Simulated Annealing for Travelling Salesman Problem Using Ben-Ameur Formula to Find Initial Temperature

Jared Ratto
jr860@exeter.ac.uk
University of Exeter
Exeter, UK

ABSTRACT

A simulated annealing algorithm is designed to solve the Traveling Salesman Problem (TSP). The algorithm parameters of initial temperature, cooling schedule, and iterations are studied and tested. In order to tune the initial temperature parameter, a formula created by Walid Ben-Ameur is implemented. The algorithm is tested on five problem sets and its performance is compared to random search and stochastic hill-climber algorithms.

KEYWORDS

optimization, local search, stochastic modelling, permutations

ACM Reference Format:
Jared Ratto. 2023. Simulated Annealing for Travelling Salesman Problem Using Ben-Ameur Formula to Find Initial Temperature. In *Woodstock '23: ACM Symposium on Neural Gaze Detection, May 03–05, 2023, Woodstock, NY*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The Travelling Salesman Problem (TSP) is one of the oldest and most studied problems in discrete optimization. It has been studied for applications in logistics, transportation, computer hardware design, and many others [9]. The problem can be described as, given a set of cities and the associated distances or travel costs among them, find the shortest route visiting each of the cities only once. Although the problem can be easily described, it is remarkably difficult to solve, especially for a large number of cities. As the number of cities grows, the number of possible routes grows by a factorial expansion. It is an NP-hard problem, meaning that no polynomial time algorithm has been created that can guarantee finding the global optimum solution [19].

Many metaheuristics have been used in order to approximate the global optimum. One of these, simulated annealing, is a local search algorithm that draws its name from a metallurgy process that initially heats a piece of metal allowing its form to change easily at first and then gradually cools it so that it hardens to a desired shape. In practice, it accepts a new solution if it is better than its current solution, but it may accept a worse solution according to a probability. This behavior of the algorithm is used to address the challenge of local optima in optimization problems. Local optimum solutions are those that are optimal in their segment of the search space but their value is far from that of the global optimum. By occasionally accepting worse solutions, the simulated annealing algorithm may escape local optima and eventually find an approximation of the global optimal solution. The probability of accepting a worse solution follows a cooling temperature analogy where the probability begins high but gradually decreases [6, 14]. This also leads the algorithm to be more exploratory in earlier iterations and to be more exploitative in later ones.

Simulated Annealing can be a powerful algorithm for optimization, but its success is highly dependent on its parameters being properly tuned. Furthermore, tuning these parameters can be a challenging and tedious task requiring much experimentation [19]. One of the most challenging of these parameters to set is the value of the initial of the initial temperature. Many works have studied systematic methods of doing this, and this paper has adopted the approach authored by Walid Ben-Ameur. Likewise, the choice of cooling schedule can greatly affect the algorithm's performance and this paper compares multiple, differing schedules.

In this paper, we will construct a simulated annealing algorithm that tunes these two central parameters while solving the TSP. The performance of the constructed algorithm will be compared to two simpler ones, namely stochastic hill-climber and random search. The rest of this paper is organized as follows: section 2 explores relevant literature on both simulated annealing and the TSP, section 3 describes the algorithm in detail and problems used, section 4 describes the experiment method and results, and finally section 5 is the conclusion.

2 LITERATURE REVIEW

2.1 Travelling Salesman Problem

The first usage of the term *Travelling Salesman Problem* in a published paper was by Julia Robinson in 1949 with her work with the Hilbert Tenths Problem [13]. In [8] Little et al. published the Branch and Bound method which remains a popular method today [7]. Nicos Christofides discovered an algorithm that approximates a solution that is guaranteed to be within 50percent of the global optimal solution in polynomial time [4]. Nathan Klein, Anna Karlin, and Shayan Oveis recently discovered an algorithm similar to that of [4] that is faster by 10^{-36} of a percent [11].

2.2 Simulated Annealing

The simulated annealing algorithm was proposed by Scott Kirkpatrick et al. explaining the analogy drawn from annealing in solids [6]. Vladimir Černý also independently proposed the simulated annealing method near the time of [6] and used the algorithm on

Woodstock '23, May 03–05, 2023, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/23/05…$15.00
https://doi.org/10.1145/1122445.1122456

the TSP [3]. Wang et al. tested a 2-stage algorithm on 23 sets of TSP problems and revealed it is difficult for the algorithm to solve benchmark problems with more than 1000 cities due to computation time [16, 19]. Geng et al. proposed an adaptive simulated annealing algorithm with a greedy criterion (ASA-GS) that scaled effectively on problem sets with more than 1000 cities [5]. (ASA-GS) was later outperformed in terms of computation time and accuracy by a multiagent simulated annealing algorithm with instance-based sampling (MSA-IBS) [15].

Many works have been published studying how to efficiently tune parameters for simulated annealing. Walid Ben-Ameur proposed an algorithm finding a near optimal value for the initial temperature [2]. The algorithm is based off of a method described in [Kirkpatrick] for finding a temperature value leading to a desired acceptance probability for your candidate solution found in the first iteration of the algorithm. Dennis Weylend explored the influence of different values of initial temperature and cooling schedule affected the optimality of the algorithm from a theoretical perspective[17]. Zhan et al. used a list-based method to efficiently tune the cooling schedule parameter and found robust performance when testing on TSP benchmark problems[19].

3 SEARCH ALGORITHM

3.1 Problem Formulation

Each of the benchmark problems in my problem sets are lists of cities, or nodes, that are each encoded by 2-dimensional euclidean coordinates. A solution is encoded as a permutation representing a route connecting each node once and then returning to the starting node. Each solution is evaluated by the cumulative distance traveled between each city in the route, and distances between cities are calculated using the euclidean distance. The distances between cities are symmetric, meaning that the distance from node A to node B is the same as the distance from node B to node A. The objective for each problem is defined as finding the route with the shortest cumulative distance.

3.2 Pseudocode

The following is a pseudocode of this paper's simulated annealing algorithm.

Algorithm 1 Simulated Annealing

Input: city list x, max iterations N, initial temperature t_0, cooling schedule formula c

Output: optimized route of x

$r_0 \leftarrow x$
$t \leftarrow t_0$
$rand \leftarrow$ randomly generated number between 0 and 1
for $i \leftarrow 1$ to N **do**
    $r_1 \leftarrow$ copy of r_0 with randomly swapped indices
    if $d_1 < d_0$ **then**
        $r_0 = r_1$
    else if $rand \leq e^{(-(d_1-d_0)/t)}$ **then**
        $r_0 = r_1$
    end if
    $t = t - c$
end for
return r_0

Where city list x is the list of cities in the TSP benchmark problem, r_0 and r_1 respectively represent the current and candidate routes, d_0 and d_1 represent the cumulative distances of the two routes, and c is the cooling factor applied to the current temperature value.

The candidate route r_1 is created by randomly swapping two cities within r_0. If r_1 has a shorter distance, it will always replace the r_0; however, if it has a greater distance, the metropolis acceptance criterion [10] is applied to decide whether r_1 will replace r_0. In this criterion, both the temperature and the difference between two distances play a role in the probability of the worse route is accepted.

The cooling factor will vary depending on the cooling schedule parameter chosen by the user. In this paper, three schedules are considered [18] :

Linear Additive:

$$t_i = t_N + (t_0 - t_N)(\frac{N-i}{N})$$

Quadratic Additive:

$$t_i = t_N + (t_0 - t_N)(\frac{N-i}{N})^2$$

Exponential Multiplicative:

$$t_i = t_0 \cdot a^i$$

Where t represents the temperature value, i represents the current iteration, 0 represents the initial iteration, and N represents the final iteration of the algorithm. a is a hyperparameter set to 0.90.

4 EXPERIMENTATION

4.1 Problem Instances

Five benchmark TSP problems from TSPLIB [1] of varying scale are used during the experiments. The number of cities in each are 52, 70, 101, 130, and 280. Each algorithm is evaluated by their performance on each of these problems. Simply put, the algorithm that returns the route with the significantly lowest distance is judged to be the

best algorithm. The returned distances are averaged across multiple runs with random seeding to account for stochasticity.

4.2 Tuning Initial Temperature

To tune the parameter of initial temperature, the algorithm proposed by Walid Ben-Ameur is used. The following is the pseudocode for this algorithm:

Algorithm 2 Find Initial Temperature

Input: acceptance probability X_0, sample size $\|S\|$, p, ϵ, t_0
Output: optimal initial temperature t_i

1: $t_i \leftarrow t_0$
2: **for** $j \leftarrow 1$ to $\|S\|$ **do**
3: $\quad r_0, r_1 \leftarrow$ randomly generate routes where $d_1 > d_0$
4: $\quad S \leftarrow$ appended by r_0, r_1
5: **end for**
6: $\hat{X}(t_i) = \frac{\Sigma_{s \in S} e^{-\frac{r_{1s}}{t_i}}}{\Sigma_{s \in S} e^{-\frac{r_{0s}}{t_i}}}$
7: **while** $\hat{X}(t_i) - X_0 > \epsilon$ **do**
8: $\quad t_i \leftarrow t_i \left(\frac{\ln(\hat{X}(t_i))}{\ln(X_0)}\right)^{\left(\frac{1}{p}\right)}$
9: $\quad \hat{X}(t_i) = \frac{\Sigma_{s \in S} e^{\left(-\frac{r_{1s}}{t_i}\right)}}{\Sigma_{s \in S} e^{\left(-\frac{r_{0s}}{t_i}\right)}}$
10: **end while**
11: **return** t_i

The acceptance probability X_0 is the probability of accepting r_1 in the first iteration if it is a worse solution than r_1. Walid Ben-Ameur suggests setting this to 0.8. S is a set of pairs of solutions, r_0, r_1, where $d_1 > d_0$. The difference between these is accounted for in the calculation of $\hat{X}(t_i$. In principle, the differences between energy states, in our case distances, of multiple solutions is used to approximate the acceptable probability $\hat{X}(t_i$ given the current temperature value. In steps 7-9, the temperature t_i is recursively adjusted until the value of $\hat{X}(t_i$ is sufficiently close to X_0. p is a hyperparameter that affects how much the temperature is changed during each iteration [2].

In [2] Walid Ben-Ameur does not specify what values of $\|S\|$ should be used. Through a small experiment on the smallest problem set of 52 cities, a value of 3000 seemed appropriate. This experiment tracked the standard deviation of the returned initial temperature value using Walid Ben-Ameur's algorithm across differing values of $\|S\|$ and is shown in Figure 1.

For the other four TSP problems, the value of $\|S\|$ was approximated using the ratio of the scales between the 52 city problem and each larger-scale problem.

4.3 Tuning Cooling Schedule

To select the most optimal cooling schedule for the algorithm, the impact on the algorithm's computed distance from each cooling schedule used was visualized and compared on the 52-city problem (Figures 2-4). To reduce the effects of stochasticity, the results of each are averaged over 10 seeded runs.

By comparing the Distance axes of the figures, it is clear that the linear additive cooling schedule leads to more inferior distances

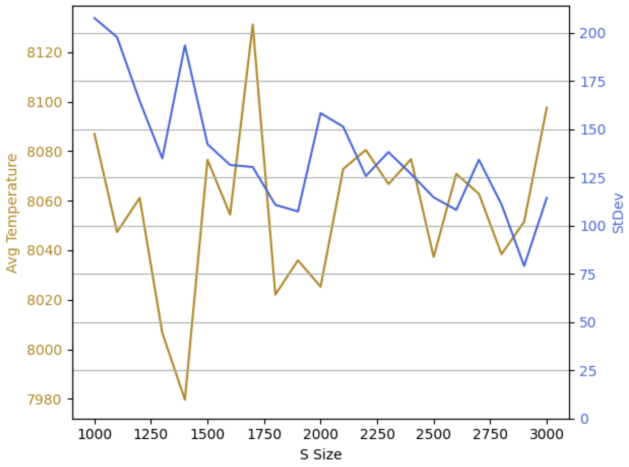

Figure 1: Standard deviations of average temperatures from differing values of $\|S\|$

than those returned by the quadratic additive and exponential multiplicative cooling schedules. Exponential multiplicative cooling proves to be slightly better than quadratic additive cooling in this TSP problem.

Both linear additive and quadratic additive cooling appear to converge later in the iterations because their higher temperature values yield less optimal distances in these earlier iterations. Because exponential multiplicative converges much faster than the other two, it was speculated that the other two schedules may reach an even more optimal solution than the faster schedule if given enough iterations. To explore this, 4000 was chosen as the maximum number of iterations in these visualizations. Even during large spans of iterations, exponential multiplicative appears to be the better schedule. It is unclear whether the two other schedules may eventually overtake the faster one, but if it does it will require a very large number of iterations that may raise computational cost concerns if implemented.

It is interesting to note that the temperature value in the exponential multiplicative cooling schedule becomes negligible after the 100^{th} iteration. Visually, it appears that few, if any, worse solutions are accepted during the algorithm's runs. This cooling schedule makes the simulated annealing algorithm behave more similarly to a stochastic hill-climber algorithm, that has no temperature parameter, than a traditional simulated annealing algorithm. Because of this, in this paper, we will use both the quadratic additive and exponential multiplicative cooling schedules when comparing simulated annealing to the other algorithms on the benchmark problems.

4.4 Compared Algorithms

Across the five TSP problems, both the stochastic hill-climber and random search algorithms are used to compare to the simulated annealing algorithms during the experiment.

The stochastic hill-climber algorithm is similar to simulated annealing but lacks a temperature and cooling schedule parameter. A neighboring solution, found by swapping a pair of cities, is compared to the current solution. If the neighboring solution has a

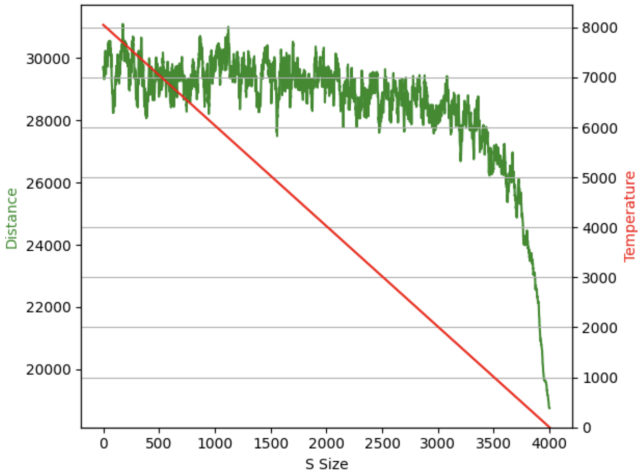

Figure 2: Linear Additive Cooling Schedule

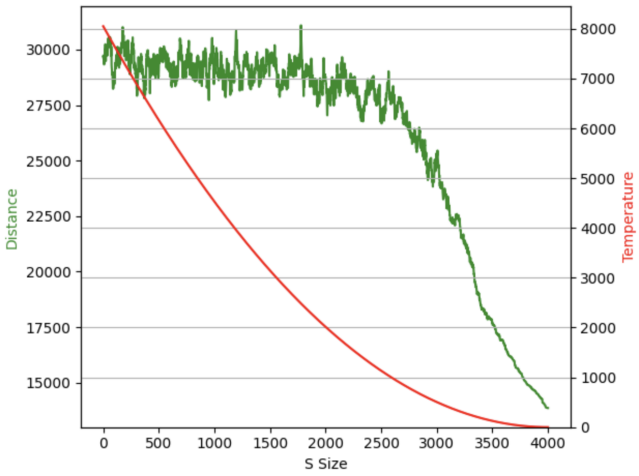

Figure 3: Quadratic Additive Cooling Schedule

better distance than the current one, it replaces the current solution. The search algorithm is stochastic because it randomly chooses a neighboring solution that is not necessarily the most optimal of all of the neighboring solutions [2].

The second algorithm used for comparison is simple random search. While more sophisticated methods of random search exist [12], the random search algorithm considered in this experiment is simply generating random solutions for a number of iterations and returning the best of these generated solutions. For each individual TSP problem in the experiment, these solutions are randomly generated routes connecting each city.

The results of each algorithm are compared after 4000 iterations. Table 1 displays the results of the algorithms on each problem set and Figures 5-9 display the results by each iteration visually.

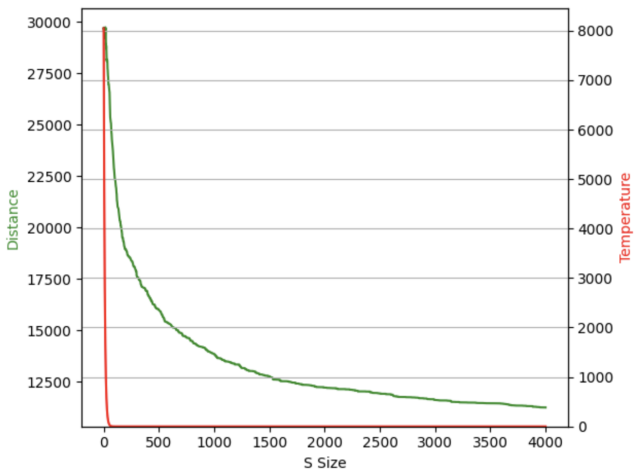

Figure 4: Exponential Multiplicative Cooling Schedule

Table 1: Comparison of Algorithm Performance

Scale	SA - Slow	SA - Fast	Stohastic HC	Random Search
52	14143	11267	11052	23329
70	1758	1207	1198	2932
101	1779	1241	1244	2902
130	25780	16821	17089	40165
280	21627	13999	13624	30501

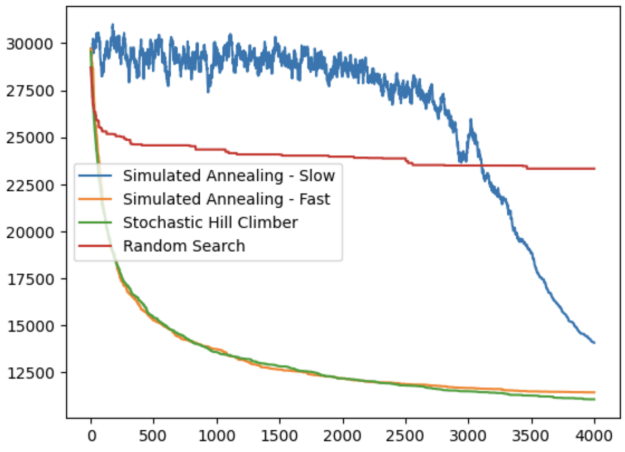

Figure 5: Exponential Multiplicative Cooling Schedule

4.5 Discussion

As observed in Table 1 and Figures 5-9, the stochastic hill-climber algorithm consistently either outperforms the other algorithms or has very similar performance to simulated annealing when its temperature parameter is negligible. The simulated annealing algorithm performs better in this experiment if it is more similar to the stochastic hill-climber. While both simulated annealing algorithms used temperature as a parameter, it becomes nearly negligible when used

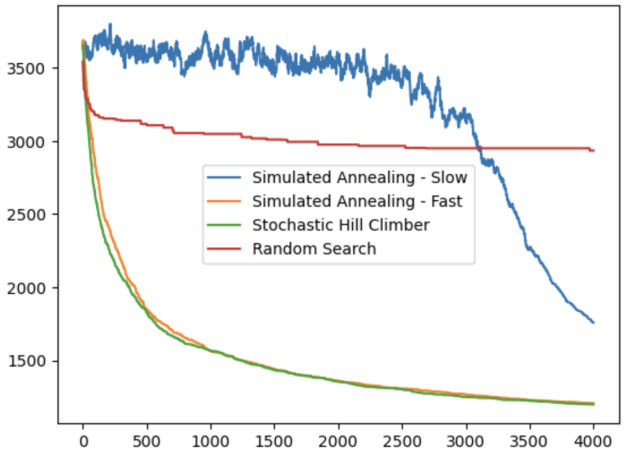

Figure 6: Exponential Multiplicative Cooling Schedule

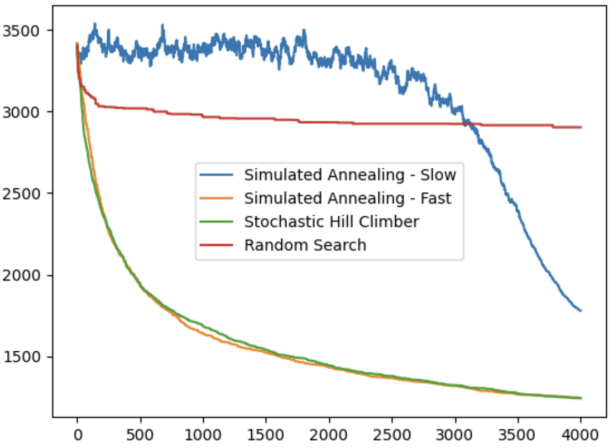

Figure 7: Exponential Multiplicative Cooling Schedule

in the exponential multiplicative cooling schedule, or fast schedule. Therefore, only the algorithm with the slower cooling schedule appears to show the traditional property of accepting worse solutions intended for escaping local minima.

Intuitively, one may reasonably have predicted that the stochastic hill-climber would have performed poorly because it would have gotten stuck in local minima. While it is possible that this is the case, the algorithm nonetheless performs remarkably well on each of the TSP problems, especially when compared against slow simulated annealing and random search. One possible explanation for this is that due to the high dimensionality of these problems, the operator of swapping a single pair of cities may have the potential of greatly changing a route's distance. While the stochastic hill-climber is limited only to moves that return a shorter distance, the vast search space and high dimensionality allow for local minima to be escaped. Further studies would be required to investigate this speculation.

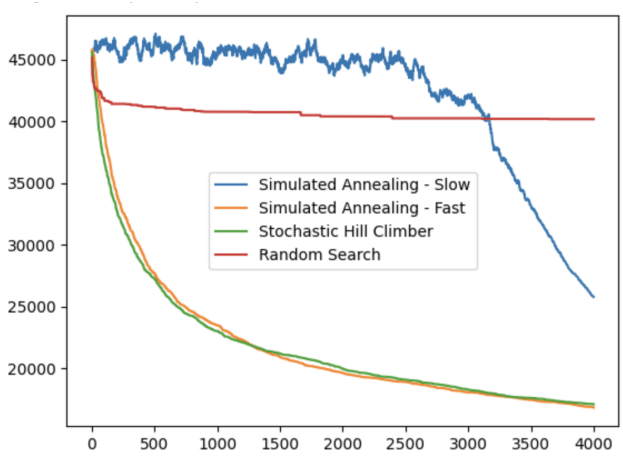

Figure 8: Exponential Multiplicative Cooling Schedule

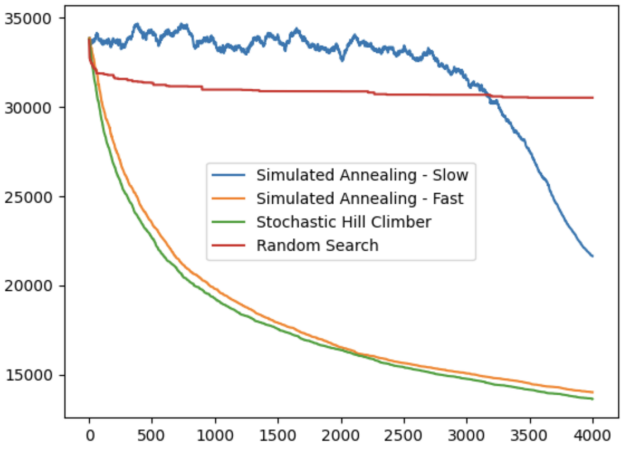

Figure 9: Exponential Multiplicative Cooling Schedule

The simulated annealing algorithm with the slower cooling schedule proves to not scale as effectively as the stochastic hill-climber in larger problem sets. By observing the figures, one can observe that the gap between the two algorithms widens as the problem's scale increases. This leads to the conclusion that the cooling schedule for the quadratic additive schedule is too slow. An interesting study would be to explore cooling schedules that are faster than quadratic additive but not as negligible as exponential multiplicative.

5 CONCLUSION

In this paper we have explored the challenging task of tuning the parameters of the initial temperature and the cooling schedule of simulated annealing. We have compared the results of two versions of simulated annealing with different cooling schedules against the stochastic hill-climber and random search algorithms. In each of the problem sets used in the experiment, the stochastic hill-climber either outperformed the other algorithms or were very similar to the performance of simulated annealing when its temperature variable

was negligible during the majority of iterations. Further work will explore cooling schedules that lead to better performance without having negligible temperatures.

REFERENCES

[1] [n. d.]. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

[2] Walid Ben-Ameur. 2004. Computing the Initial Temperature of Simulated Annealing. *Comput. Optim. Appl.* 29, 3 (2004), 369–385. https://doi.org/10.1023/B:COAP.0000044187.23143.bd

[3] Vladimír Černỳ. 1985. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications* 45 (1985), 41–51.

[4] Nicos Christofides. 2022. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. *Oper. Res. Forum* 3, 1 (2022). https://doi.org/10.1007/s43069-021-00101-z

[5] Xiutang Geng, Zhihua Chen, Wei Yang, Deqian Shi, and Kai Zhao. 2011. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. *Applied Soft Computing* 11, 4 (2011), 3680–3689.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by Simulated Annealing. *Science* 220, 4598 (1983), 671–680. https://doi.org/10.1126/science.220.4598.671 arXiv:https://www.science.org/doi/pdf/10.1126/science.220.4598.671

[7] Marc Kuo. 2023. Solving the traveling salesman problem for deliveries. https://blog.routific.com/blog/travelling-salesman-problem

[8] John D. C. Little, Katta G. Murty, Dura W. Sweeney, and Caroline Karel. 1963. An Algorithm for the Traveling Salesman Problem. *Operations Research* 11, 6 (1963), 972–989. https://doi.org/10.1287/opre.11.6.972 arXiv:https://doi.org/10.1287/opre.11.6.972

[9] Rajesh Matai, Surya Singh, and Murari Lal Mittal. 2010. Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches. In *Traveling Salesman Problem*, Donald Davendra (Ed.). IntechOpen, Rijeka, Chapter 1. https://doi.org/10.5772/12909

[10] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. 2004. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* 21, 6 (12 2004), 1087–1092. https://doi.org/10.1063/1.1699114 arXiv:https://pubs.aip.org/aip/jcp/article-pdf/21/6/1087/8115285/1087_1_online.pdf

[11] QuantaMagazine. 2020. *Computer Scientists Break Traveling Salesperson Record.* https://www.quantamagazine.org/computer-scientists-break-traveling-salesperson-record-20201008/

[12] Leonard Andreevich Rastrigin. 1967. *Random Search in Optimization Problems for Multiparameter Systems.* Technical Report. FOREIGN TECHNOLOGY DIV WRIGHT-PATTERSON AFB OHIO.

[13] Julia Jean Robinson. 1949. On the Hamiltonian Game (A Traveling Salesman Problem).

[14] Bib Paruhum Silalahi, Farahdila Sahara, Farida Hanum, and Hidayatul Mayyani. 2022. Simulated Annealing Algorithm for Determining Travelling Salesman Problem Solution and Its Comparison with Branch and Bound Method. *JTAM (Jurnal Teori dan Aplikasi Matematika)* 6, 3 (2022), 601–615.

[15] ChangYing Wang, Min Lin, YiWen Zhong, and Hui Zhang. 2015. Solving travelling salesman problem using multiagent simulated annealing algorithm with instance-based sampling. *International Journal of Computing Science and Mathematics* 6, 4 (2015), 336–353.

[16] Zicheng Wang, Xiutang Geng, and Zehui Shao. 2009. An effective simulated annealing algorithm for solving the traveling salesman problem. *Journal of Computational and Theoretical Nanoscience* 6, 7 (2009), 1680–1686.

[17] Dennis Weyland. 2008. Simulated Annealing, Its Parameter Settings and the Longest Common Subsequence Problem. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation* (Atlanta, GA, USA) *(GECCO '08).* Association for Computing Machinery, New York, NY, USA, 803–810. https://doi.org/10.1145/1389095.1389253

[18] what-when-how: In Depth Tutorials and Information. [n. d.]. *A comparison of cooling schedules for simulated annealing (artificial intelligence).* http://what-when-how.com/artificial-intelligence/a-comparison-of-cooling-schedules-for-simulated-annealing-artificial-intelligence/

[19] Shi-hua Zhan, Juan Lin, Ze-Jun Zhang, and Yiwen Zhong. 2016. List-Based Simulated Annealing Algorithm for Traveling Salesman Problem. *Comput. Intell. Neurosci.* 2016 (2016), 1712630:1–1712630:12. https://doi.org/10.1155/2016/1712630