Analysis of Meta-heuristic Algorithms to Solve NP Hard Problems

GROUP 2

Emilio Marinone marinone@kth.se

Parker Kruzel kruzel@kth.se

Panagiotis Kotsias pcko@kth.se

Tanay Rastogi tanay@kth.se

Abstract

Hard optimization problems are those that do not guarantee optimal solution in polynomial time so that a near optimal solution can be found using heuristic algorithms. In the paper, the authors analyze two all-purpose meta-heuristic algorithms called Simulated Annealing (SA) and Genetic Algorithm (GA), on a classical NP-complete combinatorial optimization problem called Traveling Salesman Problem (TSP).

Contents

1 Introduction **3**
1.1 Contribution . . . 3
1.1.1 Traveling Salesman Problem . . . 4
1.1.2 Genetic Algorithm . . . 4
1.1.3 Simulated Annealing . . . 5
1.2 Outline . . . 5

2 Related work **5**

3 Method **6**
3.1 Genetic Algorithm . . . 7
3.2 Simulated Annealing . . . 8
3.3 Implementation . . . 8

4 Experiment **9**
4.1 Experimental results . . . 9

5 Summary and Conclusions **14**

1 Introduction

Combinatorial Optimization is a set of problems in applied mathematics and computer science where an optimal solution should be searched by satisfying certain constraints, within a set of possible solutions. These problems can be expressed using concepts like set theory, subsets, permutation/combination, and graph or tree search. They are divided into two general categories: Exact and Heuristics. In exact problems, the solution can be found in polynomial time using just simple brute force search. But this becomes increasingly difficult as the number of parameters in states and constraints increases. In this kind of problems, brute force search becomes infeasible. Hard NP-complete problems in combinatorial optimization, which cannot be solved in polynomial time, are usually tackled using heuristic techniques. They do not guarantee an optimum solution, but a near-optimum solution is found in a reasonable amount of time. The heuristic functions are mostly problem-dependent. They are usually made in accordance to the problem at hand and try to take full advantage of the particularities of the problem to find a near-optimum solution. Hence, they are focused on the exploitation of the solution space. However, they are often greedy approaches and lack in an exploration of solution space. These techniques have a high susceptibility to getting stuck in local optimum, thus failing to find the global one. A meta-heuristic is a higher level of heuristic, designed to solve optimization problems. These heuristics are different in a manner that they are problem independent techniques. They do not take advantage of the particularities of the problem and work as all-purpose black boxes. Meta-heuristics are designed in a way to have a balance between exploitation and exploration of the given solution space. These techniques are not greedy and try to search in complete solution space by accepting bad solutions. In this way, they do not get stuck in local optimum and have better chances to reach the perfect solution.

1.1 Contribution

In the paper, the authors analyze and compare two of the most popular metaphor based meta-heuristic algorithms called Simulated Annealing (SA) and Genetic Algorithm (GA) in a classical combinatorial optimization problem called Traveling Salesman Problem (TSP), which aims to visit a set of places minimizing cost, distance or a combination of those.

1.1.1 Traveling Salesman Problem

A Traveling Salesman Problem is one of the most widely studied combinatorial optimization problems and is a benchmark for computer research. For the purpose of the analysis, a route map for several cities in Europe is planned. The problem statement can be stated as [10].

Input: Let *G = (V, A)* be a graph where V is a set of n vertices representing cities in the map of Europe, and A is a set of arcs or edges representing direct route from one city to another. Let

$$C = c_{ij}$$

be the cost matrix associated with A having either the distance or the price between the two cities. The C matrix is symmetrical, such that

$$c_{ij} = c_{ji}$$

for *i,j* ϵ *V*. Also C satisfies the triangle inequality if and only if

$$c_{ij} + c_{jk} \geq c_{ik}$$

for *i,j,k* ϵV.

Output: A list of cities $V = v_1, v_2, v_3..., v_n$ such that total cost

$$J = \sum_{i,j}^{n} c_{ij} * v_{ij}$$

is minimum and each city is visited once and only once except for the initial one. For n vertices TSP, the total number possible solutions are n! , where not necessarly only one solution gives the minimum cost.

A large number of real-world problems can be modeled as a version of TSP problems like computer wiring, hole-punching, vehicle routing, job sequencing, dartboard design, circuit board designing, crystallography and others.

1.1.2 Genetic Algorithm

Genetic Algorithm (GA) is a meta-heuristic algorithm that is inspired from the nature belonging to a larger class of evolutionary algorithms [2]. It is a population based meta-heuristics that do a global search in the solution space, hence taking advantage of both the exploration and exploitation to find the near optimal solution satisfying given constrains. Also, they are robust to noise thence output does not change with slight change in the input.

GA mimics the concept of survival of the fittest by generating a population of the different solutions and then accepting only those that perform best.

1.1.3 Simulated Annealing

Simulated Annealing is a meta-heuristic algorithm based on the concept of physical annealing of solids and it has been introduced in 1983 by [12]. The key concept consists on generating neighbor random solutions, where neighbor means they slightly vary from their previous one, at a constant parameter representing temperature. Then, better solutions are always accepted meanwhile worse ones are accepted according to a rule called Metropolis, [2] which represents an acceptance probability. This process is iterated until the system is considered in thermal equilibrium [9], then the temperature is slowly decreased according a so-called cooling schedule and it repeats the previous step until a new equilibrium is reached. Acceptance of worse solutions lets the system escaping from local optimum, to which otherwise the system would converge depending on the initial solution.

1.2 Outline

In the report the authors present the solution for TSP solved using two algorithms. The related work done and the motivation for the project is briefly described in Section 2. The algorithms and methods to solve TSP are discussed in Section 3. In the Section 3.3,implementation of the TSP using MATLAB and World Map is described. Following that, the experiment description and the results are described in Section 4 and Section 4.1. Finally the summary of the analysis is presented in Section 5.

2 Related work

The report presents The Traveling Salesman Problem which is a classical combinatorial NP-complete problem and has been supported by an extensive amount of scientific work where different kind of solutions are presented. [14] presents exact algorithms for NP-Hard problems like dynamic programming across the subsets, pruning the search tree and local search since they can be solved by exhaustive search. However, the paper concludes that these solutions are not feasible as the number of cities grows because, in the worst case, they would require forming a super-polynomial up to exponential time. For this reason, researchers started implementing heuristics to find a near-optimum solution in polynomial time. Unfortunately, [15] it is proven by a number of "no free lunch" theorems that there is no heuristic algorithm which performs better than all the others in all the cases. Therefore, for each different NP-Complete problem, the best heuristic needs to be figured out. This is not a limitation for meta-heuristics like Scatter Search, Genetic Algorithm, Ant Colony Optimizations, Iterated Local Search, Simulated Annealing and so on since they are problem independent. [5] presents advantages and drawbacks of each one, about which thousands of papers have been published. Since in this one, the authors will describe and compare GA and SA, some related works are now presented. According to [6], a knowledge-based GA is capable of identifying an optimum or near-optimum path even in dynamic environments, that is computationally efficient to be used in most real-world applications. Genetic Algorithm is particularly applied to mobile robot path planning applications. In [11], it is shown that a GA-based approach to this problem results in a path that converges to the optimal solution much faster than a Lazy A* algorithm, even in environments of high complexity. A very important part of any GA-based algorithm is the so-called crossover operator, i.e. the mixture strategy between a set of parents in order to produce a better child. In fact, it is still under discussion which operator results in the best outcome. In [7], eight different operators are checked for the case of the Traveling Salesman Problem and the Sequential Constructive Crossover Operator (CSX) is shown to have the lowest error with respect to the known optimal solution. The aforementioned Sequential Constructive Crossover Operator was initially presented by Z. H. Ahmed [1].

In case of Simulated Annealing, implementation of the basic algorithm is straightforward since it just requires a random function which creates a neighbour solution, an acceptance probability directly proportional to the temperature and inversely proportional to the difference between the current and the future cost and a simple cooling schedule. However, its parameters tuning might not be obvious according to [12]. Nonetheless, the SA is used in many application because execution of the algorithm takes polynomial time since the sizes of the neighbourhoods can be chosen polynomial as shown by [8]. Since the temperature decreasing must be extremely slow due to local searching, different faster solutions have been implemented passing from the classical Boltzmann distribution to a Cauchy distribution for the choice of the neighbourhood allowing temperature to linearly decrease with respect to the number of iterations [13]. As expected, its implementation becomes much harder but performs better.

3 Method

This section describes the basic algorithm for the Genetic Algorithm and Simulated Annealing.

3.1 Genetic Algorithm

For running GA for TSP, each city in the map is considered as a gene. A fully connected graph is created by connecting each city to others and edge cost is considered as the distance between the two cities. Genetic Algorithm involves six basic steps:

1. **Initialization**: In order to run the algorithm, a set of possible chromosomes are randomly created, where each gene is a permutation of all the possible cities to visit.

2. **Evaluation**: For the evaluation of the best route, the fitness function is employed. For the chromosome with p genes, the fitness function is given by, $F(p) = \alpha * distance(p) + (1 - \alpha) * cost(p)$ where the parameter alpha can be {0,1}.

3. **Selection**: For the selection of the best chromosome in the population, the population is sorted in ascending order and then a percentage of best population is chosen for crossover.

4. **Crossover**:For the crossover, two different type of functions, called Order Crossover (OX) and Sequential Constructive Crossover (SCX) are implemented.

   Order Crossover: Order crossover (OX), first proposed by Davis in 1985[4] for Job scheduling using GA. The OX takes two parents to reproduce a child, such that the child has properties from both the parents.

   Sequential Constructive Crossover: The Sequential Crossover Operator (SCX) aims at identifying the best characteristics of the best parents of a given GA population, isolate them and induce them to the child that is to be formed. The best characteristics are in the sense of an allele of two subsequent genes that, compared against a defined cost-matrix (transition matrix), leads to the minimum possible accumulated cost for transiting from one gene to the next one in a chromosome. This leads to inheriting the best edges of the parents to the child.

5. **Mutation**: In order to help GA escape the local minimum and explore solution space, mutation to the population is done on a predefined percentage of the population. For the mutation, a Reciprocal Exchange Mutation (RX) mutation is done. It is a bit-wise exchange technique where in each chromosome, two randomly selected genes are swapped. Mutation is done when the fitness of population remains constant for certain generation, allowing the GA to explore the solution space.

6. **Repeat**: Until the generation limit is reached or the algorithm is converged to the optimal solution.

3.2 Simulated Annealing

For running SA for TSP, a random sequence of cities gives the initial solution to which an overall cost representing distance is associated. Keep in mind that the first and the last city must coincide. Now, a new neighbor solution is generated swapping two or a few more cities based on the previous solution and the number of neighbors. Then, the cost of the new solution is evaluated given the cities coordinates and a function which provides spheric distance given latitude and longitude of two cities. Now, an acceptance probability function decides the probability to accept a new worst solution according to the cost difference and the current temperature, which is the most relevant tuning parameter. This step is essential in order to escape from local optimum. A better solution is always accepted. New solutions are iteratively created at constant temperature n times and accepted or rejected, since the system reaches a quasi-equilibrium state. n is the other relevant parameters and depends on the problem complexity. Then, the temperature is reduced according a cooling schedule given by

$$temp = \alpha \cdot temp$$

where $0.9 \leq \alpha < 1$ up to a minimum temperature or some other ending condition, like a maximum amount of iterations in which the solution does not change is reached.

3.3 Implementation

For the project, the world map of Europe created in MATLAB has been used. Each city gives a latitude and longitude to mark its location in the map. The distance matrix is created using the (lat,lon) for the cities. For the price matrix, a flight cost for direct flights from each city is used. The current best route in each iteration is then plotted over the world map. Parallel, the cost of best Δ cost solution found in each iteration is also plotted.

Both SA and GA are tested in three test cases with different number of cities to visit for solving TSP.
Case1: European Cities Short with Number of cities - 10
Case2: European Cities Medium with Number of cities - 20
Case3: European Cities Big with Number of cities - 41

4 Experiment

The algorithm was applied to test cases of cities with different parameters. A total of 10 tests were performed on each case. The algorithms were tested and compared against the iterations at which each algorithm converged as well as the final value of the accumulated traveled distance. The maximum runtime for both the algorithms has setted at 300 seconds and the solutions are compared.
For Genetic Algorithm, the parameters for which TSP is tested are:

- Population: {Small:500, Big:10,000}
- Generation: 200
- Crossover Percentage: 30
- Mutation Percentage: 100
- Crossover Type: {SCX,OX1}

For Simulated Annealing, the parameters on which TSP is tested are:

- Cooling rate: $\alpha = 0.999$
- Initial temperature: 100
- Final Temperature: 1
- Iterations at constant temperature: 100
- Convergence (iterations with no change): {Short:10, Medium:20, Big: 100}

4.1 Experimental results

Figure 1 shows the best solution of each test case of TSP solved with the GA. For Big Map (1a), the optimal solution is only found when running on large population, and does not always converge. For Medium Map (1b), the optimal solution is reached much more frequently and the GA showed similar result even when running on small population. For Short Map (1c), GA always converges with big as well as small population.

Table 1 presents the final summary of the tests on Genetic Algorithm on Big map using both SCX and OX1. For the Medium and Short Map, both the crossover behaves very similar on all test cases because of small number of cities and algorithm manages to generate optimal solution.

As it can be seen in the Table 1, with small population the GA struggles to converge to find an optimal solution. Because of the inherent randomness in the algorithm, it gives different solution on different runs. It is also observed that the SCX operator performs better than OX1 for small population. It converges faster then OX1 and also was able to find solution nearer to optimum.

When using a bigger population, both OX1 and SCX were able to find the optimal solution. As increasing the population increase the solution space algorithm is searching on. However in this case, OX1 performs better than SCX and was able to find the solution much more frequently. The reason is the Reciprocal Exchange Mutation operator. Because of how both SCX and OX1 operator work, bit-wise change helps OX1 to escape local optimum in much better manner than SCX.

European_city_Big								
	Convergence	Value	Convergence	Value	Convergence	Value	Convergence	Value
	Population = 500 Generation = 200 Crossover Percentage = 30 Mutation Percentage = 100				Population = 10000 Generation = 200 Crossover Percentage = 30 Mutation Percentage = 100			
	OX1		SCX		OX1		SCX	
Test1	159	18153	61	19078	146	17627	24	18026
Test2	193	19484	18	18391	96	17736	22	17736
Test3	112	18234	166	18942	127	17736	73	17916
Test4	142	17797	194	17661	182	18136	52	18026
Test5	178	18296	46	18794	128	17565	22	18026
Test6	168	18918	88	18204	147	17565	36	18026
Test7	160	19537	42	19294	112	17565	36	18026
Test8	140	18921	42	19804	147	17830	57	17588
Test9	191	19306	94	19928	118	17565	40	17565
Test10	153	17916	105	19616	142	18023	36	17934

Table 1: Test summary for the GA on Big Map

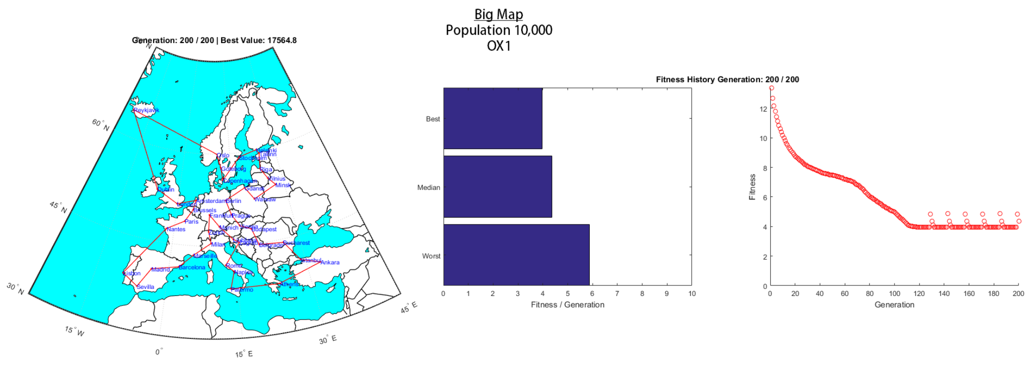
Figure 2 shows the best solution of each test case of TSP solved with the SA. For Big Map (2a), the algorithm falls in different local optimum quite close to the global one, regardless of the initial temperature and the cooling rate. For Medium Map (2b), the optimal solution is reached with the given parameters and also with lower initial temperatures ($Temp_{init} = 50$), and it fails only when the temperature decreases too fast ($\alpha < 0.99$). For Short Map (2c), SA always converges.

Table2 presents the final summary of the tests on Simulated Annealing on Big map since converges around different local optimum. For the Medium and Short Map, the same optimal solution is reached hence is not reported. For big sets of cities, the basic Simulated Annealing can converge to the global optimum but very often it gets trapped around a local optimum close to the global one. This difference depends on the random nature of the algorithm. In both cases, the algorithm is very slow since it requires many iterations evaluating the cost of the new random solution at each quasi-equilibrium with constant temperature. It is known that with logarithmic cooling it reaches the global optimum in infinite time, but this schedule is not applicable. To get good results, variations of the SA based on different probability distributions need to be implemented.

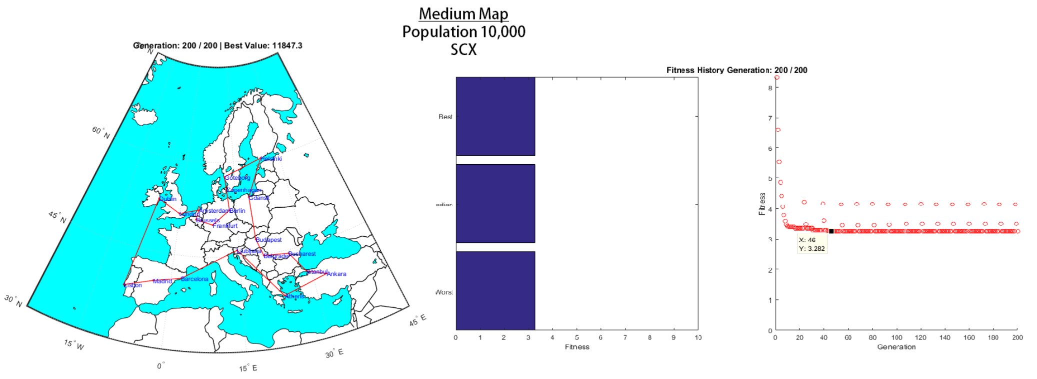
If the number of cities is not too big, it converges in a reasonable time providing the optimal solution since it does not need high initial temperature and too slow cooling. The main advantage of the basic SA is that at each iteration the algorithm stores only two solutions, therefore the memory required depends linearly on the set size. In all the cases, it is sensitive to the initial temperature since a too low one leads too small acceptance probability for worse solution and driving the solution to the closest local minimum.

European_city_Big		
	Convergence	Value
Cooling Rate = 0.99 Initial Temperature = 100 Final Temperature = 1 Iteration in Quasi-Equilibrium = 100		
Test1	96	21188
Test2	99	23378
Test3	121	24002
Test4	112	24305
Test5	71	24770
Test6	125	25001
Test7	103	25162
Test8	57	26458
Test9	110	25638
Test10	101	23701

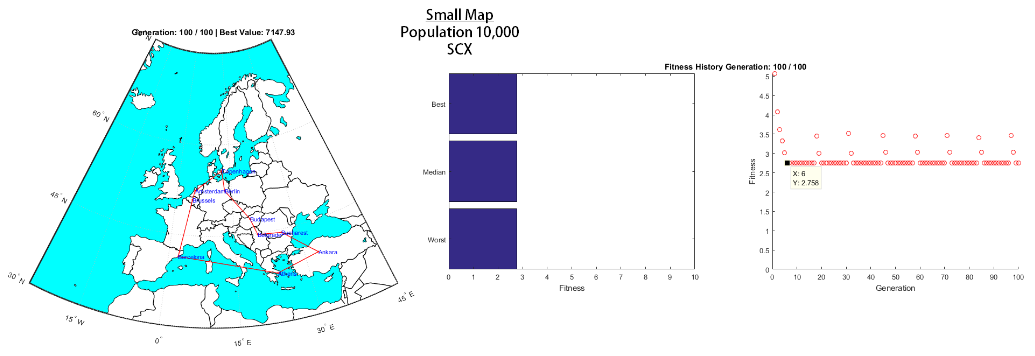
Table 2: Test summary for the SA on Big Map



(a) Big Map with OX1 crossover Final Solution: 17564.8 km

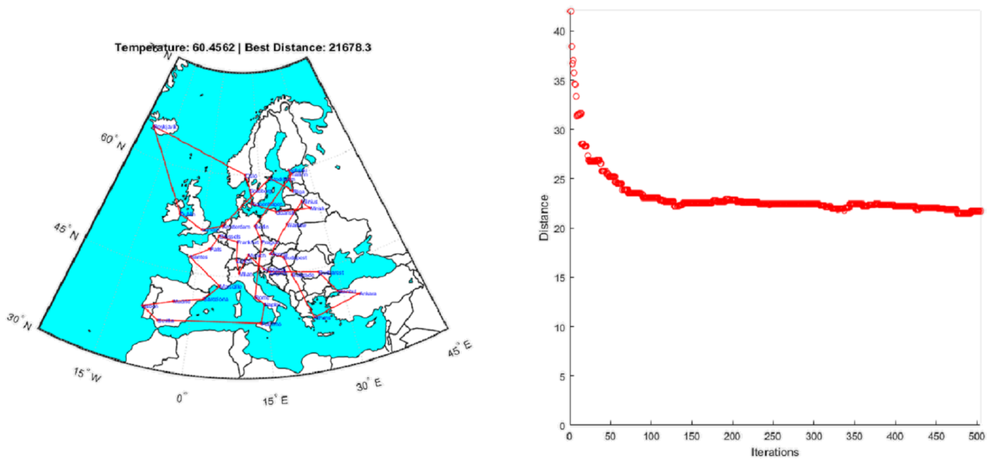


(b) Medium Map with SCX crossover Final Solution: 11847.3 km

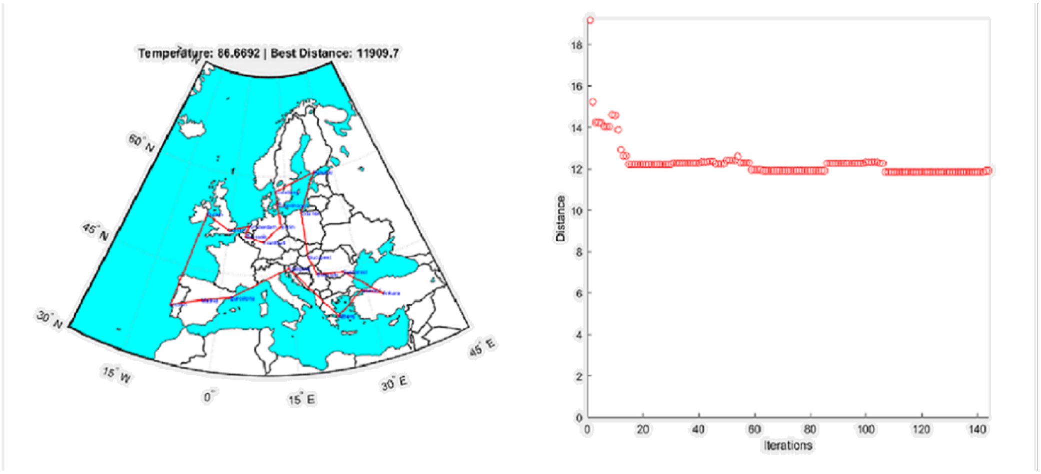


(c) Short Map with SCX crossover Final Solution: 7147.93 km

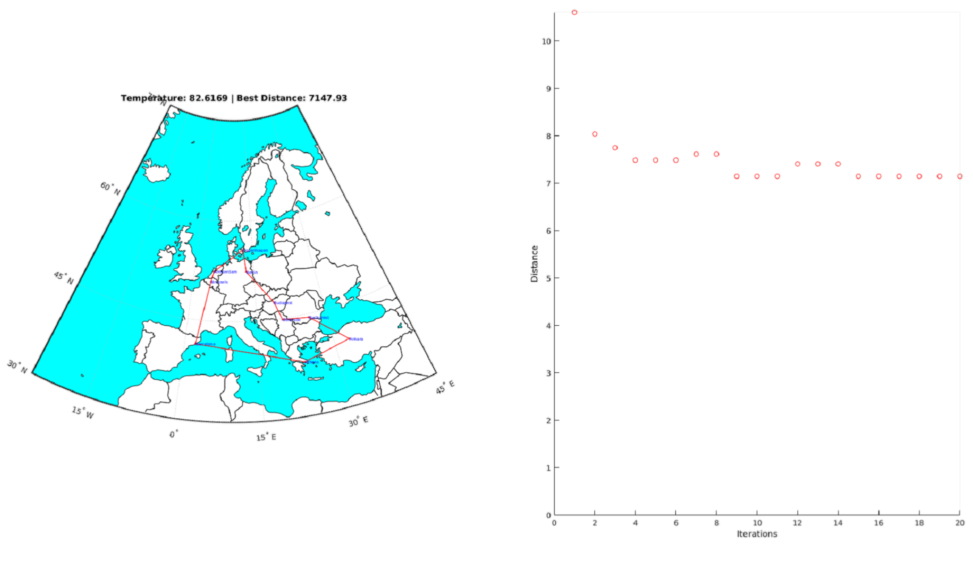
Figure 1: Best Solution for each test case solved using Genetic Algorithm



(a) Big Map Final Solution: 21678.3 km



(b) Medium Map Final Solution: 11909.7 km



(c) Short Map with SCX crossover Final Solution: 7147.93 km

Figure 2: Best Solution for each test case solved using Simulated Annealing

5 Summary and Conclusions

From the analysis, it is clear that the GA is able to find better solution and faster than the SA. In all the test case, the SA is at least ten times slower than the GA and with the big set the GA often gives a better solution and never a worse one. But SA is light on memory when compared to GA. SA stores only the best and the current solutions, therefore it can be implemented in small and light devices with little memory where computation time is not a relevant constraint. On the other hand GA is memory intensive, and needs to save all the possible solution at each iteration. As GA has better exploration property, therefore is able to reach the optimum solution.

Even though the algorithms are problem independent and flexible enough to be used in different problem scenarios, both are sensitive to their respective parameters. Both GA and SA needs to be tuned for the problem at hand.

The two different crossover operators in the GA are also compared. From the analysis, the SCX performs better than OX1 in cases when the population is small. The SCX uses the information from the problem at hand to do crossover, hence produces much better child compared to randomly generated child from OX1. SCX converges faster than OX1 and also is able to find the optimal solutions. But in case of large population OX1 performs better in finding the optimal solution. The reason behind is the way mutation is performed, which supports the OX1 better than SCX. [3]

References

[1] Zakir H. Ahmed. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator.

[2] SIMULATED ANNEALING, Emile Aarts, Jan Korst, Wil Michiels, and KORSTAND MICHIELS. Chapter 7 simulated annealing. 2007.

[3] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001.

[4] Lawrence Davis. Job shop scheduling with genetic algorithms. 1985.

[5] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics.* Springer Publishing Company, Incorporated, 2nd edition, 2010.

[6] Yanrong Hu and Simon X. Yang. A knowledge-based genetic algorithm for path-planning of a mobile robot. April 2004.

[7] Imtiaz Hussain Khan. Assessing different crossover operators for travelling salesman problem. 2015.

[8] P. J. M. Laarhoven and E. H. L. Aarts. *Performance of the simulated annealing algorithm*, pages 77–98. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[9] Delosme J.-M. Lam J. An efficient simulated annealing schedule: Derivation. 1988.

[10] Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. July 1991.

[11] Mohamed Chaalal Nouara Achour. Mobile robots path planning using genetic algorithms. 2011.

[12] Christopher C. Skiścim and Bruce L. Golden. Optimization by simulated annealing: A preliminary computational study for the tsp. In *Proceedings of the 15th Conference on Winter Simulation - Volume 2*, WSC '83, pages 523–535, Piscataway, NJ, USA, 1983. IEEE Press.

[13] Harold Szu and Ralph Hartley. Fast simulated annealing. *Physics Letters A*, 122(3):157 – 162, 1987.

[14] Gerhard J. Woeginger. *Exact Algorithms for NP-Hard Problems: A Survey*, pages 185–207. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[15] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, Apr 1997.