Revisiting Traveling Salesman Problem (TSP): Analysis of GA and SA Based Solutions

https://doi.org/10.3991/ijes.v9i2.22317

Darius Bethel, Hakki Erhan Sevil (✉)
University of West Florida, Pensacola, Florida, USA
hsevil@uwf.edu

Abstract—The purpose of this study to analyze genetic algorithm (GA) and simulated annealing (SA) based approaches applied to well-known Traveling Salesman Problem (TSP). As a NP-Hard problem, the goal of TSP is to find the shortest route possible to travel all the cities, given a set of cities and distances between cities. In order to solve the problem and achieve the optimal solution, all permutations need to be checked, which gets exponentially large as more cities are added. Our aim in this study is to provide comprehensive analysis of TSP solutions based on two methods, GA and SA, in order to find a near optimal solution for TSP. The results of the simulations show that although the SA executed with faster completion times comparing to GA, it took more iterations to find a solution. Additionally, GA solutions are significantly more accurate than SA solutions, where GA found a solution in relatively less iterations. The original contribution of this study is that GA based solution as well as SA based solution are developed to perform comprehensive parameter analysis. Further, a quantifiable comparison is provided for the results from each parameter analysis of GA and SA in terms of performance of solving TSP.

Keywords—Traveling Salesman Problem; Genetic Algorithm; Simulated Annealing

1 Introduction

In recent years, the importance of Artificial intelligence (AI) keeps increasing, and there is a new application of AI arises every day. AI based application can be found in the agriculture, education, finance, government, military, and entertainment industries, and many examples of optimization techniques used in AI can be found in the literature [1-3]. Considering all the advances in AI and increasing applications in our daily lives, the objective of this study is to explore a highly real-world applicable AI problem, the Traveling Salesman Problem (TSP), and provide a comprehensive look at adaptation of genetic algorithm (GA) and simulated annealing (SA) for TSP solution. As popular algorithms, the GA and SA are used in wide range of application areas, such as Bayesian inference links to particle methods in Bayesian statistics and hidden Markov chain models, electronic circuit design, learning fuzzy rule base, to train neural networks, and more.

There are many methodologies that have been introduced to tackle the TSP, and algorithms for solving the TSP can be divided into two classes; exact algorithms and heuristic algorithms [4]. The exact algorithms are guaranteed to find the optimal solution, in which number of iterations needed increases exponentially. Our study, however, focuses on the second of the two classes, the heuristic algorithms. There are several studies in the literature on heuristic algorithm-based TSP solution, some of them worth mentioning here. Chen presented a list of some of the methods that have been used on the TSP [5]. The approach presented contains most of the heuristic algorithms used to test TSP, including greedy algorithm, minimum-cost spanning tree, and local search. Additionally, modern optimization methods were also included such as SA, GA, ant algorithm, particle swarm optimization, tabu search algorithm, Hopefield neural networks [5].

One of these approaches is the Ant Colony Optimization (ACO), and it is studied for TSP by Fejzagic and Oputic [6]. In their approach, each node of the ACO graph represents a city, and each arc represents a connection between two cities. In each step of solution construction, an ant arriving in node (city) *i* chooses the next city to move to as a function of the pheromone values and function of the heuristic values on the arcs connecting city *i* to the cities the ant has not visited yet until all cities have been visited [6]. The most important drawback of ACO for TSP approach is that it can be easily trapped into local optima [7]. One solution to that drawback was presented where it forces ants to expand their search space considering only the distances of all unvisited paths connected to the current city [7]. Takahashi took a different approach and combined ACO and SA algorithms, where elitist ants periodically increase or decrease the quantity of pheromone with the number of tours increment [8].

Another approach presented in the literature for solution of TSP is Particle Swarm Optimization (PSO). PSO is described as the algorithm for finding optimal regions of complex search spaces through the interaction of individuals in a population of particles [9]. Basically, PSO is a population-based optimization technique on metaphor of social behavior of flocks of birds and/or schools of fish [9]. Moreover, in PSO, at every step, each particle changes position based on its velocity that depends on its previous best position and the best one among all the particles in the population [10].

Although all these studies are presented in the literature, little focus has been given on analysis of GA based solution to TSP. In this study, the goal is (i) to develop evolutionary algorithm (GA) based solution to TSP as well as SA based solution (ii) to perform comprehensive parameter analysis (iii) to present results comparing GA and SA based TSP solutions.

The rest of the paper is organized as follows: next section describes the basics of GA developed as a solution of TSP, followed by the section that describes SA based approach. The Simulation Results and Discussion section provides information about simulation design, results, and discussions. In the final section, conclusions are presented.

2 Genetic Algorithm Based Approach for TSP Solution

Genetic algorithm, which simulates Darwinian genetic selection and biological evolution, is widely studied in the literature [5, 11–16]. GA is described as adaptive search technique based on the principles and mechanisms of natural selection and the survival of the fittest concept of natural evolution [11]. GA simply operates by an iterative procedure on a fixed size population or pool of candidate solutions [11]. The candidate solutions represent an encoding of the problem in a form that is analogous to the chromosomes of biological systems. GA includes chromosomes that are typically represented as a string of bits. Each chromosome in the GA represents a possible solution for a given objective function. Associated with each chromosome is a fitness value, which is found by evaluating the chromosome with the objective function [11]. It is the fitness of a chromosome, which determines its ability to survive and produce offspring. Selecting organisms based on fitness value is a major factor in GA, and the success of algorithms lies in the propagation of the fittest scheme [17].

GA relies on three genetic operators: selection, crossover, and mutation [11, 22]. In our study, we use the distance of the route as the evaluation criteria, and the definition of the fitness function is given as the inverse of the total distance of the route, i.e. the shorter the route, the better the fitness. The fitness can be calculated by an equation similar to the following equation [5].

$$F(t) = \max - \left(\sum_{i=0}^{n-2} D(i,j) + D(0, n-1)\right) \tag{1}$$

where max is the longest path length in the present generation. In this approach, it is first needed to calculate the total distance of all the cities in the sequence, then computes the maximum total distance of the present generation [5]. Population diversity and selection are other important parts of the GA method. Initial population should be scattered randomly in the search space for achieving the global optimum [16]. Diversity of the routes can be achieved by random generation, however that can significantly affect the convergence speed. There are four main types of selection, Roulette Wheel Selection, Rank Selection, Steady State Selection, and Tournament Selection [15]. These are not competing methods but can be used interchangeably as well as in unison with each other. In our study, we adapted the Roulette Wheel Selection [18].

The principle of roulette selection follows a linear search through a roulette wheel with the slots in the wheel weighted in proportion to the individual's fitness values [19]. The probability of an individual being selected as a parent for crossover is given by the following equation [20].

$$p(i) = \frac{f(i)}{\sum_{j=1}^{n} f(j)} \tag{2}$$

In addition to use of the roulette wheel selection method, we also implement a way to keep the best route found during a particular selection sequence, Elitism [8]. Elitist strategy performs two important steps; carrying the individual with the best fitness result to the next generation, and removing the individual with the worst result from the population [16]. Furthermore, the elitist strategy can also avoid losing individual with best fitness result by mutation or crossover, and can help keeping that individual in the population [21].

Another operator in GA is crossover, which is the recombination of two individuals to create new ones that might have a better performance or better route lengths [15]. However, erroneous implementations of crossover can be problematic in a GA when applied on TSP. If crossover operation is performed repeatedly, it leads to loss of diversity of the population and can speed up the false convergence, which most likely result in local optimal and not global optimal [8]. There are many common crossover operators introduced in the literature, such as Edge Assembly Crossover (EAX) [8], sub-tour exchange crossover (CSE-X) [11], modified ordered crossover (MOX) [22], partially matched crossover (PMX) [23]. But none of them consider the relation between edges in TSP. So they may not accelerate the speed of the algorithm. All these variations help us to see that the crossover method should be chosen carefully based on the application.

Finally, last part of the GA framework is mutation. It has important aspect in improving local search capability and maintaining population variability, while preventing premature solutions [16]. Mutation operator induces changes in a small number of chromosomes units [15]. Its purpose is to maintain the population diverse enough during the optimization process. Again, just as with the other GA operators, there are many different mutation methods. The most common ones are shift mutation, insertion mutation, inverted mutation [5].

3 Simulated Annealing Based Approach for TSP Solution

In broader context, annealing is a process in material science that involves heating and cooling metals. It is the process of heating a material to a particular temperature, maintaining it at that temperature for a certain amount of time, and then cooling. Annealing is used to reduce hardness, increase ductility, and help eliminating internal stresses of metal. Simulated annealing (SA) is a probabilistic technique that uses the heating and cooling concept to help with approximating global optimum of a given function [24, 25]. SA is ideal for solving the traveling salesman problem (TSP).

SA uses randomness in its dataset in order to achieve satisfactory performance. SA has an initial system temperature, which is set to high, to allow the algorithm to traverse several different possible solutions. The algorithm accepts good and bad solution sets, at higher temperatures; as the temperature cools, the algorithm accepts less and less solutions with the designed aim that an optimal solution is found. SA fundamentally determines to what degree a solution is worse than another solution as it relates to the temperature of the system.

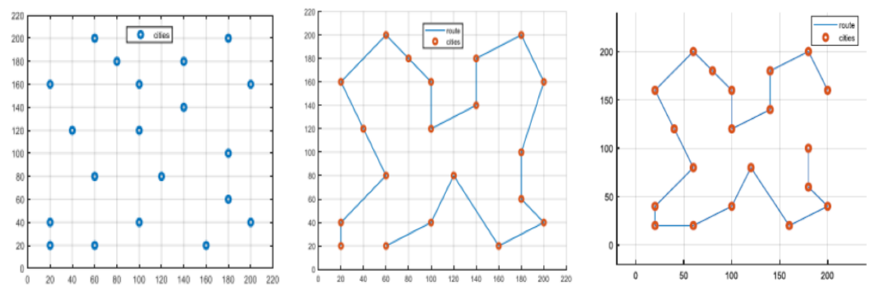
SA is applied to TSP in such a way that highlights its features. Routes in TSP are slightly, but randomly rearranged. At first, when the temperature is high, any solution is an acceptable solution; however, as the temperature of the system begins to cool, only the best solutions will remain. Key terms of SA can be listed as follows: temperature, cooling rate, and acceptance probability. The temperature is utilized to set and measure the overall system temperature. The cooling rate is the rate at which the temperature of the system is decreased. The acceptance probability is the probability in which a new solution is accepted over the previous solution. The function used to calculate the acceptance probability is given as [25].

$$P(a) = C \cdot \exp\left(\frac{-E_{conf}}{T}\right) \tag{3}$$

where T is the temperature, C is constant of normalization, and E_{conf} is the configuration energy.

4 Simulation Results and Discussion

The task of solving TSP is to find the shortest possible route between groups of cities. The salesman is to traverse all possible city locations only once in the shortest number of steps possible depending on the amount of cites visited, which could take a significantly long amount of time to find an optimal solution. We implemented two algorithms in this study, GA and SA, and their solutions were tested in simulation environment. It would be prudent to define an aspect of this test that is shared among GA and SA for simulation experiments. Since these algorithms have been adapted to be utilized by TSP, some terms that have been adapted have more to do with TSP than the actual algorithm being leveraged upon. Route, is one such term. Route is not a standard word in the GA or SA vocabulary, but it is equivalent to chromosomes in GA and it is the solution being manipulated in SA. The routes are calculated randomly in both GA and SA.



a) Group of cities b) GA solution c) SA solution

Fig. 1. Simulation results

When comparing the solutions of GA and SA search algorithms, there is very little difference as it concerns the particulars of the best route. When given a set of cities with a known solution, route the SA and GA solutions are identical, except for starting cities (Fig. 1).

Both the GA and the SA require tuning the input parameters of the environment. However, the constraints for both the SA and the GA are different. The list of the required parameters for GA is as follows: number of randomly generated routes, number of elites to keep, the mutation rates, and the number of generations to execute the experiment. The SA required a starting temperature and a cooling rate for the system. The system executed until the temperature was less than 1 degree.

Trying to find a one-to-one correlation of these parameters provided is a substantial challenge. Given that the number of cities was the same for each algorithm, it was known that changing any one parameter in either processes would cause a different result. We started our analysis with GA focusing on number of randomly generated routes. The simulation parameters were set as 20 cities, 100 elites, 0.01 mutation rate, and 2000 generations. The following generated route numbers were implemented; 100, 200, 500, 1000, 1500. Figure 2 depicts the resulting plot showing routes generated vs distance. As the number of generated routes increases, the total distance between cities decreases. The best results were obtained with 1500 generated routes.

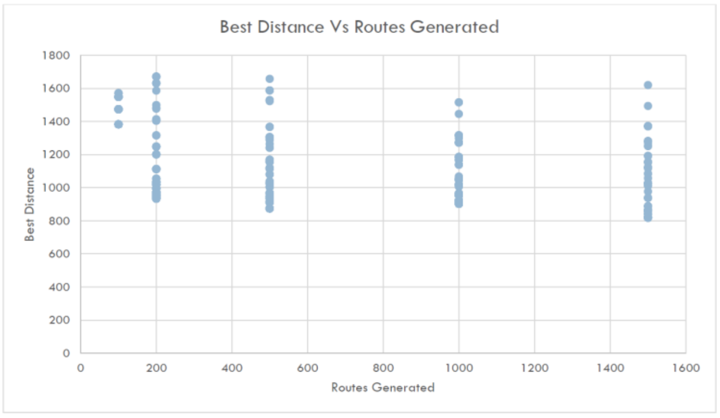


Fig. 2. Route Best Distances Compared to Number of Routes Generated

The next parameter we analyzed is number of elites to keep. The simulation parameters were set as 20 cities, 200 generated routes, 0.01 mutation rate, and 2000 generations. The parameters of elites used were selected as; 1, 10, 25, 75, 100, 130, 160, 185 and 199. The result is depicted in Fig. 3. The elite number of 10 provided best results in terms of total distance between cities.

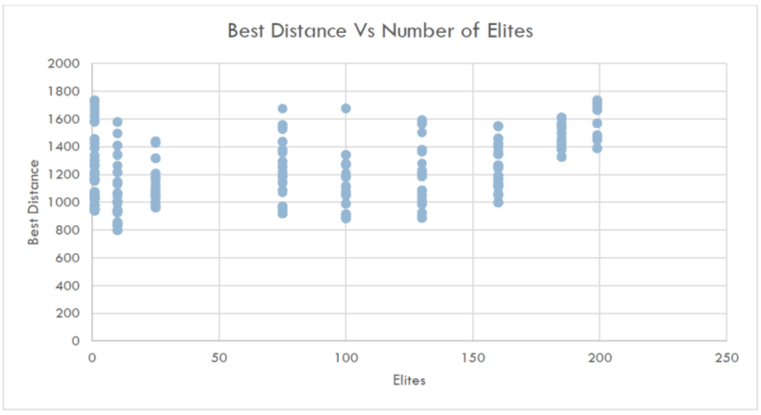


Fig. 3. Route Best Distances Compared to Size of Elite Group

We also analyzed the mutation rate in GA approach. The simulation parameters were set as 20 cities, 200 generated routes, 100 elites to keep, and 2000 generations. The values of 0.0001, 0.001, 0.01, 0.06, 0.09, 0.25, 0.55, 0.85, and 0.99 were used for mutation rates. Figure 4 depicts the results. The best results were obtained with minimum mutation rate values.

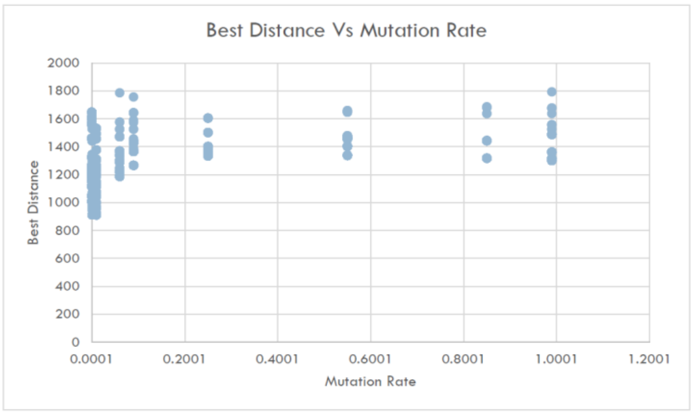


Fig. 4. Route Best Distances Compared to Mutation Rate

Finally, number of generations is analyzed. The simulation parameters were set as 20 cities, 200 generated routes, 2000 mutation rate, and 100 elites to keep. The number of generations used were selected as; 1, 10, 100, 1000, and 10000. The resulting plot is depicted in Fig. 5. The highest generation number led to best distance results.

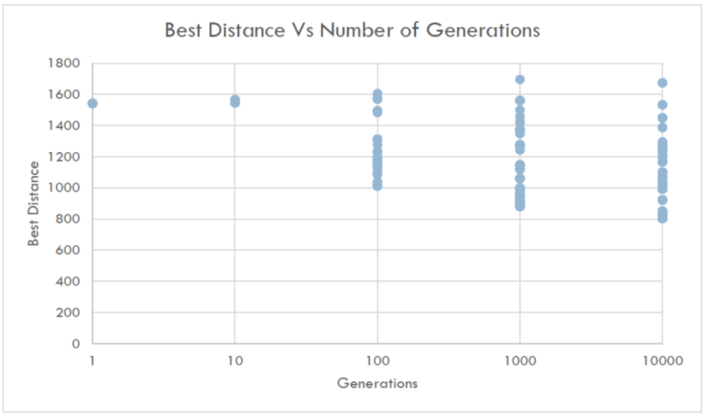


Fig. 5. Route Best Distances Compared to Number of Route Generations

In the analysis with SA, there were two parameters used; temperature and cooling rate. The results of temperature analysis are depicted in Fig. 6, where cooling rate was set to 0.00001, and values of 1, 10, 100, 1000, 10000, and 100000 were used for temperature. The best distance is not changed after temperature equals to 100. Figure 7 shows the results of cooling rate analysis, where temperature was set to 10000, and values of 0.99, 0.1, 0.01, 0.001, 0.0001, 0.00001, and 0.000001 were used for cooling rate.

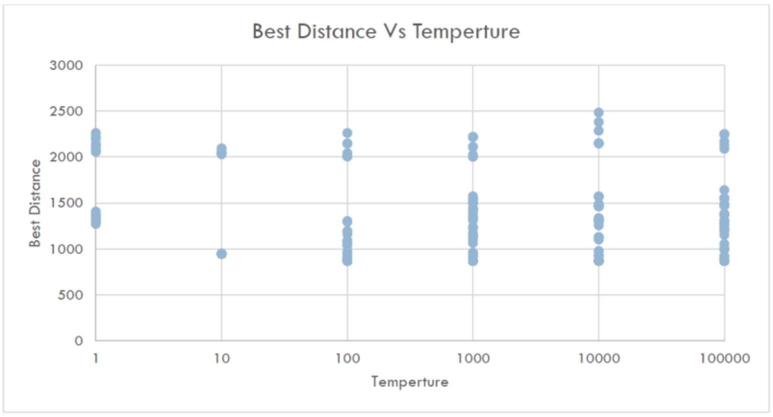


Fig. 6. Route Best Distances Compared to Temperature

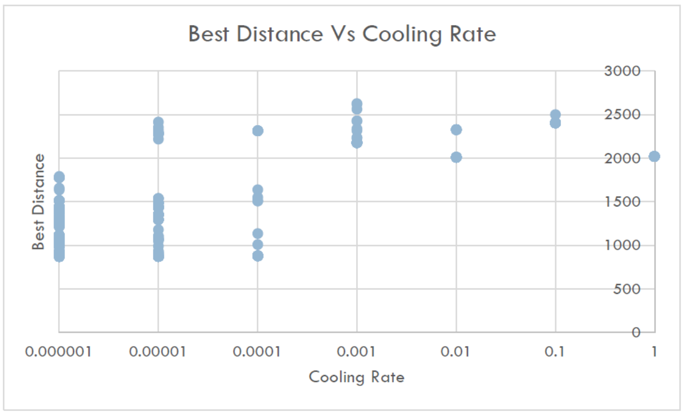


Fig. 7. Route Best Distances Compared to Cooling Rate

According to results, increasing the number of elites did not allow for much route diversification. Meanwhile increasing the mutation rate in the GA resulted in too much diversification in the selection pool. Therefore, there must be a balance as it pertains to those two parameters of the GA. The number of generations affected the total outcome only when the number of cities and/or the number of routes were increase. It is known that the increase in cities will increase the number of possible routes and thus the number of solutions; so, by increasing the number of generations, one could, increase the probability of finding a solution when presented with higher city and route totals. However, the more generations, the longer it took complete the algorithm. The significant part was that an optimal solution could be found early on and was not influenced by how long the algorithm ran (Fig. 8).

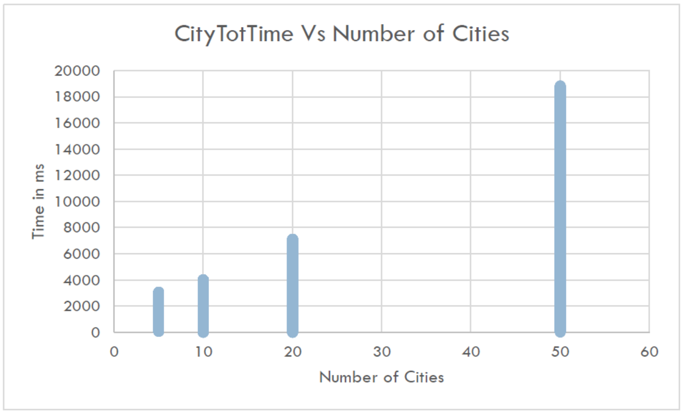


Fig. 8. Time vs Number of cities generated - GA

The SA only had two parameters, but both had a profound effect on how long it took to run the algorithm. This, however, did not affect its ability to find a solution early in the process. It was determined however, higher initial temperatures and lower cooling rates yielded better results. Thus, computingting time may indeed be a factor in finding an appropriate solution (Fig. 9).

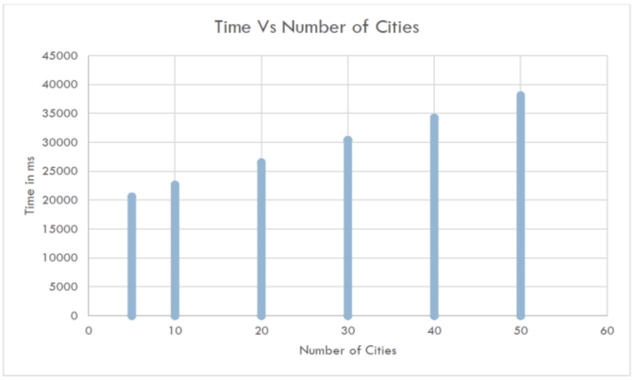


Fig. 9. Time vs Number of cities generated - SA

It was challenging to one-to-one compare the two algorithms, since they had different input parameters. The only parameters that have a one-to-one correlation were the number of cities generated, number of iterations, and time to find an optimal solution. There were still challenges even with these similarities. The time it took the SA and GA did not correlate to a better solution. Consequently, it was concluded to compare the algorithms by how many iterations it took to find an optimal solution. In Figs. 10 and 11, it will be plain to see that the GA ran considerably less iterations than the SA. This is mainly because an iteration in the GA is equivalent to a generation.

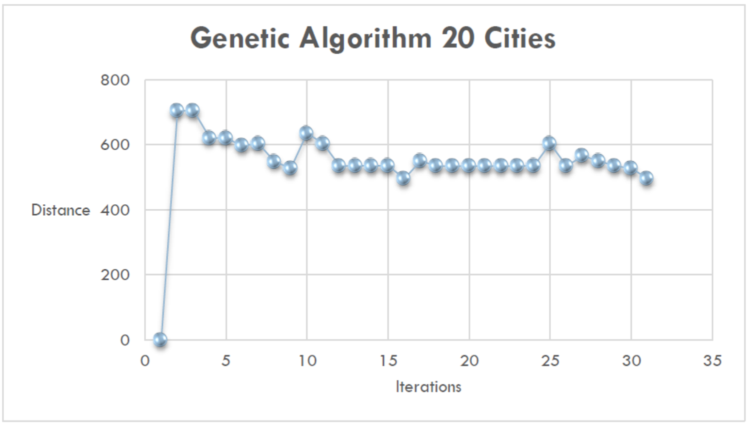


Fig. 10. GA Iteration Number

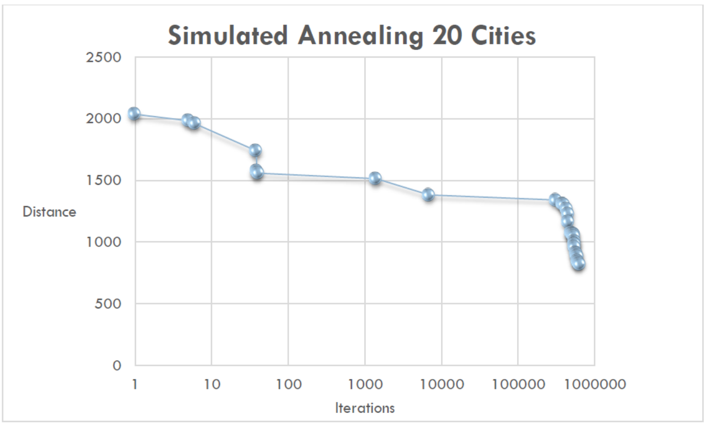


Fig. 11. SA Iteration Number

5 Conclusion

This paper presents a comprehensive analysis of GA and SA based solution methods for traveling salesman problem (TSP). Analysis of the GA based TSP solution provides four main classes: number of routes, number of elites to keep, mutation rate, and number of generations to execute the simulation, on the other hand, analysis of the SA based TSP provides for two main classes: temperature and cooling rate. According to the simulation results, it was determined that although the SA executed with faster completion times, it took more iterations to find a solution. Moreover, the found solutions were not always as accurate as the GA. The GA took longer time to run but found a solution with less iterations.

6 References

[1] Siame, A., & Kunda, D. (2018). University Course Timetabling using Bayesian based Optimization Algorithm. International Journal of Recent Contributions from Engineering, Science & IT (iJES), 6(2), 14-36. https://doi.org/10.3991/ijes.v6i2.8990
[2] Rashid, T., & Jabar, A. (2018). A modified particle swarm optimization with neural network via Euclidean distance. International Journal of Recent Contributions from Engineering, Science & IT (iJES), 6(1), 4-18. https://doi.org/10.3991/ijes.v6i1.8080
[3] Alharbi, W. N. H., & Gomm, B. (2017). Genetic Algorithm Optimisation of PID Controllers for a Multivariable Process. International Journal of Recent Contributions from Engineering, Science & IT (iJES), 5(1), 77-96. https://doi.org/10.3991/ijes.v5i1.6692
[4] Tan, Y.Y.; Tan, L.Z.; Yun, G.X.; Zheng, W. Bilevel Genetic Algorithm with Clustering for Large Scale Traveling Salesman Problems. 2016 International Conference on Information System and Artificial Intelligence (ISAI). IEEE, 2016, pp. 365–369. https://doi.org/10.1109/isai.2016.0084

[5] Chen, P. An improved genetic algorithm for solving the Traveling Salesman Problem. 2013 Ninth International Conference on Natural Computation (ICNC). IEEE, 2013, pp. 397–401. https://doi.org/10.1109/icnc.2013.6818008

[6] Fejzagic, E.; Oputic, A. Performance comparison of sequential and parallel execution of the Ant Colony Optimization algorithm for solving the traveling salesman problem. 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2013, pp. 1301–1305. https://doi.org/10.23919/mipro.2017.7973596

[7] Satukitchai, T.; Jearanaitanakij, K. An early exploratory method to avoid local minima in Ant Colony System. 2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON). IEEE, 2015, pp. 1–5. https://doi.org/10.1109/ecticon.2015.7206969

[8] Takahashi, R. A hybrid method of genetic algorithms and ant colony optimization to solve the traveling salesman problem. 2009 International Conference on Machine Learning and Applications. IEEE, 2009, pp. 81–88. https://doi.org/10.1109/icmla.2009.31

[9] Shigehiro, Y.; Katsura, T.; Masuda, T. An application of particle swarm optimization to traveling salesman problem. Proceedings of SICE Annual Conference 2010. IEEE, 2010, pp. 1629–1632.

[10] Akhand, M.; Akter, S.; Rahman, S.S.; Rahman, M.H. Particle swarm optimization with partial search to solve traveling salesman problem. 2012 International Conference on Computer and Communication Engineering (ICCCE). IEEE, 2012, pp. 118–121. https://doi.org/10.1109/iccce.2012.6271164

[11] Katayama, K.; Sakamoto, H.; Narihisa, H. The efficiency of hybrid mutation genetic algorithm for the travelling salesman problem. Mathematical and Computer Modelling 2000, 31, 197–203. https://doi.org/10.1016/s0895-7177(00)00088-1

[12] Yugay, O.; Kim, I.; Kim, B.; Ko, F.I. Hybrid genetic algorithm for solving traveling salesman problem with sorted population. 2008 Third International Conference on Convergence and Hybrid Information Technology. IEEE, 2008, Vol. 2, pp. 1024–1028. https://doi.org/10.1109/iccit.2008.373

[13] Liu, J.; Li,W. Greedy permuting method for genetic algorithm on traveling salesman problem. 2018 8th International Conference on Electronics Information and Emergency Communication (ICEIEC). IEEE, 2018, pp. 47–51. https://doi.org/10.1109/iceiec.2018.8473531

[14] Xin, J.; Zhong, J.; Yang, F.; Cui, Y.; Sheng, J. An improved genetic algorithm for path-planning of unmanned surface vehicle. Sensors 2019, 19, 2640. https://doi.org/10.3390/s19112640

[15] Wang, Z.; Duan, H.; Zhang, X. An improved greedy genetic algorithm for solving travelling salesman problem. 2009 Fifth International Conference on Natural Computation. IEEE, 2009, Vol. 5, pp. 374–378. https://doi.org/10.1109/icnc.2009.504

[16] Wei Zhou.; Yuanzong Li. An improved genetic algorithm for multiple traveling salesman problem. 2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010), 2010, Vol. 1, pp. 493–495. https://doi.org/10.1109/car.2010.5456787

[17] Frahadnia, F. A New Method Based on Genetic Algorithms for Solving Traveling Salesman Problem. 2009 International Conference on Computational Intelligence, Modelling and Simulation. IEEE, 2009, pp. 11–16. https://doi.org/10.1109/cssim.2009.45

[18] Chen, S.M.; Chien, C.Y. A new method for solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization

techniques. 2010 International Conference on Machine Learning and Cybernetics. IEEE, 2010, Vol. 5, pp. 2477–2482. https://doi.org/10.1109/icmlc.2010.5580809
[19] Saini, N. Review of selection methods in genetic algorithms. International Journal of Engineering and Computer Science 2017, 6, 22261–22263.
[20] Jebari, K.; Madiafi, M. Selection methods for genetic algorithms. International Journal of Emerging Sciences 2013, 3, 333–344.
[21] Sharma, P.; Wadhwa, A. Analysis of selection schemes for solving an optimization problem in genetic algorithm. International Journal of Computer Applications 2014, 93.
[22] Li, J.; Chen, P.; Liu, Z. Solving Traveling Salesman Problems by Genetic Differential Evolution with Local Search. 2008 Workshop on Power Electronics and Intelligent Transportation System. IEEE, 2008, pp. 454–457. https://doi.org/10.1109/peits.2008.48
[23] Vahdati, G.; Ghouchani, S.Y.; Yaghoobi, M. A hybrid search algorithm with Hopfield neural network and Genetic algorithm for solving traveling salesman problem. 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE). IEEE, 2010, Vol. 1, pp. 435–439. https://doi.org/10.1109/iccae.2010.5451917
[24] Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. science 1983, 220, 671–680. https://doi.org/10.1126/science.220.4598.671
[25] Cerny, V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of optimization theory and applications 1985, 45, 41–51. https://doi.org/10.1007/bf00940812

7 Authors

Darius Bethel is a MS student in Department of Electrical and Computer Engineering at University of West Florida, 11000 University Parkway, Pensacola, FL 32514.

Hakki Erhan Sevil is currently an Assistant Professor in the Department of Intelligent Systems & Robotics at University of West Florida (UWF), and he is the PI of the Sevil Research Group at UWF. Department of Intelligent Systems & Robotics, University of West Florida, 11000 University Parkway, Pensacola, FL 32514, http://www.bit.ly/sevilresearch

Article submitted 2021-02-25. Resubmitted 2021-04-08. Final acceptance 2021-04-09. Final version published as submitted by the authors.