A Memetic Algorithm for the Close-Enough Traveling Salesman Problem

Zhenyu Lei*, Jin-Kao Hao*

*LERIA, Université d'Angers
2 Boulevard Lavoisier, 49045 Angers, France
zhenyu.lei@etud.univ-angers.fr, jin-kao.hao@univ-angers.fr

1 Introduction

The close-enough traveling salesman problem (CETSP) [8] is a variant of the traveling salesman problem (TSP) which is one of the most famous combinatorial optimization problems. Given N targets $\{v_1, v_2, \dots, v_N\}$ and a depot p_0 in the Euclidean plane, each target v_i has a disc neighborhood of radius r_i. The objective of the CETSP is to find the shortest Hamiltonian cycle $T = \{p_0, p_1, p_2, \dots, p_N, p_0\}$, which starts from and ends at the depot p_0, passing through every point p_i in the disc neighborhood of the target v_i.

Unlike the TSP where the salesman needs to visit the exact positions of the targets, in the CETSP, it is sufficient to pass through any point in the disc neighbourhood of targets. This feature introduces continuous variables in the CETSP. As can be seen, the CETSP is a complex problem that combines the discret optimization of visiting sequence like TSP with the continuous optimization of exact visiting positions.

At the same time, its characteristics may also lead to shorter trips than TSP in some specific applications, therefore, the CETSP has many practical applications in the real world, such as automated meter reading with radio frequency identification (RFID), wireless sensor network operations, and military and civilian missions with unmanned aerial vehicles (UAVs).

The CETSP has attracted significant research interest over the past two decades. To the best of our knowledge, the CETSP was first introduced in [8]. Some exact algorithms [2][6] were proposed to obtain the optimal solutions, but not practical for solving large size instances. A three-phase Steiner-zone heuristic was presented in [10]. [12] followed the idea of Steiner-zone and proposed a Variable Neighborhood Search (VNS) based heuristic approach. [3][4][5] proposed heuristic methods based on a discretization schema to explore the lower and upper bounds. Very recently, [7] presented a genetic algorithm combined with a very simple local search method and achieved excellent results, which proved the potential of memetic algorithm for the CETSP.

2 Proposed algorithm

We propose a heuristic approach based on memetic algorithm to solve the CETSP. Memetic algorithm [9,11] is a hybrid evolutionary algorithm combining the genetic algorithm and local search, which has proven to be efficient and effective in many combinatorial optimization problems.

The framework of the proposed algorithm is shown in Figure 1. For the moment, we only have a preliminary version which still needs to be improved. Our proposed algorithm follows the general schema of memetic algorithm. It maintains a population to represent a set of solutions. It explores the search spaces by generating iteratively offspring solutions through the crossover of parent solutions. Additionally, it employs the local search to improve the quality of the offspring solutions. The design of each component of the proposed algorithm will be described below.

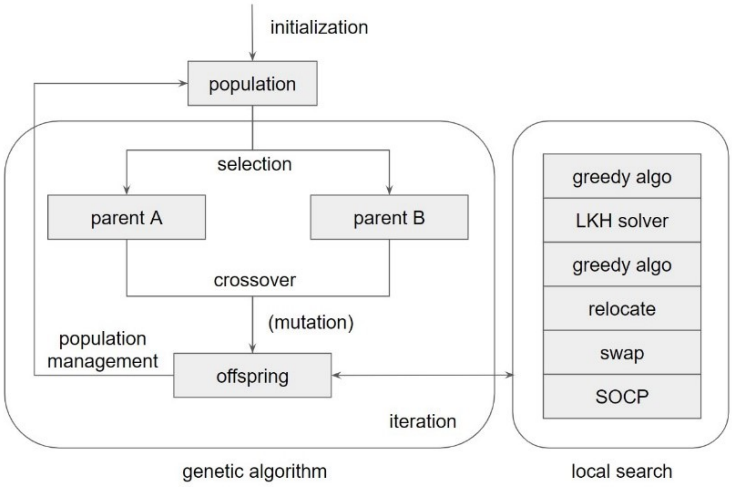

Figure 1: Framework of the proposed algorithm.

2.1 Solution representation

Considered in the Euclidean plane, the solution S can be represented by a sequence of visiting points $\{p_0, p_1, p_2, \dots, p_N, p_0\}$, where p_i represents the exact visiting point whose coordinates are (x_i, y_i), and p_0 represents the depot. Clearly, the cost of the solution can be represented by the sum of Euclidean distance between the connected visiting points. The objective is to find the minimum cost of the solution.

2.2 Genetic algorithm

In the proposed algorithm, the genetic algorithm focuses on diversification rather than intensification, which entails exploring more promising search spaces. Therefore, it is only responsible for solving the standard TSP which means exploring visiting sequence with fixed positions of visiting points. It has the following important components.

Initialization: A random generation method is adopted to initialize the population.

Selection: In each iteration, two parental solutions will be selected to perform crossover to produce one offspring. Due to the population management which will be described later, a good balance of intensity and diversity of solutions in the population will be maintained. Therefore, the two parents will be selected at random.

Crossover: Crossover is one of the most important operators in genetic algorithm. It can guide the direction of the search. A good crossover should inherit good features from the parents and maintain some diversity, and incite offspring solution to explore more promising search spaces.

In the current version of the proposed algorithm, the crossover described in [7] is adopted, which could be regard as a kind of multi-point crossover.

Mutation: Mutation is an important method to maintain the diversity of population. In the proposed algorithm, a random *Swap* operator is used as the mutation operator, which swaps the order of two random visiting points in the sequence. The operation is executed with a certain probability which is related to the number of stagnation iterations: $p = N_{si}/1000$.

Population management: To maintain the balance of intensity and diversity, population management needs to be carefully designed. First, we use a type of edit distance to define the distance or dissimilarity between two solutions: $dist = (1 - N_c/N) * 100\%$, where N is the number of given targets which is equal to the number of edges, N_c represents the number of common edges of the two solutions. Then, we define the minimum distance between a solution and other solutions in the population as the distance of the solution from the population.

$$fitness(s) = \frac{\alpha * rank(s, cost) + \beta * rank(s, dist)}{N_{pop}}$$

Finally, a fitness function is defined to evaluate the fitness score of solutions. The fitness function considers rank of both cost and distance in the population, where α and β are coefficients, function $rank(*)$ returns the rank of solution s in the population according to the cost or the distance, N_{pop} represents the number of individuals in the population. Note that the lower the fitness score, the better the fitness of the solution. The population will reduce to N_{pop} for every $N_{pop}/2$ iterations.

Stopping criteria: The algorithm stops when the given threshold of stagnation iterations th_{si} or the maximum number of iterations N_{mi} is reached.

2.3 Local search

Local search is the key component of the proposed algorithm. It will explore the search spaces intensively and do its best to find the best solution. We employ the variable neighborhoods descent (VND) as the local search schema, and design various search operators including sequence-only optimization (LKH solver), position-only optimization (greedy algorithm, SOCP) and joint optimization of sequence and position (*Relocate, Swap*). These operators are executed in the order shown in the Figure 1. The logic is that, we first alternatively perform the position-only

optimization and the sequence optimization (greedy algo - LKH solver – greedy algo). Note that the greedy algorithm will be executed twice because the first one optimizes the visiting positions of the offspring solution generated by the crossover, and it also can be regarded as a pre-processing for the LKH solver. The second optimizes the visiting positions in the sequence given by the LKH solver. We then perform the joint optimization of sequence and position. Finally, we obtain the final solution with SOCP. The details of these operators are as follows.

LKH solver: LKH solver is still the state-of-the-art among the TSP solver. We adopt it to solve the standard TSP by fixing the visiting positions.

Second-order cone programming (SOCP): The SOCP model for the CETSP was first proposed in [10]. Given a visiting sequence, the SOCP model can be built to find the optimal exact visiting positions in polynomial time. Note that SOCP is still a time-consuming operation.

Greedy algorithm: Considering the high cost of SOCP, we refer to the greedy algorithm proposed in [11] to replace SOCP before and after the LKH solver to optimize the visiting positions. The core idea is based on the geometric relationship between circles and points. For any three consecutive visiting points A, X, B, let O be the center of the disc where X is located, we can fix A and B to find the best position of X satisfying the constraints. There are several cases:

1. If both A and B are in the disc of O, any point in the line segment AB can be selected as X in theory. We directly choose A for convenience. If only one of them is in the disc, we choose the one inside.
2. If both of them are outside the disc O, there are two subcases. One is that the line segment AB intersects the disc O, we take the only one intersection point or the midpoint of two intersection points as X. The other is a bit more complicated, in which the problem turns into a well-known mathematical problem – Alhazen's problem [1]. In short, we need to find the X on the circumference of the disc O such that the sum $AO + BO$ is minimized. Considering the complexity of the problem, we take a binary search algorithm to obtain an approximate solution.

***Relocate* and *Swap*:** The above operators only consider the visiting sequence or the visiting position. The operators *Relocate* and *Swap* take into account both sequence and position. *Relocate* can remove a point from the sequence and insert it into a better position. *Swap* can exchange the visiting order of two disconnected points. These two operators take the best-improvement search strategy in a k-nearest-neighbors for every point, which means they take the best one after trying every move in the k-nearest-neighbors until no improvement can be found. The k-nearest-neighbors of a point are dynamically updated by the average of the historical positions of each point. An important feature is that these two operators will take the position into account. They recalculate the exact position of point according to the principles of the greedy algorithm introduced earlier. One difference is that, since the binary search algorithm employed in the case of Alhazen's problem is still a high complexity operation for the best-improvement search strategy, we propose another alternative strategy. Geometric method can prove that, for the optimal X^*, the straight line OX^* is the bisector of $\angle AX^*B$. Here we use the midline of $\angle AOB$ to

approximate the bisector of angle, and take the intersection point with the circumference of the disc O as X. Obviously, the straight line OX is also the midline of $\angle AXB$. This strategy may lose some precision, but it greatly improves computational efficiency, so it is acceptable.

3 Computational results

The proposed algorithm was evaluated on the benchmark given by [10], which contains 62 instances with different sizes from 30 to 1000 targets. There are three groups of instances: instances with varied overlap ratios (G1), instances with fixed overlap ratios (2%, 10% and 30%) (G2_0.02, G2_0.1 and G2_0.3), and instances with arbitrary radii (G3). The best-known solutions (BKS) are from all algorithms in the literature including GA [7] (the state-of-the-art algorithm) and are used as references to assess the effectiveness of the proposed algorithm.

In our preliminary experiments, the population size N_{pop} was set to 20; the threshold of stagnation iterations th_{si} was set to 500 and the maximum number of iterations N_{mi} was set to 10000. The stopping criteria is comparable to the refence algorithm GA [7]. We ran 20 times for every instance and recorded the best results. Table 1 shows the summary of our preliminary results compared to those of GA [7] and BKS on the different groups of benchmark instances. The column #instances indicates the number of instances in the corresponding group, and the column #optima shows the number of instances whose optimal solutions are known. The columns #wins #ties #losses respectively denote that the number of instances where our proposed algorithm achieve better, same and worse values compared to the references. In addition, in order to confirm the statistical difference of the results, the Wilcoxon signed-rank test with a confidence level of 0.05 is performed and the *p-value* is shown in the table.

It is obvious that the proposed algorithm provides comparable or better results than the BKS and GA [7]. The results show that the proposed algorithm reaches all 23 known optimal values and refreshes 30 values of the BKS in the 39 remaining instances. The *p-value* (<<0.05) indicates that the proposed algorithm statistically performs better than reference algorithms.

Table 1: Summary of computational results compared with BKS and GA [7].

group	#instances	#optima	Ours vs BKS				Ours vs GA[7]			
			#wins	#ties	#losses	*p-value*	#wins	#ties	#losses	*p-value*
G1	27	9	**13**	14	0	-	**13**	14	0	-
G2_0.02	7	0	**6**	1	0	-	**6**	1	0	-
G2_0.1	7	4	**3**	4	0	-	**6**	1	0	-
G2_0.3	7	7	0	7	0	-	0	7	0	-
G3	14	3	**8**	6	0	-	**8**	6	0	-
total	62	23	**30**	32	0	1.92E-06	**33**	29	0	5.71E-07

4 Conclusions

We propose a heuristic approach based on memetic algorithm to solve the close-enough traveling salesman problem. The designed VND local search method, especially the operators *Relocate* and *Swap*, greatly improved computing efficiency and play an important role. The preliminary experimental results have shown its effectiveness. However, it takes a long time for some instances, and some components still need to be refined. In future work, we will continue to optimize the algorithm and conduct more experiments to analyze the functions of components.

References

[1] Baker, M. (1881). Alhazen's problem. *American Journal of Mathematics*, *4*(1), 327-331.

[2] Behdani, B., & Smith, J. C. (2014). An integer-programming-based approach to the close-enough traveling salesman problem. *INFORMS Journal on Computing*, *26*(3), 415-432.

[3] Carrabs, F., Cerrone, C., Cerulli, R., & Gaudioso, M. (2017). A novel discretization scheme for the close enough traveling salesman problem. *Computers & Operations Research*, *78*, 163-171.

[4] Carrabs, F., Cerrone, C., Cerulli, R., & D'Ambrosio, C. (2017). Improved upper and lower bounds for the close enough traveling salesman problem. In *Green, Pervasive, and Cloud Computing: 12th International Conference, GPC 2017, Cetara, Italy, May 11-14, 2017, Proceedings 12* (pp. 165-177). Springer International Publishing.

[5] Carrabs, F., Cerrone, C., Cerulli, R., & Golden, B. (2020). An adaptive heuristic approach to compute upper and lower bounds for the close-enough traveling salesman problem. *INFORMS Journal on Computing*, *32*(4), 1030-1048.

[6] Coutinho, W. P., Nascimento, R. Q. D., Pessoa, A. A., & Subramanian, A. (2016). A branch-and-bound algorithm for the close-enough traveling salesman problem. *INFORMS Journal on Computing*, *28*(4), 752-765.

[7] Di Placido, A., Archetti, C., & Cerrone, C. (2022). A genetic algorithm for the close-enough traveling salesman problem with application to solar panels diagnostic reconnaissance. *Computers & Operations Research*, *145*, 105831.

[8] Gulczynski, D. J., Heath, J. W., & Price, C. C. (2006). The close enough traveling salesman problem: A discussion of several heuristics. *Perspectives in Operations Research: Papers in Honor of Saul Gass' 80 th Birthday*, 271-283.

[9] Hao, J. K. (2012). Memetic algorithms in discrete optimization. *Handbook of memetic algorithms*, 73-94.

[10] Mennell, W. K. (2009). *Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehicle routing problem, and sequence-dependent team orienteering problem*. University of Maryland, College Park.

[11] Moscato, P. (1999). Memetic algorithms: a short introduction. In D. Corne, M. Dorigo, F. Glover (Eds), *New ideas in optimization*, McGraw-Hill Ltd., Maidenhead, UK. 219–234.

[12] Wang, X., Golden, B., & Wasil, E. (2019). A steiner zone variable neighborhood search heuristic for the close-enough traveling salesman problem. *Computers & Operations Research*, *101*, 200-219.