Solving the Travelling Salesman Problem using Simulated Annealing and Randomly Generated Solutions

Sérgio Caldas* and Filipe Oliveira†
University of Minho
Computer Science Department
*Email: *a57779@alunos.uminho.pt, †a57816@alunos.uminho.pt*

***Abstract*—Many optimization problems have several equivalent mathematical models. In high performance computer science and mathematical optimization, it is often not apparent which of these models is most suitable for practical computation, in particular, when a certain application with a specific range of instance sizes is in focus.**

Our case study addresses the Travelling Salesman "Zero Cost Solution" Problem Solving throw randomly generated solutions and using simulated annealing.

We were provided codes based on both alternatives. We injected a zero cost solution on a distance matrix, for a particular path. Computational results for real-world instances with up to 100 nodes are reported.

***Index Terms*—Travelling Salesman Problem – Randomly Generated Solutions – Simulated Annealing – Heuristics - Monte-Carlo**

1. Introduction

1.1. Traveling Salesman Problem

Given a list of cities and a matrix with the distance between a pair of cities, the TSP ask the following question: *What is the shortest possible route that visits each city exactly once and returns to the origin city?*

In the Theory of computational complexity the TSP is a problem that belongs to the class of NP-Complete problems, whose computation time for an exact solution increases with N as exp(const. N), becoming rapidly prohibitive in cost as N increases.

The traveling salesman problem also belongs to a class of minimization problems for which the objective function has many local minimum.

In practical cases, it is often enough to be able to choose from these a minimum which, even if not absolute, cannot be significantly improved upon.

1.2. Simulated Annealing

The Simulated Annealing is an adaptation of Metropolis–Hastings algorithm, that is a Monte-Carlo Method. This technique is a probabilistic technique that calculates the minimum global of a function, it is a meta-heuristic to approximate a minimum optimization for a large and discrete search space (e.g. the Traveling Salesman Problem).

For problems where finding the precise minimum optimum is less important than finding an acceptable local minimum in a fixed amount of time, simulated annealing may be preferable to alternatives such as brute-force search or gradient descent.

2. Hardware characterization

The computing platform, our team laptop, is a Intel® Core™ i7-3635QM (Ivy Bridge architecture). The system features 16 GB of DDR3 RAM, supported at a frequency of 1600MHz, divided in 2 memory channels.

The characteristics of the CPU on the computing platform is presented in the table 1.

System	team laptop
# CPUs	1
CPU	Intel® Core™ i7-3635QM
Architecture	Ivy Bridge
# Cores per CPU	4
# Threads per CPU	8
Clock Freq.	2.4 GHz
L1 Cache	128KB 32KB per core
L2 Cache	1024KB 256KB per core
L3 Cache	6144KB shared
Inst. Set Ext.	SSE4.2 & AVX
#Memory Channels	2
Vendors Announced Peak Memory BW	25.6 GB/s
Measured Peak Memory BW	15.5 GB/s

Table 1: Architectural characteristics of the evaluation platform.

3. Solving the Travelling Salesman Problem

3.1. Solving the Travelling Salesman Problem using Randomly Generated Solutions

The most common heuristics to solve the TSP problem requires to work on a randomly generated tour and to

improve the solution starting from that, until a stop condition is satisfied [1] and the best travelling solution found is returned as the final solution.

3.2. Solving the Travelling Salesman Problem using Simulated Annealing

In similarity to randomly generated solutions method we start by picking an arbitrary initial tour. From that initial tour we iterated and check random neighboring tours to see how good they are.

Due to the large set of valid possibilities it is in practical terms not feasible to test every possible solution, but a well-designed annealing process eventually reaches a solution that, if it is not the global optimum, is at least an acceptable local minimum in a fixed amount of time.

4. Results Analysis

4.1. Comparison of the two models

In a first phase we performed computational experiments with the two models of the TSP described in Sect. 3. Our aim was to gather from these test runs information that would allow us to choose a "winner", i.e., a model that, for the range of problem instances we address, displays the best computational performance in practice.

We supplied optimum solutions as input to see how the implementations behave if the best upper bound is provided (O cost for all travels).

We considered as performance metrics both the final achieved best solution and the number of iterations required to produce it. In order to gather the number of iterations we added a variable for that matter. These results are summarized in Table 2 and a visual result for each city set is presented in Appendix A in Figures A to F. Each value results from the average of 100 tests.

Nº of Cities	Avg. Cost SA	Avg. Iterations SA	Avg. Cost RGS	Avg. Iterations RGS
5	-6.0396e-16 (0)	106	-4.9738e-16 (0)	101
10	11.363	664	17.662	104
20	48.197	1309	61.03	109
40	130.47	2043	150.67	118
80	274.32	2570	322.43	129
100	342.7	2670	401.87	133
1000	3861.1	4369	5024.8	169

Table 2: gives, for each algorithm (SA - Simulated Annealing; RGS - Randomly Generated Solutions) and set of cities (5,10,20,40,100,1000), the number of generated cuts in relation to the achieved best solution.

1. in our case study we stopped if there were no changes for 100 iterations

5. Conclusion

The TSP solving formulations of both differ significantly in the number of computational effort, number of constraints and their degree of difficulty. The solution resulting from TSP-RGS (**Randomly Generated Solutions**) seem to be the smallest and easiest for the TSP solver.

The numerical problems for TSP-SA (**Simulated Annealing**) resulted in high computing efforts to solve the TSP, although achieving a better solution in terms of AVG cost for all sets of cities.

The results of our test runs made us conclude that TSP-SA Model is suited best for our particular case study, with the remark that for larger sets of cities, more experiments should be produced.

We should also remark here that our computational experience does not indicate the SA approach to the TSP is the unchallenged winning strategy, since RGS achieved good results at a portion of the computational resources needed to produce it.

As said further research has to show where the relative advantages of the different methodologies lie. Nevertheless, the algorithms described in this short paper were able to solve real–world instances of a particular application of the asymmetric travelling salesman problem in a way that is satisfactory for practice.

References

[1] R. Anthony and E. D. Reilly. *Encyclopedia of Computer Science*, 2013.

[2] A. H. G. R. E. L. Lawler, J. K. Lenstra and D. B. Shmoys. The traveling salesman problem. *John Wiley Sons, Chichester*, 1985.

[3] B. Korte. Applications of combinatorial optimization. *talk at the 13th International Mathematical Programming Symposium, Tokyo*, 1988.

[4] C. A. M. C. U. Aybars, K. Serdar and A. A. Genetic algorithm based solution of tsp on a sphere. *Mathematical and Computational Applications*, 14(3):219–228, 2009.

[5] W. R. Z. Ismail and W. Ibrahim. Travelling salesman problem for solving petrol distribution using simulated annealing. *American Journal of Applied Sciences*, 2008.

Appendix

1. Visual Results for TSP-SA and TSP-RGS for 5 Cities

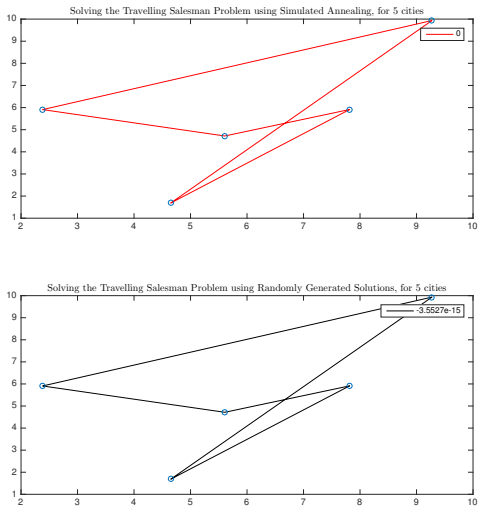

Figure 1: Visual Results for TSP-SA and TSP-RGS for 5 Cities

2. Visual Results for TSP-SA and TSP-RGS for 10 Cities

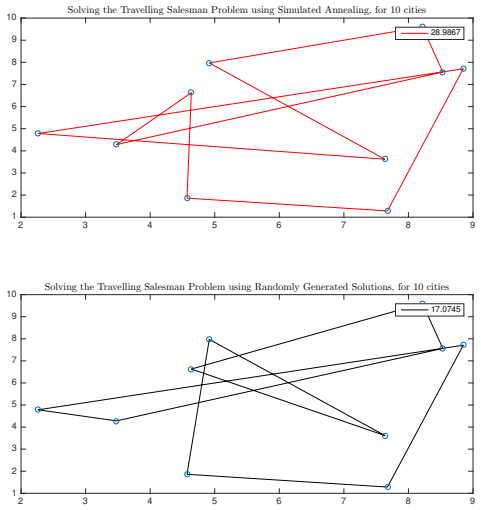

Figure 2: Visual Results for TSP-SA and TSP-RGS for 10 Cities

3. Visual Results for TSP-SA and TSP-RGS for 20 Cities

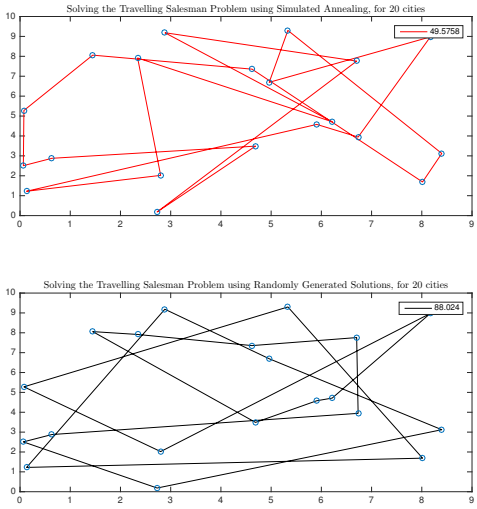

Figure 3: Visual Results for TSP-SA and TSP-RGS for 20 Cities

4. Visual Results for TSP-SA and TSP-RGS for 40 Cities

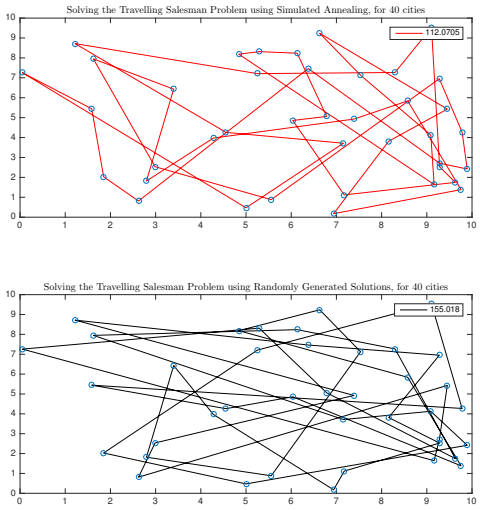

Figure 4: Visual Results for TSP-SA and TSP-RGS for 40 Cities

5. Visual Results for TSP-SA and TSP-RGS for 80 Cities

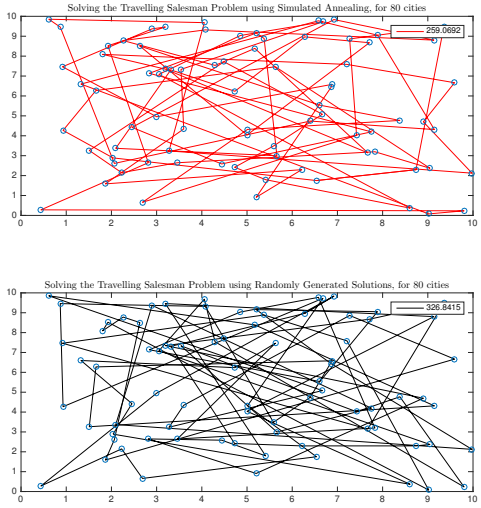

Figure 5: Visual Results for TSP-SA and TSP-RGS for 80 Cities

6. Visual Results for TSP-SA and TSP-RGS for 100 Cities

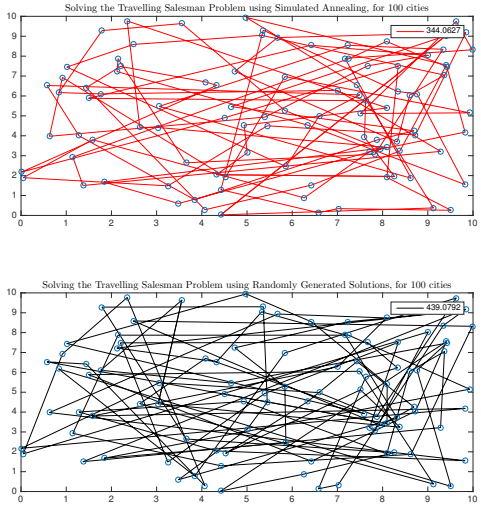

Figure 6: Visual Results for TSP-SA and TSP-RGS for 100 Cities