Use non-traditional simulated annealing algorithm to solve the traveling salesman problem

Wang Yangxuan
Department of Computer Science University of Exeter
Exeter, UK
yw648@exeter.ac.uk

ABSTRACT

The traveling salesman problem (TSP) is an NP-hard problem and is one of the most important benchmark problems in optimization research. The simulated annealing methodSAis a heuristic algorithm which based on randomized improvement, and it often used in optimize problem. The TSPLIB is a library founded in 2001[1], which provides various different difficult level sample instances for the TSP, includes the currently researched best solution. This experiment tried to use the simulated annealing algorithm with a cooling enhancer [2]to solve the 101 cities TSP, and the dataset is provided by TSPLIB eil101[3]. In addition, we also implement Random Search and Hillclimb algorithm in the same dataset, compare the performance of different search algorithms on the TSP of the 101 cities.

KEYWORDS

Traveling Salesman Problem, TSP, Simulated Annealing, SA, Heuristic Algorithm, Evolutionary Optimization, TSPLIB

1 INTRODUCTION

The source of the traveling salesman problem is not yet clear, it was first posed as a mathematical problem in 1930. The problem can be simply described as: a salesman intends to visit several cities, each city only needs to visit once, and then return to the city of departure. What is his shortest route? [4] Generally, TSP problems can be modeled as symmetric TSP or asymmetric TSP. The former means that the weights of the two cities are the same, that is, an undirected graph is formed, while the latter means that there may be changes between the two cities. Different weights or single rows form a directed graph. The significance of studying the TSP problem is that many other sorting problems can be transformed into the expression of traveling salesman problems, such as robot path planning, workshop task allocation, and chromosome sequencing. [5,6] The traveling salesman problem is an NP-complete problem. Due to its high computational complexity, it cannot be solved by brute force search. However, many heuristic algorithms such as Genetic Algorithm (EA), Simulated Annealing algorithm (SA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) can be applied to TSP problems to find an approximate optimal solution.

The simulated annealing algorithm (SA) is a random optimization algorithm based on the Monte-Carlo iterative solution strategy. It is inspired by the annealing process of metals and was first proposed by N. Metropolis et al. in 1953. [7] In 1983, S. Kirkpatrick and others successfully introduced the annealing idea into the field of combinatorial optimization. The simulated annealing algorithm jumps out of the local optimal solution by accepting the new solution in the current solution field with a certain probability and approaches the global optimal solution. This experiment will use the non-traditional simulated annealing method, the Dynamic Simulated Annealing with Cooling Enhancer Modified Acceptance Probability (DSA-CE&MAP), [1] which introduces a new parameter cooling Enhancer on the basis of traditional SA to affect the cooling strategy, in order to achieve a more reasonable cooling effect. At the same time, a more reasonable selection probability P is introduced.

In addition, we implemented Hillclimb and Random Search on the same problem instance as the control experiment group. Compared with other heuristic algorithms, the Hillclimb method is simpler and usually produces better results than other algorithms when the search time is limited [1]. The Hillclimb method graph optimizes the objective function f(x), and each iteration is based on whether optimize f(x) to determine whether to accept the new solution x until it unable to find an improvement f(x) (usually when the old solution is equal to the new solution), it ends and outputs the current x as the optimal solution. For non-convex problems, the Hillclimb method will converge to the local maximum and cannot jump out and cannot reach the global maximum. The simulated annealing algorithm can be used to overcome this problem. The random search method used in this experiment will randomly generate N solutions, calculate the fitness (distance) of each solution, and then select the best solution as the initial optimal solution.

The rest of the report is organized as follows: the second part is a literature review, which describes the current research results of the traveling salesman problem and simulated annealing algorithm in the experimental literature; the third part will describe in detail the data set selected for the experiment and the specific TSP problem solved, the simulated annealing algorithm used and how the experiment was designed; the fourth part presents the experiment plan, parameter selection, and the analysis of the experimental results; the fifth part summarizes the experimental results and discusses the advantages and disadvantages of different algorithms for this type of optimization problem.

2 LITERATURE REVIEW

The main references in the experiment will be presented in this chapter. The 2.1 part is about the definition, development, and mainstream solution algorithm of TSP, the second part is about the detailed description and research development of the simulated annealing algorithm, and the third part presents some research on how the simulated annealing algorithm can be applied to solve the TSP problem.

2.1 Traveling Salesman Problem

The TSP problem has a long history of research. The problem of traveling around the knight studied by Euler in 1759 is the earliest description of it, that is, the knight needs to visit 64 squares on the chessboard, each square only goes once, and finally returns to the starting point. [8] It can be expressed by graph theory model as: given an undirected weighted graph

$$G = (V, E)$$

where

$$V = \{1, 2, \ldots, n\}$$

and

$$E = \{(dij)|i, j = 1, 2, ..., n, i \neq j\}$$

, where V is a set of n vertices representing all cities, and E is a set of edge set consisting of interconnected vertices i, j and edge weights(dij,which means the distance from i to j), the sum of the weights of all edges is the cost. The goal of the TSP problem is to find a Hamilton route with the shortest distance to minimize this cost. [9] Since the feasible solution of this problem is the full arrangement of all vertices, an increase in the number of vertices will lead to a combinatorial explosion. Early researchers used accurate algorithms to solve this problem. Commonly used methods include branch and bound method, linear programming method, dynamic programming, etc., but as the scale of the problem increases, accurate algorithms will be powerless, and some heuristic algorithms have gradually become the main research directions for solving TSP problems.

The Nearest Neighbor Algorithm (NN) is the first stochastic algorithm to be used to solve the TSP problem [10,11]. It is based on the greedy procedure, randomly selecting a city as the starting point, and taking the nearest city that has not been passed as the last one of the trips, until all the cities were visited. [10] The Genetic Algorithm (GA) developed by John Holland in the 1960s was also used in many studies to solve the TSP problem. The algorithm was inspired by the genetic operator in the theory of biological evolution: selection, crossover and mutation. [13] These genetic operators can generate new solutions and approach the optimal solution by adjusting the solution nodes in the TSP problem. One of the most effective genetic operators to drive search to a better solution is mutation [14,15,16]. Some studies in the late 1990s developed many variants of genetic operators. [17] In 1986, Fred Glover proposed the Tabu Search (TS) algorithm, [18]which is also widely used in TSP problems. Its advantage lies in the ability to set the tabu list to avoid some of the operations that have been experienced to improve the search efficiency. In addition, the Ant Colony Optimization (ACO) [13] proposed by Marco Dorigo in 1992, artificial ants will choose the path with higher pheromone concentration. Applying it to the TSP problem, all ants' paths compose the solution space. The distance between cities is inversely proportional to the pheromone concentration, which means that a better path has higher pheromone. Combining with tabu list makes each ant only follow the compliance route and find the optimal solution after multiple iterations. Moreover, the simulated annealing algorithm (SA) is also one of the popular algorithms widely used to solve the TSP problem, and it is also the main algorithm used in this experiment. Its details will be presented in section 2.2.

2.2 Simulated Annealing Algorithm

Simulated annealing (SA) is a meta-heuristic local search algorithm. Compared with the Hillclimb method, it can escape local optima and converge to an approximate optimal solution. It is usually used to solve discrete-continuous optimization problems [19] the so-called annealing refers to the principle of solid annealing. The physical process of annealing can be expressed as: when the solid is heated, the particles inside the object become disordered as the temperature increases, the internal energy increases, and when the temperature decreases, the particles tend to be stable, reaching an equilibrium state at every temperature, reaching the ground state at room temperature, with the smallest internal energy and the most stable molecular state.

The traditional Simulated Annealing algorithm can be divided into the following steps:

1) Initialization: set the initial temperature, the initial solution S, and the termination condition (the termination condition can be set when the temperature drops to a certain value or reaches the maximum number of iterations);

2) Use a certain perturbation method (such as exchanging the order of the two cities in the solution S) produces a new solution S';

3) Determine whether S' is better than S, if it is, replace S with S', otherwise accept S' with a certain probability;

4) Cool down, judge whether the termination condition is reached, if it is reached, output S and end loop, otherwise skip back to step 2.

However, the traditional SA algorithm has a slower convergence speed and poor performance when applied to a complex TSP [1].

As a result, many studies have provided us with some improved SA algorithms. Szu et al. proposed a Fast Simulated Annealing (FSA) method, which is a semi-local search. The classic simulated annealing algorithm has a fast cooling time, so the search efficiency is higher. [20] Ingber et al. proposed the Very Fast Simulated Re-annealing method (VFSR), which generates perturbations from model parameters based on the Cauchy-like distribution, and changes each iteration, so the search process converges faster. It points out that the annealing schedule of this method is better than Boltzmann annealing and fast Cauchy annealing. [21] In addition, some studies have combined SA and GA to solve signal analysis, language processing and other problems. [22]

2.3 SA for TSP

In addition to the DSA-CE&MAP method mentioned above, many scholars have studied various variants of SA algorithms or combined them with other algorithms to solve the TSP problem. Fang et al. combined the SA algorithm with Particle Swarm Optimization (PSO) and used SA to slow down the degradation of PSO swarm and increase its diversity to overcome the shortcomings of PSO, which is trapped in a local minimum. The research results show that this method can solve the TSP problem. The performance is better than other traditional optimization algorithms. [23] Liu et al. proposed a Hybrid Simulated Annealing Algorithm, which is based on the adaptive cooling schedule and combined with the Tabu Search Algorithm (TS) to provide an effective method to solve the TSP problem. [24] The Improved Genetic Simulated Annealing

Algorithm (IGSAA) was developed by Qing et al. This algorithm uses an improved genetic operator to improve efficiency and proposes an adaptive criterion to make the changes in the SA algorithm more adaptive. This method has been verified in experiments to improve the convergence speed of traditional SA and overcome the shortcomings of GA. [25] The Dynamic Simulated Annealing with Cooling Enhancer Modified Acceptance Probability (DSA-CE&MAP) used in this experiment was proposed by Vyas et al. and used to solve the traveling salesman problem. Its dynamics is reflected in the introduction of a new parameter "Cooling Enhancer", which can be based on the number of city nodes after each iteration is adjusted to the temperature parameter to adjust the cooling rate. Moreover, the Modified Acceptance Probability (MAP) helps to reduce the acceptance of solutions whose cost is much higher than the cost of the current solution, thereby significantly improving the search effect. [2] Research has shown that compared with the original SA algorithm, the use of this method significantly improves understanding the quality of. [2]

3 EXPERIMENT DESIGN

3.1 Problem Formulation

The TSP problem case that will be used in this experiment is eil101[3] provided by TSPLIB. This case contains a total of 101 cities, and each city is recorded in a .txt file with its horizontal and vertical coordinates. We randomly select a city as the starting point, and the salesman departs from this city to the next city. Each city goes once and only once. After completing all the cities, salesman returns to the original city to complete an solution. We use an array to represent a solution:

$$S = \{[x1, y1], [x2, y2], \ldots, [xn, yn], [x1, y1]\}$$

where [x,y] represents a city, and x, y are its abscissa and ordinate. Let

$$D = \sum_{i=1}^{n} \{(dij) | i = 1, 2, ..., n; j = i + 1; when\ i = n, j = 1\}$$

represent the total distance(include the distance of back to the first city) of one solution, namely cost, where

$$dij = \sqrt{(xj - xi)^2 + (yj - yi)^2}$$

represents the Euclidean distance between city i and city j. The goal of the TSP problem is to find S that minimizes D, that is, to find a travel path with the smallest total cost. According to the information of TSPLIB, the current optimal solution of case eil101 is 629, and the purpose of our experiment is to find S as close to D=629 as possible.

3.2 Algorithm Design

This section is divided into four parts. The first part describes the design of the non-traditional simulated annealing algorithm DSA-CE&MAP. The second and third parts respectively show the Hillclimb method and Random Search method as comparisons. The last part describes different designs of perturbation operators for finding neighbor solutions.

3.2.1 *DSA-CE&MAP.*

According to the paper by Vyas et al. [2], the steps of using the DSA-CE&MAP algorithm to solve the TSP problem are shown in Figure1. In our experiment: first we initialize the cooling rate, starting temperature T and ending temperature and then the new parameter cooling Enhancer is determined according to the number of cities. According to the data provided by the paper, for instance br17 in TSPLLIB, the acceptable probability calculated by the traditional SA method for the two new solutions (neighbor tour) a and b under the current conditions (the experimental setting of the paper) is 0.874 and 0.9811, and the acceptable probability calculated by DSA-CE&MAP is 0.6150 and 0.8236 [2]. It can be seen that the introduction of cooling Enhancer to reduce the acceptable probability, this is an important factor for this algorithm to perform better than basic SA.

Step 1: Initialize cooling_rate with a small value such as 0.001.
Generate an initial random tour x.

Step 2: Initialize T with a large value such as 100000.

Step 3: if totalCities < 30, then set coolingEnhancer = 0.5.
else if totalCities < 150 then set coolingEnhancer = 0.05.
else if totalCities < 750 then set coolingEnhancer = 0.005.
Otherwise, set coolingEnhancer = 0.0005.

Step 3: Repeat:
i. Generate next tour (x + Δx) by applying some operations on the current tour x.
ii. Evaluate ΔE(x) = E(x + Δx) − E(x), (i.e. neighborTourCost - currentTourCost):
if ΔE(x) < 0, keep the new state (i.e. new path distance less than current distance);
otherwise, evalutate ΔE` = $E_{bestSoFar}$(x) - E(x + Δx), (i.e. bestTourCost - neighborTourCost) and then accept the new state with
acceptance probability, $P = \frac{e-\Delta E/T}{e-\Delta E`/T}$
iii. If $E_{bestSoFar}$(x) > E(x + Δx), then set $E_{bestSoFar}$(x) = E(x + Δx).
iv. Set T = T − ΔT , ∀ ΔT= T x coolingEnhancer x cooling_rate.
until T is small enough.

Figure 1: description of DSA-CE&MAP algorithm [2]

In the iterative part of the algorithm, the first step is to generate the initial solution: randomly disrupt the city to generate a set of new solutions, and select the solution with the smallest cost (distance) among them (including the original order) as the initial solution S0; the second step is to generate a new solution: Use the perturbation operator to generate a set of neighbor solutions, and select the neighbor with the smallest cost as the new solution S1; the third step is to compare the distance D0 of S0 with the distance D1 of S1, and accept S1 with probability P as the new solution. When D1<D0, P=1, when D1 > D0, take a random number R in the interval [0,1), if R < P (the value of P is described in Figure 1, accept S1, otherwise Not accepted; the fourth step is to cool down: update

$$T = T - T \times coolingEnhencer \times cooling\_rate$$

then determine whether it reaches the end temperature, if it reaches the end temperature, it will exit the loop and output the current solution, otherwise skip back to the second step. After reach the termination condition, output the results.

3.2.2 *Hillclimb Algorithm.*

This experiment uses a simple Hillclimb algorithm as a control, and its design can be described as follows: first use the same method as presented in 3.2.1 to generate the initial solution S0; then generate 10 x city_number neighbors through the perturbation operator; then select the neighbor with the smallest cost as the new solution S1; if the distance D1 of S1 is less than the distance D0 of S0, replace S0 with S1, otherwise loop back to the second step; when the new solution is equal to the old solution, the

loop ends and output the final solution, because in the climbing algorithm, usually when the new solution and the old solution are equal, the local optimal solution is found.

3.2.3 Random Search. Random search is the most efficient and simple search method, but it usually performs the worst. Generate n solutions by randomly disrupting the order of cities and take the solution with the lowest cost as the final solution.

3.2.4 Perturbation Operator. The perturbation operator in this experiment refers to the method of generating neighbor solutions. We have designed three methods: 1. Randomly select two cities a and b in the solution, and exchange the positions of a and b to disrupt the original order to generate the neighbor solution; 2. Randomly select three cities a, b, and c, first exchange the positions of a and b and then exchange the positions of b and c to generate the neighbor solution; 3. Randomly select two cities a and b, and reverse the city order between a and b, include a and b to produce the neighbor solution. This method can be interpreted as: when

$$D(i, i+1) + D(j, j+1) > D(i, j) + D(i+1, j+1)$$

replace (i,i+1),(j,j+1) with edges (i,j) and (i+1,j+1), which can also be understood as using two convex quadrilaterals The opposite side replaces two diagonal lines, which can effectively solve the problem of route crossing.

4 CODE DESCRIBE

The code of this experiment can be divided into four parts.

- Data preparation: Read: load the data set, and save the abscissa and ordinate of all nodes (cities) as two lists x and y respectively; class Map: This class saves the coordinates of all cities as a travel path= [[x],[y]], and defines the function NBHDdistance that calculates the distance between two adjacent cities, calculates the total distance of a path, which includes the distance to all cities in the tour and the distance back to the departure city. In addition, the function exchange for exchanging two cities is defined.
- Algorithm implementation: class SA, class HC, and class RS respectively include all functions that implement the DSA-CE&MAP algorithm, Hillclimb algorithm and Random Search algorithm, as well as visualization of the initial state and final state, and the state of the nth iteration can be drawn.
- Parameter adjustment: alrun: The main run and tune parameters experiment part. In this part, adjust the state parameters to select different algorithms, and further adjust the parameters of the selected algorithm, such as temperature T, cooling rate, disturbance operator, etc.
- Run multiple experiments: Perform n experiments in a loop, calculate the average distance, the best distance, the worst distance, average performance and average running time.

5 EXPERIMENTATION

This chapter is divided into three parts. The first part introduces the experimental evaluation standards, the second part presents the test results of the three algorithms, and the third part compares and analyzes the test results.

5.1 Evaluation Criteria

According to the literature [2], this experiment uses Excess as the algorithm evaluation criteria, which is specified by the TSPLIB website [3], and the calculation formula is as follows:

$$Excess(\%) = \frac{SolutionValue - OptimumSolutionValue}{Optimum\ Solution\ Value} \times 100$$

In addition, we will also consider search time in the test.

5.2 Experimental result

In the experiment, each algorithm was tested 10 times, and the best and worst results and the average results were recorded.

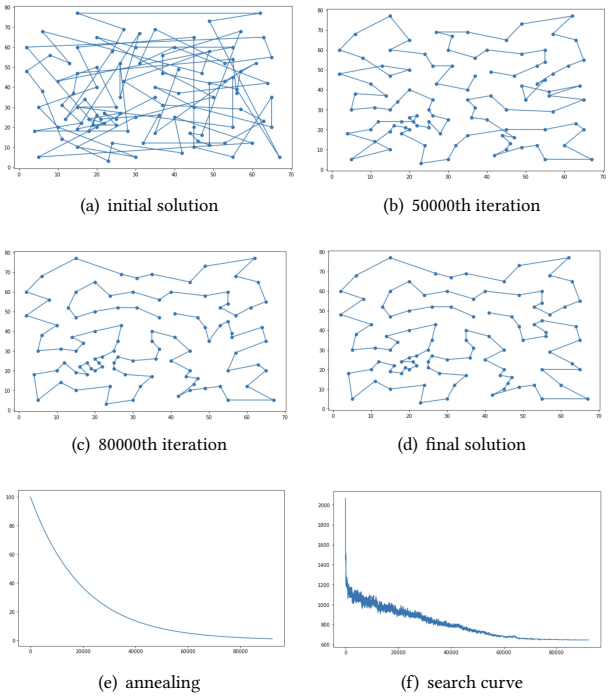
5.2.1 DSA-CE&MAP algorithm. The DSA-CE&MAP algorithm test uses different initial temperatures T=1000/100, the number of neighbors (range) n=5/10, cooling rate=0.01/0.001, and the same termination condition=T<1 (Simulated annealing algorithm usually convergence at T<1) and perturbation operator 3. The test results are shown in Table 1.

DSA-CE&MAP	Best	Excess	Worst	Excess	A_dist	A_excess	A_time
T=1000 Range=10 Coolrate=0.01	670.99	6.68%	697.78	10.94%	684.92	8.89%	23.80s
T=100 Range=10 Coolrate=0.01	671.88	6.82%	690.53	9.78%	680.79	8.23%	15.94s
T=100 Range=5 Coolrate=0.01	693.89	10.32%	719.38	14.37%	705.36	12.14%	11.25s
T=100 Range=10 Coolrate=0.001	645.85	2.68%	677.70	7.74%	664.22	5.60%	155.57s

Table 1: DSA-CE&MAP results

According to the experimental results, other parameters are the same. The performance of initial temperature T=1000 is slightly better than T=100. The average distance is 684.92 and 680.79 respectively. Correspondingly, the search time is increased by 7.86s; and the parameter range (number of neighbors) is increased. The performance improvement effect of the algorithm is more obvious. The average excess is 3.91% lower when range=10 than range=5; the parameter that has the most obvious impact on the algorithm performance is the cooling rate, because the smaller the cooling rate, the more detailed the search, and the greater number of iterations, correspondingly, the more time it takes. Under the conditions of T=100 and Range=10, when cooling rate=0.001, the performance of the algorithm is significantly improved compared to when cooling rate=0.01. The average distance is shortened from 680.79 to 664.22, and the average excess is optimized from 8.23% to 5.60%. On the contrary, the search time has also been dramatically extended by nearly 10 times. It is worth mentioning that the best result in 10 tests is 645.85 and its excess is 2.68%, which is a satisfactory result.

Figrue2 illustrates the result of the best parameter and the analysis is as follows: the path map of the initial solution is shown in subfigure (a), with a total distance of 2064.49; subfigure (b) is the result of the 50000th iteration, with a total distance of 715.52, and the crossing route is significantly reduced at this time; The 80000th



(a) initial solution (b) 50000th iteration

(c) 80000th iteration (d) final solution

(e) annealing (f) search curve

Figure 2: DSA-CE&MAP results

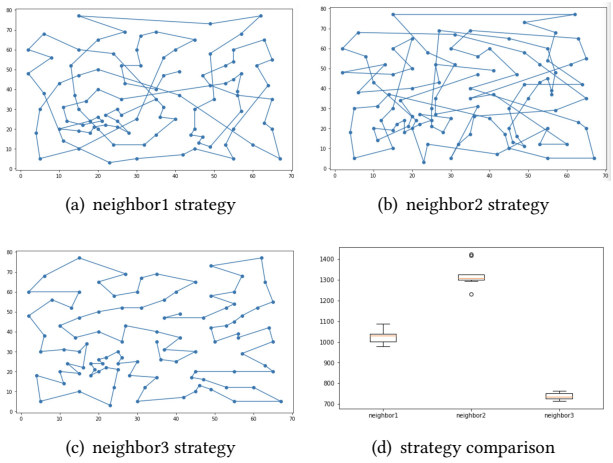
iteration are shown in the subfigure (c), we can find that there is no cross route at this time, and the total distance is 650.40; the final result is described in the subfigure (d), the total distance is also the optimal solution at 645.85, excess=2.68%. The subfigure (e) shows the cooling process, and the subfigure (f) clearly shows the change of the total distance during the iteration process. We can analyze that the total distance is drastically reduced to about 1500 at the beginning of the search, which means that the algorithm optimization is extremely fast. With 65,000 iterations, the optimization speed gradually slows down and fluctuates, but the fluctuations gradually decrease to stability as time increases. After 65,000 iterations, the optimization results basically converged, and the temperature is now close to the termination condition T<1.

5.2.2 Hillclimb algorithm. The Hillclimb algorithm test uses three different perturbation operators=neighbor/neighbor2/neighbor3, and the test results are shown in Table 2.

Hillclimb	Best	Excess	Worst	Excess	A_dist	A_excess	A_time
Neighbor1	979.46	55.72%	1088.13	72.99%	1027.89	63.42%	20.71s
Neighbor2	1257.01	99.84%	1397.81	122.23%	1316.89	109.36%	9.09s
Neighbor3	713.31	13.40%	764.38	21.52%	715.98	13.83%	14.16s

Table 2: Hillclimb results

The roadmap of the best results in the 10 tests of three different strategies and the comparison are shown in subfigures in Figure3. According to the experimental results, the performance of strategy neighbor3 is significantly better than the other two strategies, and



(a) neighbor1 strategy (b) neighbor2 strategy

(c) neighbor3 strategy (d) strategy comparison

Figure 3: Hillclimb results

strategy 2 (exchange three cities to generate neighbors) has the worst performance. Furthermore, We can clearly see that there are no cross paths in the results of the neighbor3 strategy, and there are obviously many cross paths in the results of the other two strategies. Consequently, the experimental results confirmed our idea that strategy three can effectively solve the problem of route crossing, which is the reason why strategy three is directly selected in the DSA-CE&MAP algorithm.

5.2.3 Random Search Algorithm. The Random Search algorithm tests the size solution space n=100/1000/10000/100000, and the test results are shown in Table 3. We clearly see from the test

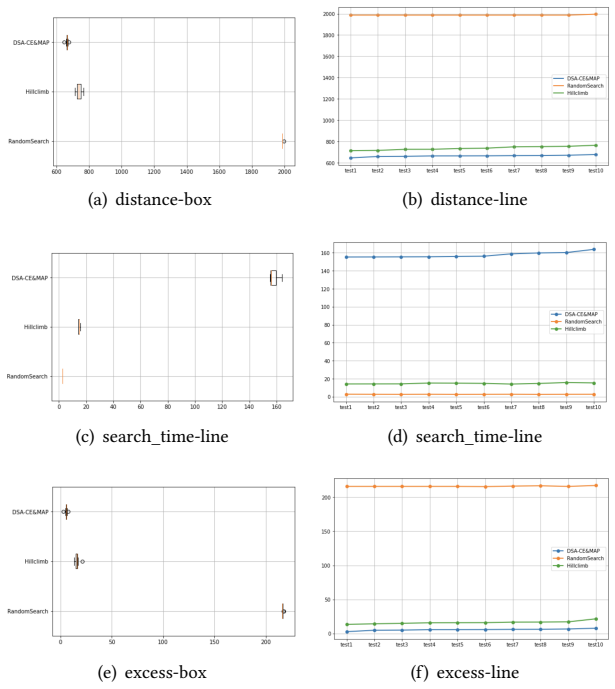
Random Search	Best	Excess	Worst	Excess	A_dist	A_excess	A_time
N=100	2008.84	219.37%	2042.92	224.79%	2026.50	222.18%	0.18s
N=1000	1987.10	215.91%	2027.54	222.34%	2004.406	218.67%	0.42s
N=10000	1987.10	215.98%	1995.69	217.28%	1987.96	216.05	2.80s
N=100000	1987.10	215.91%	1987.10	215.91%	1987.10	215.91%	26.75s

Table 3: Random Search results

results that the random search algorithm is an algorithm with a short search time, but the performance is the worst, and the size of the solution space has a very small effect on the optimization results. When N=100000, the results of 10 tests are the same and the shortest distance of which is 1987.10, which means that the best performance of the random search algorithm in this instance can only reach 1987.10.

5.3 Algorithm comparison and analysis

In this section, we compared these three different algorithms in terms of distance, search time, and excess. Figure 4 clearly illustrates box charts and line charts of test results comparison in best parameters. According to subfigure (a) and subfigure (b), we can easily conclude that the random search algorithm is not suitable for the TSP problem, on the contrary, both the DSA-CE&MAP algorithm and Hillclimb method have an acceptable optimized distance,



(a) distance-box (b) distance-line

(c) search_time-line (d) search_time-line

(e) excess-box (f) excess-line

Figure 4: Algorithms comparison

and the former has better performance, the distance is about 50 units shorter than Hillclimb on average. However, the RS is the fastest algorithm, and DSA-CE&MAP consumed the longest search time. Comparison of Excess has shown on subfigure (e) and subfigure (f), DSA-CE&MAP algorithm still the most ideal method, its performance is slightly better than Hillclimb, far beyond RS. Fundamentally speaking, if without consider the search time, DSA-CE&MAP is the best optimization algorithm in this experiment. The annealing process makes it jump out of the local optimal value and approach the global optimal value; if search time is also an optimization goal, Hillclimb is also a good choice.

6 CONCLUSION

This report discusses the background and development of the TSP problem and the feasibility of using heuristic algorithms to solve the TSP problem. Studies the traditional simulated annealing algorithm and its application and then focuses on the application of SA on TSP problems in different researches and the advantages of the DSA-CEMAP algorithm over the traditional SA algorithms. In terms of experiments, we implemented the DSA-CEMAP algorithm to solve the TSP problem in 101 cities, compared the effects of different parameters on the optimization process, and implemented a simple Hillclimb algorithm and random search algorithm as a comparison, and based on experimentswe analysis summarized the conclusion that compared with the control method, the simulated annealing algorithm has obvious advantages in optimizing the results, but it also takes more time to search. In further research, we may apply other improved SA algorithms such as SA combined with PSO or GA to multiple TSP problem instances, and compare and analyze the results to the traditional SA algorithms.

REFERENCES

[1] Travelling salesman problem. In Wikipedia, the free encyclopedia. Retrieved August 10, 2004, from https://en.wikipedia.org/wiki/Travelling_salesman_problem
[2] Vyas, A., Chawla, D. K., Thakar, U. Dynamic Simulated Annealing for solving the Traveling Salesman Problem with Cooling Enhancer and Modified Acceptance Probability.
[3] TSPLIB http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
[4] Hoffman, K. L., Padberg, M., Rinaldi, G. (2013). Traveling salesman problem. Encyclopedia of operations research and management science, 1, 1573-1578.
[5] Shahriari, S. (2020). Traveling salesman problem. Salem Press Encyclopedia of Science.
[6] Liang, X., Du, Z. (2020). Genetic Algorithm with Simulated Annealing for Resolving Job Shop Scheduling Problem. 2020 IEEE 8th International Conference on Computer Science and Network Technology (ICCSNT), Computer Science and Network Technology (ICCSNT), 2020 IEEE 8th International Conference On, 64–68. https://doi-org.uoelibrary.idm.oclc.org/10.1109/ICCSNT50940.2020.9305010
[7] Steinbrunn M, Moerkotte G, Kemper A. Heuristic and Randomized Optimization for the Join Ordering Problem [J]. The VLDB Journal, 1997, 6 (3) :8 - 17.
[8] Baidu Baike. Traveling saleman problem [EB/OL]. (2014- 3-17). http://baike.baidu.com/view/1162183.htm.
[9] Gong Gu, Hu Xiaoting, Hao Guosheng, Huang Yongqing, An Improved Genetic Algorithm for Traveling Salesman Problem. Computer Applications and Software, vol. 26, No. 4, pp. 87-89, 2009.
[10] Chauhan C, Gupta R, Pathak K (2012) Survey of methods of solving TSP along with its implementation using dynamic programming approach. Int J Comput Appl 52(4):12–19
[11] Wiak S, Krawczyk A, Dolezel I (2008) Intelligent computer techniques in applied electromagnetics. Springer, Berlin, p 190
[12] Kizilateş G, Nuriyeva F (2013) On the nearest neighbor algorithms for the travelling salesman problem. In: Advances in computational science, engineering and information technology. Advances in intelligent systems and computing, vol 225. pp 111–118
[13] Yang X-S (2010) Nature-inspired metaheuristic algorithms, 2nd edn. Luniver Press, Frome
[14] Rani K, Kumar V (2014) Solving travelling salesman problem using genetic algorithm based on heuristic crossover and mutation operator. IJRET 2(2):27–34
[15] Abdoun O, Jaafar A, Chakir T (2012) Analyzing the performance of mutation operators to solve the travelling salesman problem. Neural Evolut Comput Int J Emerg Sci 2(1):61–77
[16] Kirk J (2014) Travelling salesman problem-genetic algorithm. http://blogs.mathworks.com/community/2010/12/13/citing-file-exchange-submissions/
[17] Jean-Yves P (1996) Genetic algorithms for the travelling salesman problem. Ann Oper Res 63:339–370
[18] Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 13(5):533–549
[19] Rutenbar, R. A. (1989). Simulated annealing algorithms: An overview. IEEE Circuits and Devices magazine, 5(1), 19-26.
[20] Szu, H., Hartley, R. (1987). Fast simulated annealing. Physics letters A, 122(3-4), 157-162.
[21] Ingber, L. (1989). Very fast simulated re-annealing. Mathematical and computer modelling, 12(8), 967-973.
[22] Davis, L. (1987). Genetic algorithms and simulated annealing.
[23] Fang, L., Chen, P., Liu, S. (2007, February). Particle swarm optimization with simulated annealing for TSP. In Proceedings of the 6th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (pp. 206-210).
[24] Liu, Y., Xiong, S., Liu, H. (2009). Hybrid simulated annealing algorithm based on adaptive cooling schedule for TSP. In Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation (pp. 895-898).
[25] He, Q., Wu, Y., Xu, T. (2018). Application of improved genetic simulated annealing algorithm in TSP optimization. Control and Decision, 33(2), 219-225.