Article: A Performance Comparison of GA and ACO Applied to TSP

Article *in* International Journal of Computer Applications · May 2015

DOI: 10.5120/20674-3466

3 authors:

Ahmed Haroun Sabry
Ecole Nationale Supérieure d'Electricité et de…
12 PUBLICATIONS **11** CITATIONS
SEE PROFILE

Jamal Benhra
Ecole Nationale Supérieure d'Electricité et d…
55 PUBLICATIONS **47** CITATIONS
SEE PROFILE

Hicham El Hassani
Hassania School of Public Works
14 PUBLICATIONS **18** CITATIONS
SEE PROFILE

A Performance Comparison of GA and ACO Applied to TSP

Sabry Ahmed Haroun
Laboratoire LISER, ENSEM, UH2C
Casablanca, Morocco.

Benhra Jamal
Laboratoire LISER, ENSEM, UH2C
Casablanca, Morocco.

El Hassani Hicham
Laboratoire LISER, ENSEM, UH2C
Casablanca, Morocco.

ABSTRACT
This work presents a contribution to comparing two nature inspired metaheuristics for solving the TSP. We run ACO and GA on three benchmark instances with varying size and complexity, in addition to one real world application in the field of urban transportation and logistics. A first chapter presents algorithmic approaches. Results and discussion chapter outlines the computational behavior of the algorithms throughout the problem sets. The conclusion closes the discussion with recommendations and future scopes.

Keywords
Traveling salesman problem; Genetic algorithm; Ant colony optimization;

1. INTRODUCTION
A huge majority of engineering problems can be expressed in a general form of an optimization problem, wherein an objective function or cost function is defined, that is to be minimized with respect to all the involved constraints.

For example the widely studied traveling salesman problem (TSP), where the objective is to minimize the total travelled distance on a single closed tour visiting each city once. K. Menger [1] was one of the first researchers to address the TSP and study it in detail. The Euclidian TSP is a special case of the problem in which cities have coordinates in a Euclidian plane [2].

It was proven that this problem is NP-hard [3], solving it would require the use of very efficient algorithms, therefore we opt for the use of two famous algorithmic approaches: Ant colony optimization and Genetic Algorithms. A real world TSP is a more general case in which cities are represented in a geographic coordinates system.

These two approaches are called metaheuristics. Generally, metaheuristics are widely applied to all aspects of combinatorial optimization. We find in this family other approaches like Tabu search, simulated annealing [4], particle swarm optimization, and Evolutionary programming. They all aim to provide sufficiently good solutions to challenging problems [5] [6].

This work aims to compare ACO and GA at solving different instances of the famous TSP [7] [8], for this we are using one real world asymmetric TSP from the complex urban environment of the city of Casablanca and three benchmark Euclidian symmetric TSPs with increasing complexity. This work is structured as follow: the second chapter dresses a state of art of the methods implemented, the third chapter presents the results of the different simulations, and the final chapter concludes this work.

2. STATE OF THE ART
2.1 The Genetic Algorithm
The concept of GA was first introduced by Holland and his colleagues in the late 1960s [8] The GA are inspired from evolutionary theory.

In nature, unfit and weak species compared to their environment are facing extinction through natural selection. The strongest species have a greater opportunity to pass on their genes to future generations through reproduction; this process is time taking and gradual. In the long term, the species carrying the right combination of genes become more dominant [9].

Sometimes, during this slow process of evolution, random changes can occur to the genes. These unintended variations offer supplementary benefits to the natural selection process through diversification. In this never-ending challenge for survival, new species evolve from old ones, unsuccessful changes and combinations are automatically eliminated by natural selection [10] [11].

In the terminology of GA, a solution vector is called individual or chromosome. These chromosomes are made of discrete units called genes. Each gene controls one or more elements of the chromosome. The original implementation of GA's by Holland had binary digits as genes [8]. Latest implementations brought a greater variety of gene representations and encodings.

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. As the algorithm runs, it progressively finds individuals of higher fitness; each candidate has a set of properties (genotype) which can be mutated resulting in a higher chance of finding better solutions [12]. The algorithm terminates when a criterion was met (maximum number of iterations or a satisfactory fitness level reached). Hereby, the pseudo code of the GA:

Start Genetic Algorithm

Initialize population;

Evaluate population;

While : not termination do

Create new solutions:

Apply **Crossover** operator;

Apply **Mutation** operator;

Evaluate created solutions;

End While;

2.2 Ant Colony Optimization

The ant colony optimization algorithm is an agents based metaheuristic that is used to find solutions to various optimization problems. Agents are called artificial ants. These algorithms were inspired by the behavior of ants and constitute with other swarm inspired algorithms a distinct family of optimization metaheuristics [13].

The algorithm is a result of the close observation of the behavior of social insects in general and ants in particular. The ants are social insects and they have a collective behavior aimed towards the good of the colony. Every ant is independent and communicates with other ants through highly volatile chemicals called pheromone [14].

Ants are very sensitive to these substances and they use them to mark their path, to be discovered by their nest mates. The most common example is the journey from the nest to food sources, upon discovery of the source; it deposits pheromones on the ground via its abdominal glands. Each new ant can easily follow this odorous trail that will guide it to the food source, and subsequently to the nest. Each individual is thus indirectly and collectively helped by the community [15].

Several ACO variants are available. The first ACO to appear is Ant System (AS) by Dorigo et al. [16]. Formally, An ant k positioned on city i at time t will choose the next city j according to the visibility η of this city and the amount of deposited pheromone τ on the arc connecting these two cities, other algorithmic variants use different pheromone deposit practices [7][15]. The selection of the next city is made stochastically, with a probability of selecting the city j as follow :

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, \forall c_{ij} \in N(s^p),$$

Hereby, the pseudo code of the Algorithm:

Initialize pheromone trails ;

While : not termination do :

Construct : the construction is done by ants travelling through the graph;

Daemon activities : optional use of a local search heuristic ;

Update pheromone trails, locally or globally;

End While;

2.3 The Travelling Salesman Problem

The Traveling Salesman Problem is one of the most studied problems in computational mathematics. It is an NP-hard problem and due to its easy to use formalism, it is often used as a benchmark for many optimization methods [17]. The problem is as follow, given a set of cities and the distances between them, what is the shortest possible route that visits each city exactly once and returns to the origin city (least total distance).

The problem can be formulated as a weighted graph, where edges are the cities, and the arc costs are the distances between them. A TSP can be symmetric, it is the general form of a TSP where the distance from city i to city j is the same as going from j to i. the weighted graph is said to be undirected. Another form of the TSP is the asymmetric or directed weighed graph, where arcs may not exist or have different weights in the two directions. The ATSP can be found in several real world applications mainly in logistics, planning and DNA sequencing [2][17][18].

To test the algorithmic approaches we decided to run several experiments, varying the complexity and the type of the TSP. The two algorithms are tested on three benchmark instances and one real word asymmetric problem. Most of the benchmark problems can be found in TSPLIB: http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html. The benchmark TSPs are Berlin52, Eil76, and A280. The real world application is Casablanca40 [19][20].

TSPLIB offers problem coordinates, best tour and the optimum solutions for the three test problems [19]. Casablanca40 is different; the problem is asymmetric and respects geographical configuration and city architecture [20]. The city is economically and demographically the largest of Morocco and is characterized with high traffic flows and surfacing transportation problems.

3. RESULTS AND DISCUSSIONS

All the simulations were completed on a Windows 64 bits personal computer with an i5-3470 processor clocked at 3.20GHz, and 4 GB of Ram. The Genetic algorithm was developed in C++ using the GAlib [21]. The Ant colony optimization algorithm was written in native C#.

3.1 The Genetic Algorithm

3.1.1 Berlin52

The first instance to test the GA is Berlin52 with 52 locations in Berlin (Groetschel) [19]. The Genetic Algorithm finds the optimal tour in 3.485 seconds. Fig 1 presents the evolution of the objective function (here distance) over time. Fig 8 (a) shows the found tour overlaying the instance optimal tour, both tours are 7542.

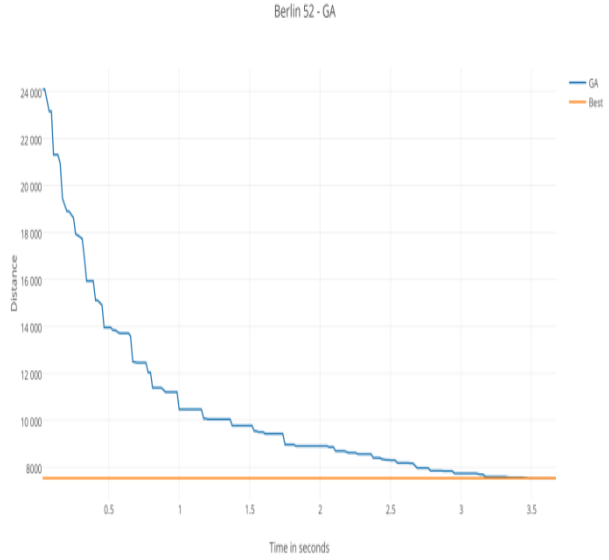


Fig1- Evolution of GA results over time in Berlin52

3.1.2 Eil76

The second instance is Eil76 with 76 cities. it is considered a large TSP instance[19]. The GA gives satisfying results but doesn't find the optimal solution. Fig 8 (b) illustrates the differences between the optimal tour of 538 and the GA tour. The convergence time is considerably short. The GA returns a best tour of 570 in 13.441 seconds. With an approximate error of 5.9479%.

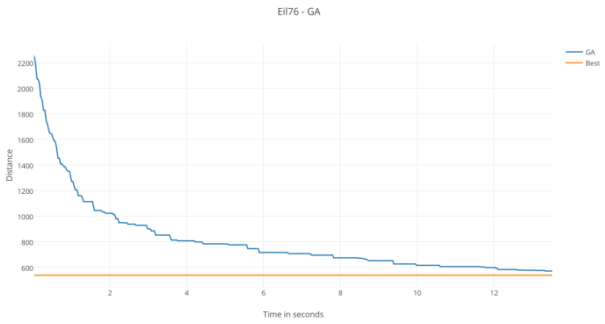


Fig2- Evolution of GA results over time in Eil76

3.1.3 A280

The largest instance in this paper has 280 cities, its optimal tour is 2579 [19]. Due to its complexity, this instance requires more computational time and a precise tuning. The GA handles this instance with a steady improvement factor as seen in Fig 3 and returns a best tour of 3218 in 517.965 seconds. With an error rate of 24.777%.

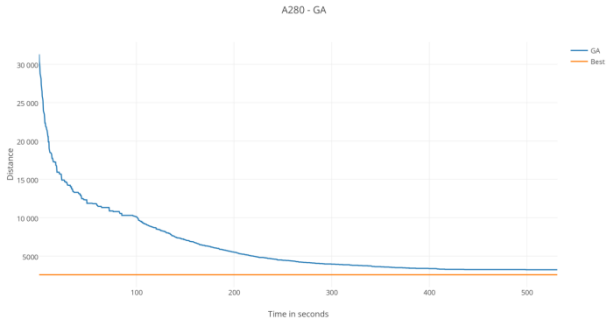


Fig3- Evolution of GA results over time in A280

3.1.4 Casablanca40

This instance is a special case of the TSP applied to logistics, more specifically urban transportation [20]. The routes are presented as a directed weighted disjunctive graph; the result will be a single tour that passes by all the 40 markers of the problem. The disjunctive graph has 1560 arcs connecting 40 edges. The total cumulated distance of the arcs is 16273.18 Kilometers. The solution space contains theoretically 2.04E+46 possible solutions. The GA returns a best tour of 125.974 Kilometers in less than 24.759 seconds. The returned tour is illustrated in Fig 5.

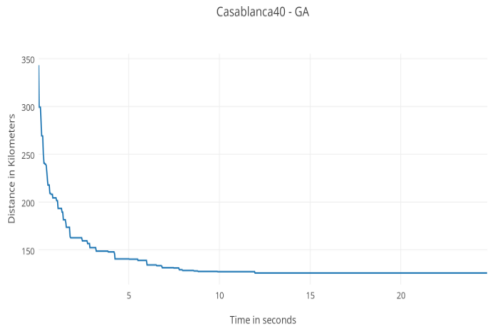


Fig4- Evolution of GA results over time in Casablanca40

Fig5- A map visualization of the tour returned by the GA

3.2 The Ant Colony Optimization

3.2.1 Berlin52

The ACO converges quickly but doesn't find the optimal tour. It finds a tour of length 7575 in 17.727, with an error rate of 0.4375% from the optimal tour. Fig 6 shows the evolution of the objective function over time. The evolution can be separated in two main phases: The first phase where the algorithm converges quickly and finds a solution of distance 7680 in less than 3.568 seconds, this behavior is close to that of the GA. The second phase where there is no improvement until seconds 16.619 and 17.727 with respectively minor improvements of 0.2218% and 1.1617%.

This second phase is due to the high pheromone deposit on

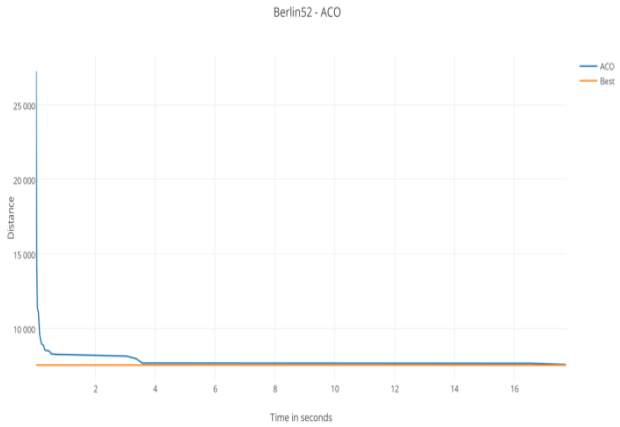


Fig6- Evolution of ACO results over time in Berlin52

trails that leads to early convergence followed by a phase of stagnation. The late minor improvements can be explained by pheromone evaporation that led to a late diversification of the discovered paths.

3.2.2 Eil76

Eil76 optimal tour is 538. The ACO maintains the same two phases behavior; it converges quickly to a tour of 588 in 4.886 seconds. With an error rate of 9.2936%. Then returns an overall best tour of 551 after 73.906 seconds with an error rate of 2.4163%. Fig 8 (b) illustrates the performance of the ACO in providing a near optimal tour.

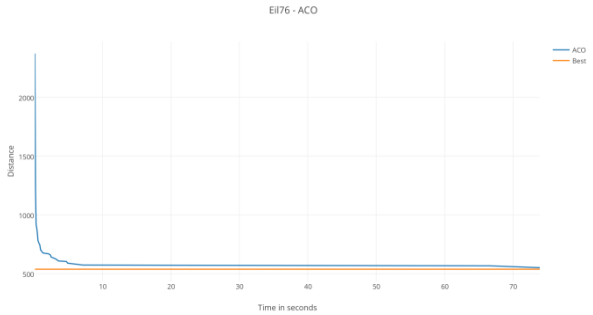
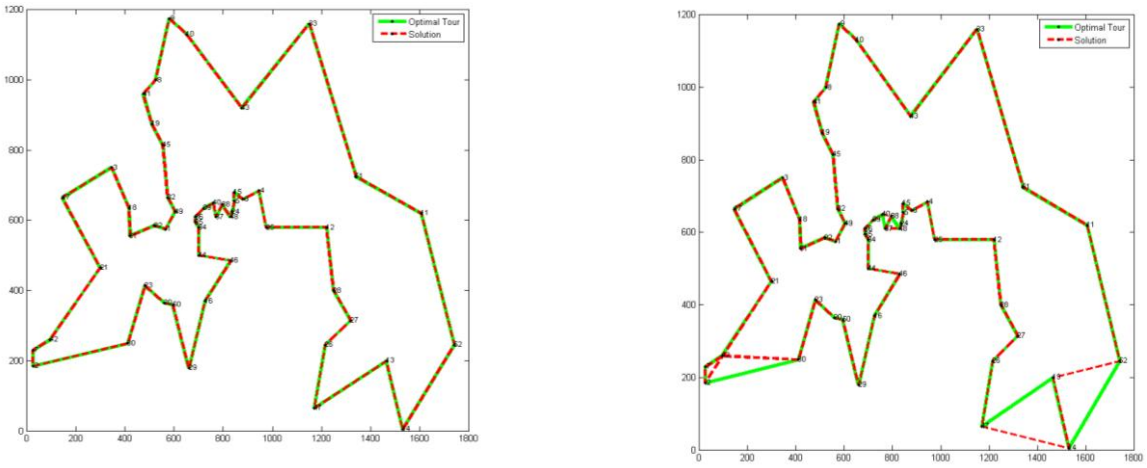
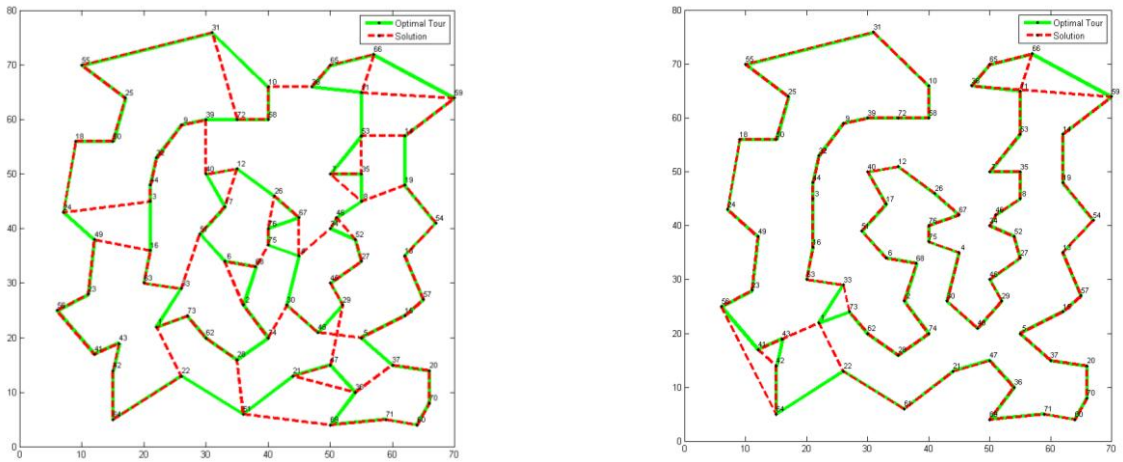


Fig7- Evolution of ACO results over time in Eil76

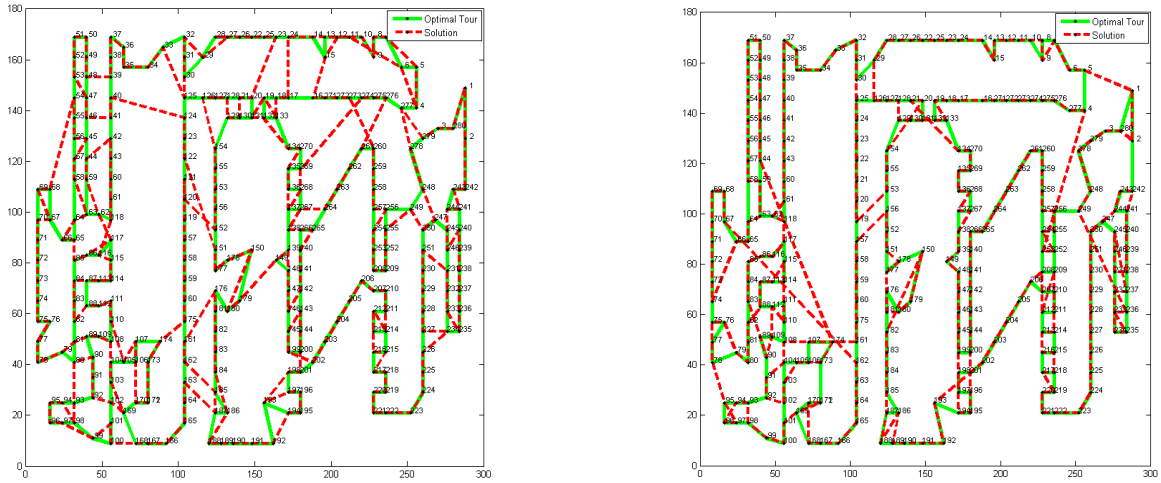
GA **ACO**



a- Berlin52



b- Eil76



c- A280

Fig8- GA and ACO results compared to optimal tours

3.2.3 A280

This instance is large and the behavior of the ACO slightly changed. A middle phase can be observed between seconds 45.717 and 323.242. This phase is characterized with a slow but steady improvement ratio and the absence of any stagnation behaviors.

The runtime ends returning the best tour value 3092 in 767.711 seconds with an error rate of 19.8914%.

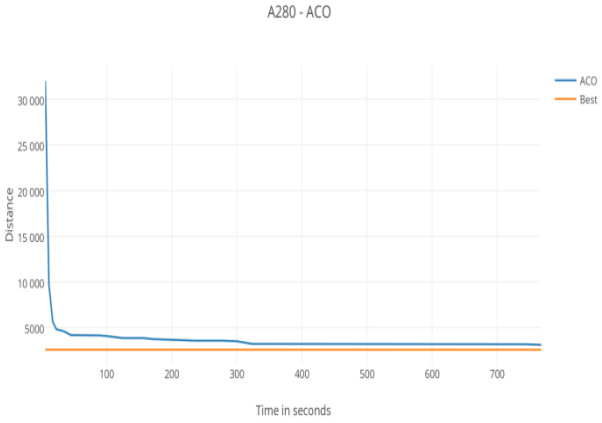


Fig9- Evolution of ACO results over time in A280

3.2.4 Casablanca40

The last instance to test the ACO with is Casablanca40 [20]. There is no mathematically proven optimal tour of this real world problem, the best actual tour we have is the one provided by the GA previously. ACO terminates really fast and reaches a better solution of distance 125.216 in 20.618 seconds. The ACO is qualitatively 0.6053% better, and computationally 20.0843% faster. Actual best tour of Casablanca40 is shown in Fig11.

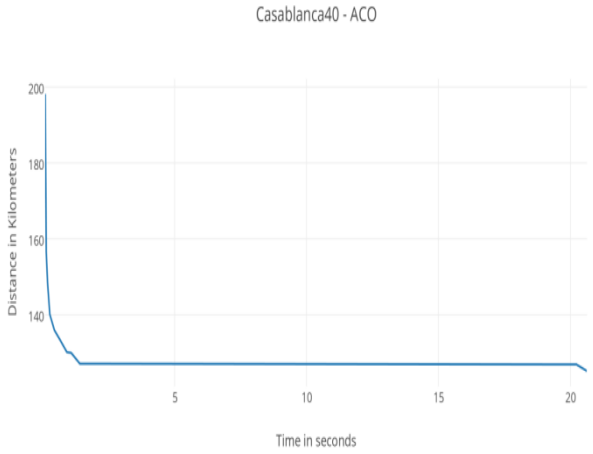


Fig10- Evolution of ACO results over time in Casablanca40

Fig11- A map visualization of the tour returned by the ACO

TABLE 1. OVERVIEW OF SIMULATION RESULTS

Instance		GA			ACO		
Name	**Best**	**Tour**	**Error rate %**	**Time seconds**	**Tour**	**Error rate %**	**Time seconds**
Berlin52	7542	7542	0.0	3.673	7575	0.4375	17.727
Eil76	538	570	5.9479	13.519	551	2.4163	73.906
A280	2579	3218	24.7770	531.087	3092	19.8914	767.711
Name		**Tour**		**Time seconds**	**Tour**		**Time seconds**
Casablanca40		125.974		24.759	125.216		20.618

3.3 Comparison of the results

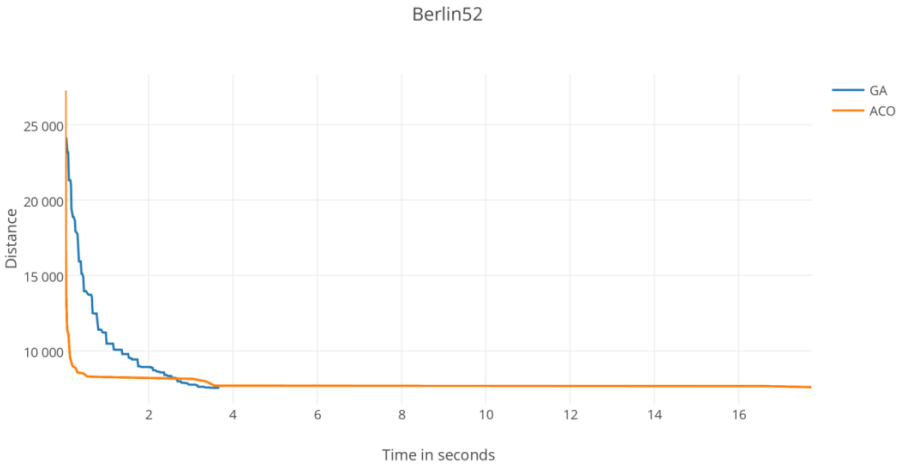
Qualitative and quantitative comparisons were made to understand these two algorithmic approaches. Both algorithms give good results in handling this NP-hard optimization problem. They are both efficient in attacking all instances of the TSP. Where the search space is large, complex and poorly known and no prior mathematical analysis was given. Table 1 shows the computational results.

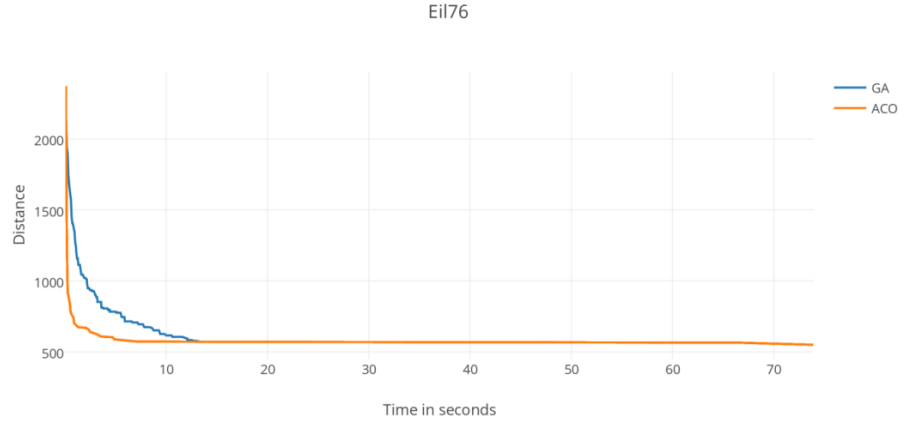
Both algorithms provide satisfying results over the four instances; they show adequate performance over small instances and robustness over medium and large ones.

The GA is excellent with small to medium size instances; without any prior knowledge of the solution space or the help of any local search routine, it ensures rapid convergence and finding of optimal solutions. Through all the instances, the algorithm is robust and maintains a steady improvement behavior, it always gives solutions and solutions get better with time.

The ACO gives better results in large instances. Even with its significant performance compared to GA. It is prone to falling in the local optima because it updates the pheromone according to the current best path [13]. But given enough time for the pheromones to evaporate as seen in Fig13, The ACO ensures finding better results than the GA. And the convergence speed is notable in early iterations.

.





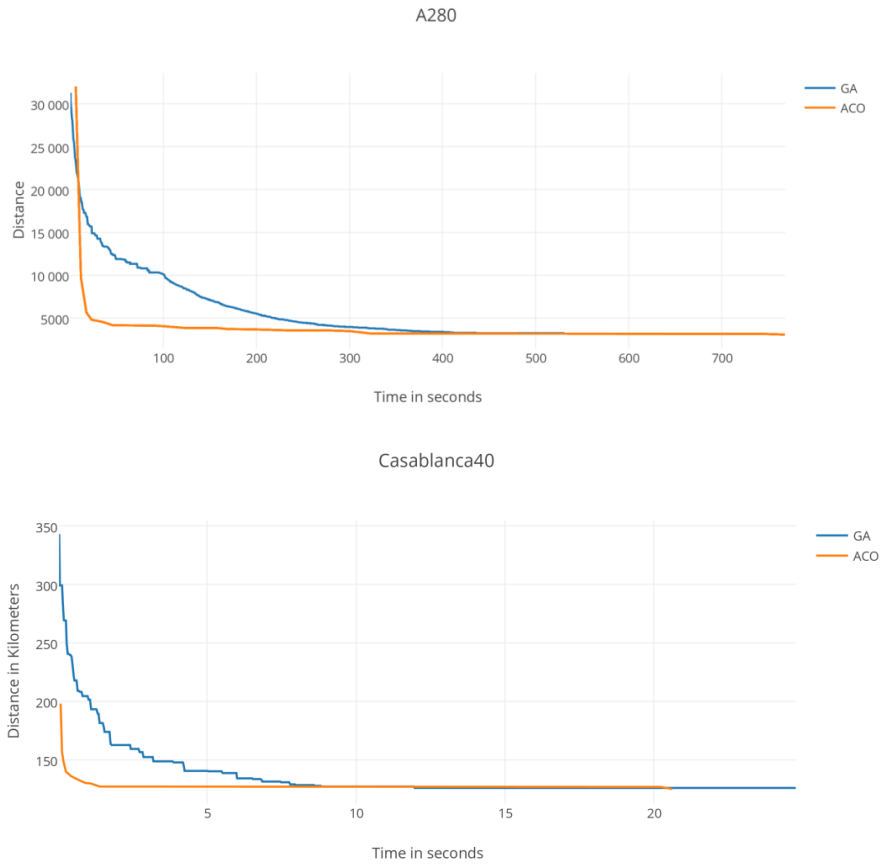


Fig12– Overlaid computational results of GA and ACO

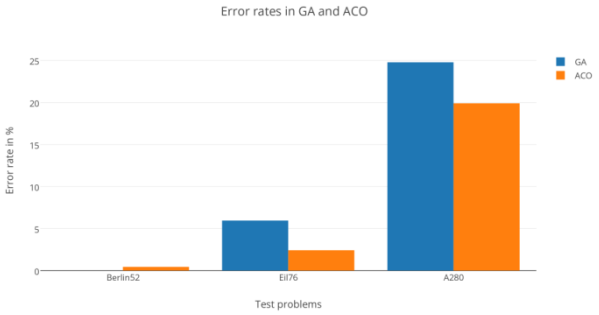


Fig13- Error rates in GA and ACO

When it comes to computational cost, a close comparison through Fig12 shows that ACO converges earlier and maintains an explorative behavior throughout the available runtime. GA guarantees a steady improvement behavior and once it stagnates, it leaves the processor fruitless. During the active runtime, ACO is greedier due to its distributed processing nature [16].

Fig13 shows the error rate of the algorithms, Casablanca40 results were omitted because the problem is a real world logistics problem with no mathematically proven optimal tour, and the two approaches returned nearly the same tours.

It is clear that error rate increases with problem complexity. When the number of cities is less, GA provides better results, when the number of cities and solution space is larger ACO outperforms GA in the quality of the results. The difference in error rates between the two algorithms is higher with larger problems, thus advantaging the ACO.

4. CONCLUSION AND FUTURE SCOPE

This paper presented a comparison of the qualitative and computational performances of two famous metaheuristics: GA and ACO. While both algorithms provided satisfactory results all over the test problems a few remarks can be outlined.

The GA is fast, easy to implement and cost efficient in terms of computational resources. The ACO is greedier but provides better results, particularly with large problems.

Based on our simulation results we recommend:

∴ GA for *low computational resources* and/or small to medium size problems. Suitable for rapid finding of optimum solutions. Is easy to prototype and implement.

∴ ACO for *greater computation resources* and/or large and complex problems. Suitable for finding high quality solutions in a noticeable runtime. Is distributed, easy to hybrid and accommodate with other algorithms.

The two algorithmic approaches have a great potential in solving many TSP based real world applications ranging from logistics to manufacturing and robotics. Future scope has been focused to finding better performing derivatives and hybridations of these algorithms.

The algorithms are very sensitive to parameters variation. In the present work we rely on the authors experience to fine

tune the algorithms in order to provide the best results. Future work can be focused to automatic parameters adaptation routines based on problem type and other accessible data.

5. REFERENCES

[1] K. Menger, Ergebnisse eines mathematischen Kolloquiums: Deuticke, 1932.

[2] 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.

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

[4] P. Siarry, "La méthode du recuit simulé: théorie et applications," Automatique-productique informatique industrielle, vol. 29, pp. 535-561, 1995.

[5] H. El Hassani, J. Benhra, and S. Benkachcha, "Utilisation des algorithmes génétiques (AG) dans l'Optimisation multi-objectif en logistique avec prise en compte de l'aspect environnemental (émissions du CO2)," in Colloque international LOGISTIQUA, RABAT, 2012.

[6] TALBI, El-Ghazali. Metaheuristics: from design to implementation. John Wiley & Sons, 2009.

[7] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," IEEE Transactions on Evolutionary Computation, vol. 1, pp. 53-66, 1997.

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

[9] L. J. Fogel, A. J. Owens, and M. J. Walsh, Artificial Intelligence Through Simulated Evolution: John Wiley & Sons, 1966

[10] D. B. Fogel, "The evolution of intelligent decision making in gaming," Cybernetics and Systems, vol. 22, pp. 223-236, 1991.

[11] L. D. Davis, Handbook Of Genetic Algorithms, 1 ed.: Van Nostrand Reinhold;, 1991.

[12] M. Vazquez and L. D. Whitley, "A Hybrid Genetic Algorithm for the Quadratic Assignment Problem," in GECCO, 2000, pp. 135-142.

[13] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 26, pp. 29-41, 1996.

[14] J.-L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels, "The self-organizing exploratory pattern of the argentine ant," Journal of insect behavior, vol. 3, pp. 159-168, 1990.

[15] T. Stützle and H. H. Hoos, "MAX–MIN ant system," Future generation computer systems, vol. 16, pp. 889-914, 2000.

[16] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in Proceedings of the first European conference on artificial life, 1991, pp. 134-142.

[17] G. Gutin and A. P. Punnen, The traveling salesman problem and its variations vol. 12: Springer Science & Business Media, 2002.

[18] SHIN, Soo-Yong, LEE, In-Hee, KIM, Dongmin, et al. Multiobjective evolutionary optimization of DNA sequences for reliable DNA computing. Evolutionary Computation, IEEE Transactions on, 2005, vol. 9, no 2, p. 143-158.

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

[20] 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.

[21] WALL, Matthew. GAlib: A C++ library of genetic algorithm components. Mechanical Engineering Department, Massachusetts Institute of Technology, 1996, vol. 87, p. 54.