It is considered a large TSP instance [13]. The PSO gives average results but doesn't find the optimal solution. It returns an optimal tour equals to 1576 in 53.346 seconds.

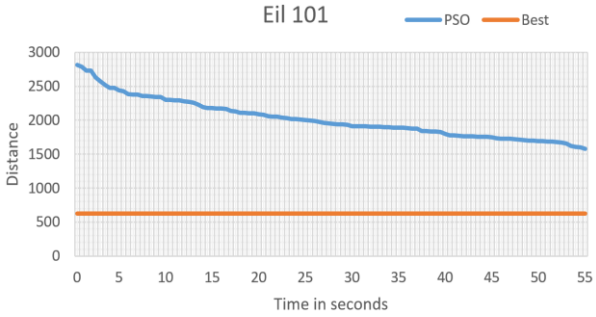

Fig 5: Evolution of PSO results over time in Eil 101

3.1.4 A 280

A280 is the largest instance in this work. It has 280 location and its optimal tour is 2579 [13]. It requires more computational time due to its complexity. Even if the PSO doesn't find the optimal tour, its convergence is considerably good. It starts with a value equals to 20702 and ends with a value equals to 7437 in 1563.709 seconds.

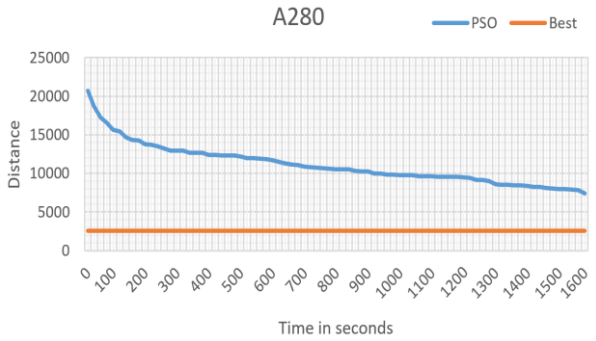

Fig 6: Evolution of PSO results over time in A280

3.2 Genetic algorithms

3.2.1 Personal instance

The GA converges quickly and finds the optimal tour in 1.371 seconds.

Fig 7 illustrates the performance of the GA in providing the best tour

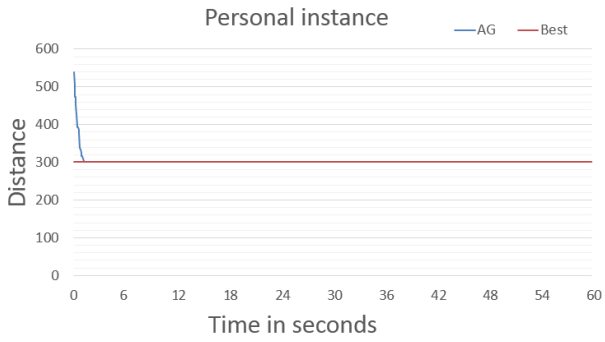

Fig 7: Evolution of GA results over time in the personal instance

3.2.2 Berlin 52

The GA converges quickly in this instance too but doesn't find the optimal tour wich is 7542 [13]. It finds a tour of length 7935 in 119.394. The evolution can be described as a set of two phases: The first one, where the algorithm converges rapidly and provides a tour with a distance equals

to 8988 in less than 10 seconds. The second phase where the improvement goes slowly.

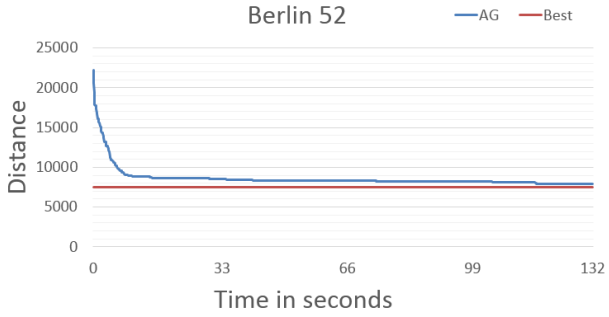

Fig 8: Evolution of GA results over time in Berlin 52

3.2.3 Eil 101

Eil101 optimal tour is 629. The GA maintains the same two phases' behavior; it converges to a tour of 808 in 46 seconds and then its evolution goes slowly. Then it returns an overall best tour of 728 after 227.611 seconds.

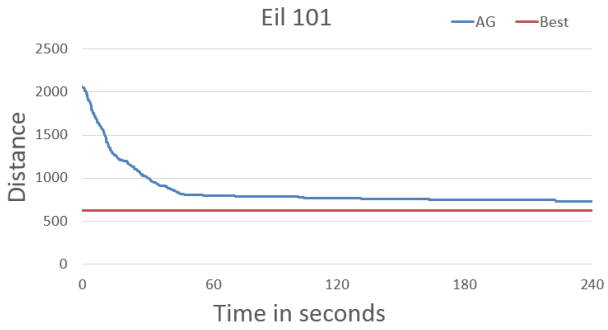

Fig 9: Evolution of GA results over time in Eil 101

Table 2. Comparison between PSO and GA

Max iterations = 1000	PSO			GA		
	Starting value	Optimal found	Time	Starting value	Optimal found	Time
Personal instance	538.1	302.7	3.978	539.2	301.1	1.371
Berlin 52	24659.1	12568.8	19.23	22167.3	7935.8	119.394
Eil 101	1873.5	1576.2	53.346	2064.4	728.6	227.611
A 280	20702.3	7437	1563.709	31000	2818.6	402.101

3.2.4 A 280

This instance is large and the evolution of the GA changed a little but it treats this case with a good improvement factor as seen in Fig 10 and returns a best tour of 2818 in 402.101 seconds.

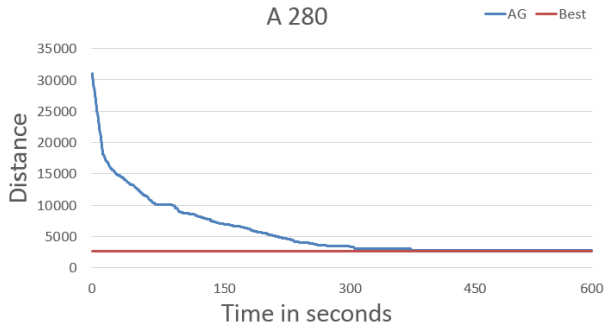

Fig 10: Evolution of GA results over time in A 280

4. COMPARISON OF THE RESULTS

Qualitative and quantitative comparisons were made to compare these two algorithmic approaches. Both algorithms give acceptable results in solving this NP-hard optimization problem. They are both efficient in attacking all instances of the TSP even if the search space is large and complex. Table 2 shows the computational results, and Fig 11 compares the best tours obtained with the two heuristics.

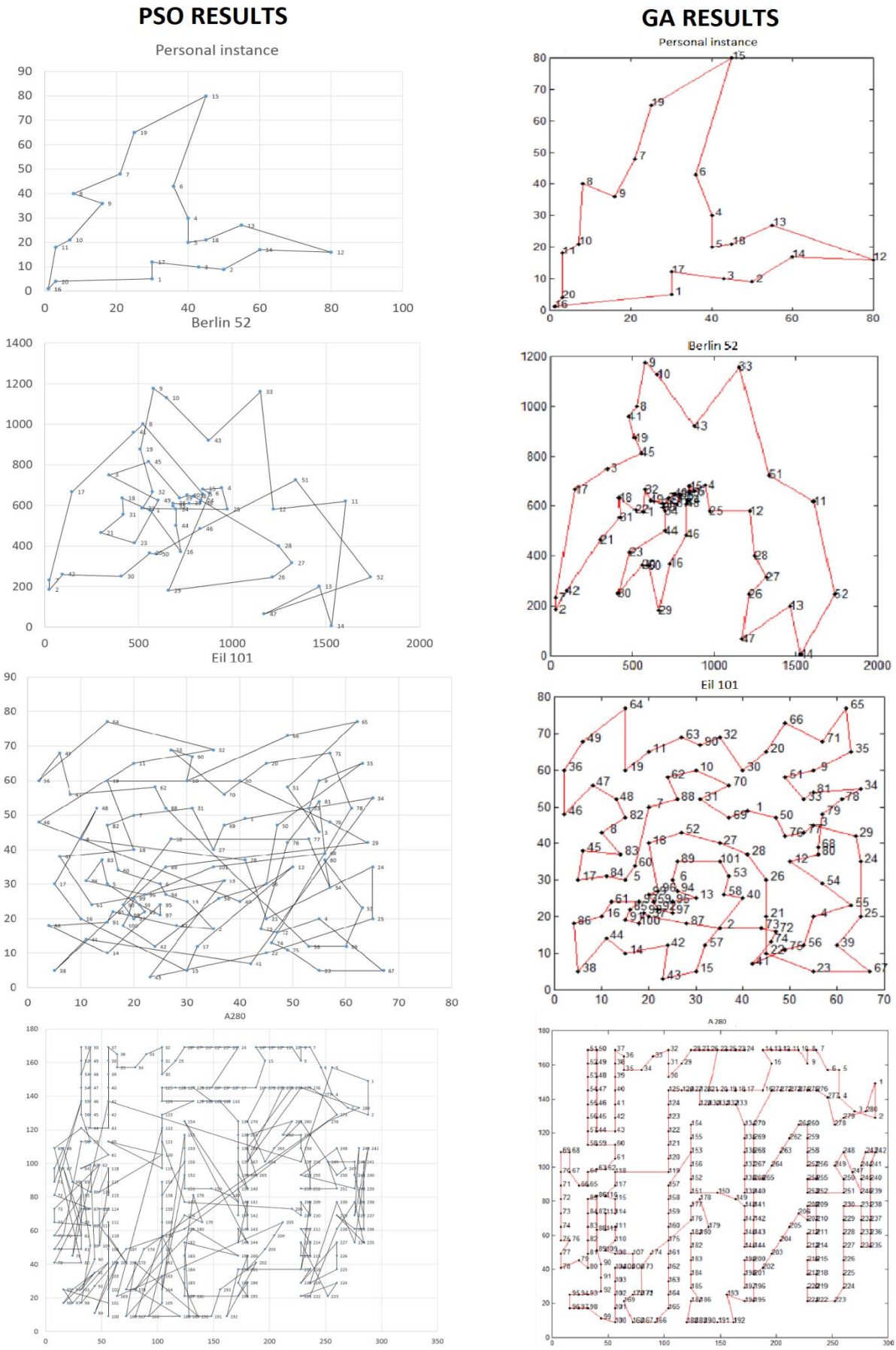

Fig 11: Best Tours obtained with PSO and GA

Even if the starting values of the PSO is smaller than the GA's one, except for Berlin 52, the GA provides more satisfying results than the PSO over the four instances and its convergence time is less than the PSO's one.

This is due to the fact that PSO in its standard configuration falls into local optima. This is because the major problem when it comes to using the PSO is in the limits of the search area. If the particles are functioning normally, they borrow many solutions that contain a lot of visits per node. It is also possible that the next node in the tour will be the current node. Both of these problems lead to a large number of possible solutions that are illegitimate. Indeed, the search space changes in the same manner as (n^n), while the number of possible solutions grows proportionally to (n!), where "n" is the number of nodes. To limit the search space methods have been developed to contain and move the particles around. [14]

5. CONCLUSION

This paper presented a comparison of two famous metaheuristics: PSO and GA. It was found that the GA provides better solutions than the standard PSO.
Although the PSO algorithm to its initial state, without modification or hybridization, showed a success on a variety of continuous functions, limited success has been shown to fit the PSO algorithm to more complex and richer areas such as combinatorial optimization.

This was the case in this research because the algorithm using simple PSO can not find better solutions than the GA for the four benchmark instances. It falls into local optimum.

For this reason, thinking about making an automatic tuning for the different parameters of the PSO (ω, C1, C2 and the number of particles) in order to find the best combination between the four parameters that may give better results, seems to be an effective solution. This would be done by hybridizing the PSO algorithm with other heuristic optimization algorithms such as neural network and compare the results that will be obtained with the actual results.

6. REFERENCES

[1] Aardal, K., Hoesel, S. v., Lenstra, J. K. and Stougie, L. (1997). A Decade of Combinatorial Optimization. Department of Information and Computing Sciences, Utrecht University, UU-CS-1997-12.

[2] Festa, P. and Resende, M. G. C. (2008). Hybrid Grasp Heuristics. AT&T Labs Research, Florham Park, July.

[3] El Hassani Hicham, Said Benkachcha, Jamal Benhra, "New genetic operator (jump crossover) for the traveling salesman problem". International Journal of Applied Metaheuristic Computing, 6(2), 33-44, April-June 2015 33.

[4] A. H. Sabry, A. Bacha, and J. Benhra, "A contribution to solving the traveling salesman problem using ant colony optimization and web mapping platforms Application to logistics in a urban context," in Codit'14, Metz, France, 2014.

[5] K. Menger, Ergebnisse eines mathematischen Kolloquiums: Deuticke, 1932Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[6] A. Punnen, "The Traveling Salesman Problem: Applications, Formulations and Variations," in The Traveling Salesman Problem and Its Variations. vol. 12, G. Gutin and A. Punnen, Eds., ed: Springer US, 2007, pp. 1-28.

[7] C. H. Papadimitriou, "The Euclidean travelling salesman problem is NP-complete," Theoretical Computer Science, vol. 4, pp. 237-244, 1977.

[8] Goldbarg, E.F.G; Souza, G.R. & Goldbarg, M.C. (2006a). Particle swarm for the traveling salesman problem, Proceedings of the EvoCOP 2006, Gottlieb, J. & Raidl, G.R. (Ed.), Lecture Notes in Computer Science, Vol. 3906, pp. 99-110, Budapest, Hungary, April 2006, Springer, Berlin.

[9] Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks, pp. 19421948, 27th November – 1 December, 1995.

[10] J. H. Holland, "Adaption in Natural and Artificial Systems," The University of Michigan Press, 1975.

[11] D.E. Goldberg. "Genetic Algorithms in Search, Optimization, and Machine Learning". AddisonWesley, New York, 1989.

[12] Kemighan, L. S. (1973). "An Effective Heuristic Algorithm for the Traveling Salesman Problem." Opérations Research 21: 2245-2269.

[13] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library," ORSA Journal of Computing, vol. 3, pp. 376-384, 1991.

[14] Shi XH, Liang YC, Lee HP, Lu C, Wang QX, "Particle swarm optimization-based algorithms for TSP and generalized TSP", Inf Process Lett 103(5), 2007, pp.169–176.