Applied Soft Computing 49 (2016) 937–952

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

# Developing a dynamic neighborhood structure for an adaptive hybrid simulated annealing – tabu search algorithm to solve the symmetrical traveling salesman problem

Yu Lin [a], Zheyong Bian [b,*], Xiang Liu [b]

[a] College of Management & Economics, Tianjin University, 92 Weijin Road, Nankai District, Tianjin, 300072, China
[b] Department of Civil and Environmental Engineering, Rutgers, The State University of New Jersey, CoRE 736, 96 Frelinghuysen Road, Piscataway, NJ, 08854-8018, United States

ARTICLE INFO

*Article history:*
Received 31 March 2015
Received in revised form 18 May 2016
Accepted 20 August 2016
Available online 31 August 2016

*Keywords:*
Traveling salesman problem
Tabu search
Simulated annealing
Dynamic neighborhood structure
Adaptive parameters

ABSTRACT

This paper applies a hybrid *simulated annealing – tabu search* algorithm to solve the Traveling Salesman Problem (TSP). Fully considering the characteristics of the hybrid algorithm, we develop a dynamic neighborhood structure for the hybrid algorithm to improve search efficiency by reducing the randomness of the conventional 2-opt neighborhood. A circle-directed mutation is developed to achieve this dynamic neighborhood structure. Furthermore, we propose adaptive parameters that can be automatically adjusted by the algorithm based on context specific examples. This negates the need to frequently readjust algorithm parameters. We employ benchmarks obtained from TSPLIB (a library of sample instances for the TSP) to test our algorithm, and find that the proposed algorithm can obtain satisfactory solutions within a reasonable amount of time. The experimental results demonstrate that the proposed hybrid algorithm can overcome the disadvantages of traditional simulated annealing and tabu search methods. The results also show that the dynamic neighborhood structure is more efficient and accurate than the classical 2-opt. Also, adaptive parameters are appropriate for almost all of the numerical examples tested in this paper. Finally, the experimental results are compared with those of other algorithms, to demonstrate the improved accuracy and efficiency of the proposed algorithm.



## 1. Introduction

The Traveling Salesman Problem (TSP) is a classical combinatorial optimization problem, originally presented by Dantzig (1959).The TSP belongs to the class of NP-complete problems [1]. In other words, its computational complexity increases exponentially as the number of cities increases. The TSP has received widespread attention in the last 50 years and remains an important research topic. The TSP is often selected to test the performance of newly developed algorithms. Furthermore, the TSP can be assimilated to some other combinatorial optimization problems, such as vehicle routing problems [2–5], pickup and delivery problems [6,7], transportation and logistics problems [8–10], and assembly line sequencing and balancing problems [11].

The concept of TSP can be illustrated as follows. A salesman departs from a city, visits all cities once and only once, and finally returns to the city from which he departed. The objective is to minimize the total distance the salesman must travel to visit all of the cities. We employ the graph theory ($G = (V, A)$, where $V$ represents the set of cities and $A$ is the set of all the arcs between each set of two cities) to describe the problem. The problem can be formulated as follows [76]:

$$Minimize\ z = \sum_{i \in V} \sum_{j \in V} l_{ij} x_{ij} \tag{1}$$

$l_{ij}$ represents the distance between the cities $i$ and $j$. $x_{ij}$ is the decision variable. If the salesman travels from city $i$ to city $j$ directly, $x_{ij} = 1$, otherwise, $x_{ij} = 0$.

Subject to

$$\sum_{i \in V} x_{ij} = 1, j \in V \tag{2}$$

* Corresponding author at: Rutgers, The State University of New Jersey, Department of Civil and Environmental Engineering, CoRE 736, 96 Frelinghuysen Road, Piscataway, NJ 08854-8018, United States.
*E-mail addresses:* linyu@tju.edu.cn (Y. Lin), zb91@scarletmail.rutgers.edu, zheyongbian@gmail.com (Z. Bian), xiang.liu@rutgers.edu (X. Liu).

http://dx.doi.org/10.1016/j.asoc.2016.08.036
1568-4946/

$$\sum_{j \in V} x_{ij} = 1, i \in V \quad (3)$$

$\tilde{x}$ forms a Hamiltonian cycle, where $\tilde{x}$ represents the vector of decision variables $x_{ij}$ (4)

$$x_{ij} = \{0, 1\}, i, j \in V. \quad (5)$$

Formula (1) is the objective function that aims to minimize the total length that the salesman must travel. Formulas (2)–(4) ensure that the route forms a legal solution. Formula (5) signifies the range of decision variable $x_{ij}$.

In this paper, only symmetrical traveling salesman problems are considered, which indicates that $l_{ij} = l_{ji}$. In fact, $l_{ij}$ and $l_{ji}$ are identical elements that belong to the arc set $A$.

Many efficient heuristic algorithms have been developed to solve the TSP. Among these algorithms, simulated annealing and tabu search algorithms are among the simplest yet effective methods. They are widely applied in many different research fields, such as vehicle routing [2], scheduling [52–55], transportation and logistics [8,10], assembly sequencing [55–57], facilities location [58], etc. The effectiveness of simulated annealing and tabu search to solve the TSP has been verified by numerous studies [19–23,59–65]. However, both simulated annealing and tabu search have some disadvantages. Simulated annealing requires a sufficient number of iterations to guarantee a high-quality solution and is sensitive to parameter selection. In addition, repeated searches are inevitable because the search may return to old solutions and oscillate between solutions without the memory function. In contrast, tabu search algorithm has a fast decrease rate and can prevent repetitive searches by using a tabu list function. However, it cannot guarantee convergence. Its performance largely depends on the initial solution. Moreover, owing to lack of effective control, the later stages of the tabu search process seem to move randomly at a comparatively low objective function value. Therefore, we incorporate the Metropolis acceptance criteria [66] of simulated annealing into tabu search to control the movement during the search process. A hybrid *simulated annealing –tabu search* algorithm is expected to eliminate some of the main weaknesses of simulated annealing and tabu search.

In addition, the effectiveness of both tabu search and simulated annealing algorithms largely depends on the neighborhood structure. The most commonly used method is the classical λ-opt neighborhood structure [67,68]. However, λ-opt has a random component. In other words, during iterative process of an algorithm, the λ-opt randomly generates neighborhood solutions. We fully consider the properties of the hybrid algorithm and propose an effective neighborhood structure, called dynamic neighborhood structure. A circle-directed mutation is developed to achieve the dynamic neighborhood structure. This structure is more adaptive and can eliminate a certain amount of the randomness of traditional neighborhoods. In other words, low-quality mutation can be avoided through the dynamic neighborhood. A detailed description of the structure will be provided in Section 3.1. The experimental results demonstrate that this dynamic neighborhood structure can significantly reduce computational time and improve the quality of solutions obtained.

Finally, since the simulated annealing algorithm is sensitive to parameters, and each iteration of our algorithm is conducted by simulated annealing, the setting of parameters affects the performance of the hybrid algorithm. Thus, this paper proposes a set of reasonable parameters. We first analyze the influence of some characteristics of a specific benchmark on the setting of parameters and subsequently propose a set of adaptive parameters that can be automatically adjusted by the algorithms based on these characteristics. The computational complexity of the developed algorithms is only $O(N^{1.1})$ based on the proposed adaptive parameters, where $N$ is the number of cities representing the scale of a problem.

**Table 1**
Heuristic methods applied to TSP.

| References | LK | SLS | SA | TS | GA | ACO | PSO | HBMO | NN |
|---|---|---|---|---|---|---|---|---|---|
| [12–17,71–73] | √ | | | | | | | | |
| [12,15,68] | | √ | | | | | | | |
| [18–21,59–62] | | | √ | | | | | | |
| [22,23,61] | | | | √ | | | | | |
| [25–27,29,30] | | | | | √ | | | | |
| [32–37,78] | | | | | | √ | | | |
| [38–41,79] | | | | | | | √ | | |
| [42,43] | | | | | | | | √ | |
| [44–47,74] | | | | | | | | | √ |
| [22] | √ | | | √ | | | | | |
| [31] | | √ | | | √ | | | | |
| [24,28] | √ | | | | √ | | | | |
| [48] | | √ | | | | √ | | | |
| [50,51] | | | | | √ | √ | | | |
| [49] | | | √ | | √ | √ | √ | | |
| [75] | | √ | | | | √ | √ | | |
| [77] | | | | | | √ | | √ | |

*Notes*:
LK: Lin-Kernighan implementations and Lin-Kernighan-based heuristics.
SLS: Simple local search algorithms.
SA: Simulated annealing algorithms.
TS: Tabu search algorithms.
GA: Genetic algorithms.
ACO: Ant colony optimization algorithms.
PSO: Particle swarm optimization algorithms.
HBMO: Honey bee mating optimization algorithms.
NN: Neural networks.

This paper is structured as follows: Section 2 provides a literature review on recent related works. Section 3 introduces the adaptive hybrid simulated annealing – tabu search algorithm with a dynamic neighborhood structure. The computational experiments and results are presented in Section 4. Finally, conclusions are provided in Section 5 and related future work is described in Section 6.

## 2. Literature review

### 2.1. Existing algorithms

In the past two decades, many efficient heuristic algorithms have been developed to solve the TSP. These include simple local search algorithms (such as 2-opt and 3-opt local search algorithms), Lin-Kernighan (LK) implementations and Lin-Kernighan-based heuristics, simulated annealing (SA), tabu search (TS), genetic algorithms (GA), ant colony optimizations (ACO), particle swarm optimizations (PSO), newly developed honey bee mating optimizations (HBMO), neural networks (NN), and some hybrid algorithms. Table 1 summarizes the heuristic algorithms applied to solve the TSP.

In recent years, a number of studies are focusing on the improvement of traditional heuristics, mainly through two methods: (1) developing hybrid algorithms to overcome the disadvantages of individual algorithms, and (2) developing effective mutation methods for the iterative process.

Hybrid algorithms can combine the advantages and overcome the disadvantages of individual algorithms. Thus, hybrid algorithms are usually more effective than each individual algorithm. Nguyen et al. [24] and Baraglia et al. [28] developed a hybrid algorithm combining the genetic algorithm (GA) with the Lin-Kernighan heuristic for the TSP and successfully found high-quality solutions. Wang [31] proposed two novel local searches and incorporated them

into a genetic algorithm. The hybrid genetic algorithm found better approximate solutions than the traditional genetic algorithms. Tsai et al. [48] introduced the multiple ant colonies concept from parallel genetic algorithms to search for solutions and proposed two novel local searches for their algorithms. Chen & Chien [49] combined the genetic algorithm, simulated annealing, ant colony system, and particle swarm optimization to solve the TSP and obtained satisfactory solutions. Chen & Chien [50] and Dong et al. [51] developed hybrid algorithms combining the genetic algorithm with ant colony optimization for the TSP. Both of them verified the effectiveness of the proposed algorithms. Mahi et al. [75] employed a hybrid method based on particle swarm optimization, ant colony optimization, and 3-opt algorithms to solve the TSP. They demonstrated that the algorithm can overcome the disadvantages of individual algorithms used.

On the other hand, the efficiency and accuracy of heuristic algorithms largely depend on mutation methods. Mutation represents the change in solutions between two iterations, such as the neighborhood structure of local-search-based algorithms, crossover in genetic algorithms, mutation defined by the PSO formula in particle swarm optimization, etc. The Lin-Kernighan heuristic [13] proposed in 1973 is one of the most effective mutation methods for local-search-based algorithms. Geng [21] incorporated three neighborhood structures into a simulated annealing algorithm to improve search efficiency. For some swarm algorithms, several researchers also proposed improvements for mutation methods. For example, Nagata and Kobayashi [30] proposed edge assembly crossover for genetic algorithms. The experiment resulted in high-quality solutions obtained by the designed algorithm. Marinakis et al. [43] designed a multiple phase neighborhood search-greedy randomized adaptive search procedure (MPNS-GRASP), expanding neighborhood search (ENS), and a new crossover operator based on an adaptive memory procedure for the honey bees mating optimization algorithm to solve the TSP with high efficiency and accuracy. Both Albayrak & Allahverdi [29] and Wang [31] developed new local search mutations for genetic algorithms. Escario et al. [37] developed two new search strategies for ant colony optimization (ACO) and formed the ant colony extended (ACE) algorithm. Their experiment showed the ACE's superiority over the traditional ACO.

### 2.2. Knowledge gaps

Lin-Kernighan implementations and Lin-Kernighan-based heuristics (LK) and simple local search (SLS) algorithms have strong ability of local search but lack global search ability. In contrast, swarm algorithms, genetic algorithms (GA), ant colony optimization algorithms (ACO) and honey bee mating optimization algorithms (HBMO) have strong ability of global search, but have relatively low ability of local search. Simulated annealing and tabu search have both strong local search ability and global search ability. The hybrid simulated annealing – tabu search algorithm's advantages can overcome each individual algorithms' disadvantages. In fact, designing hybrid algorithms and advanced mutation methods is an emerging research subject for solving combinatorial optimization problems. However, very little prior research has demonstrated the effectiveness of a hybrid simulated annealing – tabu search algorithm to solve the TSP. In addition, most of the advanced mutation methods in the literature are static, except a few variable neighborhood structures like Lin-Kernighan neighborhoods. Although these variable neighborhoods are dynamic, they still exhibit considerable randomness in neighbor solution selection during iterative process and are not adaptive to different iterative stages. Finally, parameters have an influence on the performance of an algorithm, yet this influence has received relatively little attention. Geng et al. [21] alone has proposed a set of adaptive parameters for simulated annealing using a greedy search, but their parameter setting techniques are not entirely applicable to our algorithms. Moreover, they did not study the relationship between the characteristics of numerical examples and the setting of parameters. All these deficiencies in the current work incentivized this paper to propose a hybrid simulated annealing – tabu search algorithm based on dynamic neighborhood structure with adaptive parameters for the TSP.

### 2.3. Intended contributions of this paper

To narrow the above-mentioned knowledge gaps, this paper aims to bring the following contributions to the body of knowledge:

- This paper represents one of the earliest studies that apply hybrid simulated annealing – tabu search algorithm to solve the TSP.
- This paper proposes the concept of dynamic neighborhood structure, which aims to promote the algorithms' adaptiveness to different iterative stages and reduce randomness of traditional mutation methods. This is an improvement to previous neighborhood search methods.
- This paper studies the relationship between the characteristics of context specific examples and the parameters used in the algorithm. Adaptive parameters are determined based on characteristics of specific examples.

## 3. The adaptive hybrid *simulated annealing – tabu search* algorithm with a dynamic neighborhood structure based on a circle-directed mutation (AHSATS-D-CM)

In this section, we review the classical neighborhood 2-opt and provide a detailed introduction for the dynamic neighborhood structure based on a circle-directed mutation. Then two hybrid *simulated annealing – tabu search* algorithms are introduced in Section 3.2, one based on classical 2-opt (AHSATS-2-opt) and the other based on the dynamic neighborhood structure using the circle-directed mutation (AHSATS-D-CM). Finally, in Section 3.3, we analyze several characteristics of numerical examples and propose the adaptive parameters for the two hybrid algorithms according to these characteristics.

### 3.1. The dynamic neighborhood structure based on the circle-directed mutation (D-CM)

A dynamic neighborhood structure is defined as the neighbor solutions that are dynamically selected and are adaptive to different stages of an algorithm. Therefore, the algorithm can reduce randomness of neighbor solution generations and avoid low-quality mutations. In this paper, the dynamic neighborhood structure is designed for the classical 2-opt neighborhood structure.

Two-opt is one of the simplest neighborhood structures, but it is effective for simulated annealing and tabu search [21,22]. It can be described as follows: two links in the current tour are replaced by two new links. In other words, the new tour is obtained by deleting two links and joining the resulting paths together in a new arrangement.

Most meta-heuristic algorithms have the iterative process in which low-quality solutions are generated at the initial stage, with the quality of solutions improving as the number of iterations increases, and finally high-quality solutions are obtained at the later stage. Our hybrid algorithm is no exception. The common classical 2-opt neighborhood structure has a randomized component, in which the two edges are randomly selected to be replaced by two new edges. The random 2-opt can perform global searches at the initial stage of an algorithm when the objective function value of solutions reached is relatively high. However, owing to

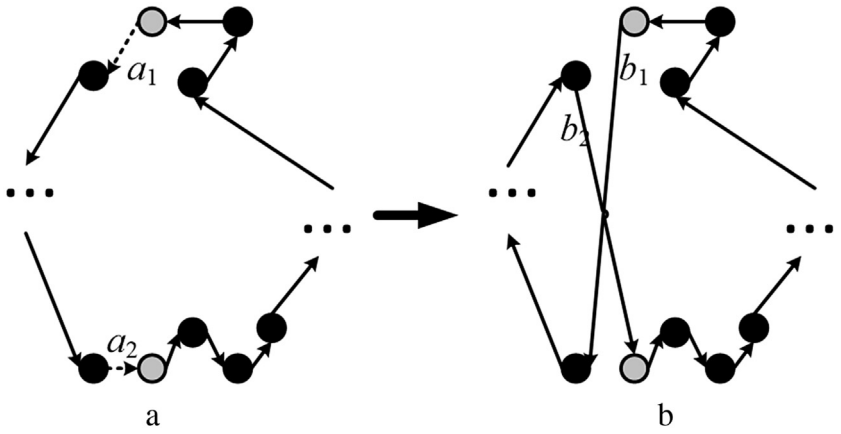


**Fig. 1.** A random mutation of 2-opt based on a short tour.

its randomness, it becomes inefficient when better-quality solutions are obtained at the later stage of an algorithm. It will expend unnecessary search time and even break the high-quality solutions obtained at the later stage of an algorithm. This can be demonstrated through the following example. Fig. 1 (a) represents a short tour obtained by an algorithm during a later stage of execution. At this moment, the algorithm generates a random mutation of 2-opt, as Fig. 1 (b) shows, with a significant increase in the total length, because $b_1 + b_2 - a_1 - a_2$ is a number much larger than zero. Although such mutations may be generated as solution candidates, it is unlikely to be selected by the tabu search. Thus, such random mutations will expend unnecessary search time during the later stage of the algorithm. Fig. 2 demonstrates that the hybrid algorithm proposed in this paper has a very fast decrease rate. This indicates that short tours are obtained in few iterations. The algorithm will be inefficient and perform many unnecessary iterations if the random 2-opt neighborhood structure is adopted for the overall search process.

To overcome the disadvantage of random searches during the search stage, we develop a dynamic neighborhood structure using circle-directed mutation in place of the random mutation.

Fig. 3 (a) represents a short tour obtained during the later stage of execution. We first find a random location in the graph $G = (V, A)$, where $V$ is the set of all vertexes (i.e. cities) and $A$ is the set of all arcs connecting any two vertexes. We then draw a circle with a diameter of $r$, which is longer than $p$ percent of the arcs ($A$) in graph G, and then reverse the visiting order of the cities between two randomly selected vertexes (cities) within the circle. Here, we define the diameter of the circles by determining the percentage $p$. First, we sequence all arcs ($A$) in the graph from the shortest to the longest; we then obtain the percentage $p$ of the shortest arcs (the

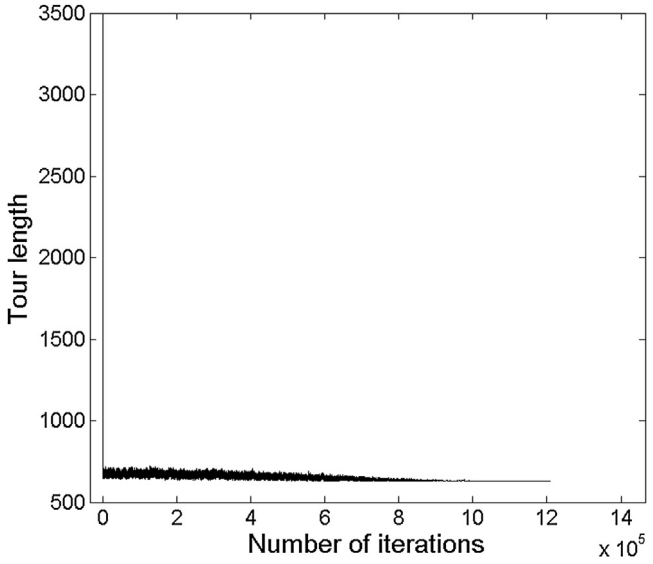


**Fig. 2.** Iterative process of the hybrid simulated annealing – tabu search algorithm.

set is defined as $PA$) that accounts for the arcs in set $A$. The longest arc in $PA$ is the circle's diameter. We take Fig. 3 as an example. Given a specific value of percentage $p$, the diameter of the circles $r$ is thus determined. Hypothetically, the set $PA$ contains these arcs: (1, 2), (2, 3), (2, 7), (2, 8), (3, 4), (3, 7), (3, 8), (3, 9), (4, 5), (4, 8), (4, 9), (6, 7), (7, 8), (8, 9), (9, 10), etc. whose lengths are all no greater than the diameter $r$. We randomly select a pair of vertexes from $PA$; in this example, the pair (3, 8) is selected. Then, we reverse

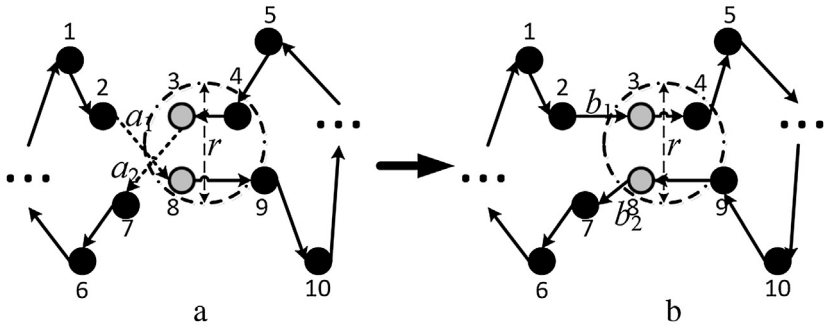


**Fig. 3.** A circle-directed mutation of 2-opt based on a short tour.

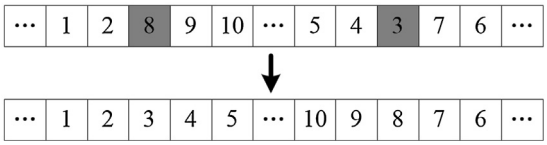


**Fig. 4.** An example of mutation.

the visiting order of the cities between nodes 3 and 8, as Fig. 4 shows. Poor-quality mutations, such as a reversal of the traversal sequence between vertexes 1 and 10, are excluded by the circle restriction. This circle-directed mutation decreases the probability of poor-quality mutations during the later stage of an algorithm because the circle restricts the length of newly generated edges ($b_1$ and $b_2$ in Fig. 3).

The percentage $p$ of the shortest arcs that accounts for the arcs in set $A$ is not a constant. At the initial stage of the hybrid algorithm, the tour lengths reached are relatively long. Therefore, the circles should be large enough to ensure that all pairs of vertexes have a probability of being selected to achieve the mutation so that the algorithm has global search capability. In contrast, when shorter tours are generated in the later stage of the algorithm, the diameter of circles should be decreased to rule out poor-quality mutations. Therefore, the mechanism of the hybrid algorithm requires that the percentage $p$ should begin at a high ratio and end at a relatively low value. The objective function value (the tour length) decreases very fast at the initial stage of the hybrid algorithm, while its decrease rate slows down during the late stage of the search process (as Fig. 2 shows). The decrease rate of the percentage $p$ should correspond to that of the objective function value. We anticipate that it is appropriate for the percentage $p$ to decrease at an exponential rate whose decrease rate is high at the initial stage while slowing at the later stage. The dynamic neighborhood structure is thus proposed based on considerations above, and the detailed settings of the structure are described in Sections 3.2 and 3.3.

### 3.2. The detailed mechanism of AHSATS-2-opt and AHSATS-D-CM

For AHSATS-2-opt, simulated annealing begins at the initial temperature ($t_{start}$), and ends at the final temperature ($t_{end}$). The temperature is decreased at an exponential rate via multiplication by a cooling coefficient ($t_{cool}$) which is slightly lower than 1. At each temperature, SA performs *Elen* (epoch length) iterations and accepts a newly generated solution ($X_{opt}$) with a probability of $\rho$ at each iteration. The tabu search is responsible for generating a new solution ($X_{opt}$) based on a current solution ($X_{current}$). We employ the 2-opt neighborhood to generate *CN* candidate solutions. Among the candidate solutions, the solution with the shortest tour length that conforms to the aspiration criterion or is not in the tabu list is selected as $X_{opt}$ – the solution waits to be accepted (or not) by the Metropolis acceptance criteria of simulated annealing. The structure of AHSATS2-opt is listed in pseudo-code in Table 2.

For AHSATS-2-opt, the operational rules of SA are the same as those of AHSATS-2-opt. We employ the dynamic neighborhood structure proposed in Section 2.1, instead of the classical 2-opt usually used in tabu search. In the initial stage of the algorithm, the objective function value is high (as Fig. 2 shows); thus the percentage $p$ must be initiated at a high value ($p_{start}$) to ensure a globally scoped search; simultaneously, the initial number of candidate solutions ($CN_{start}$) generated by the tabu search is comparatively large. However, the objective function value decreases rapidly during the search process. Therefore, the percentage $p$ and the number of candidate solutions *CN* should be decreased to avoid unnecessary calculations. We stipulate that the percentage $p$ and the number of candidate solutions (*CN*) decrease as the temperature decreases

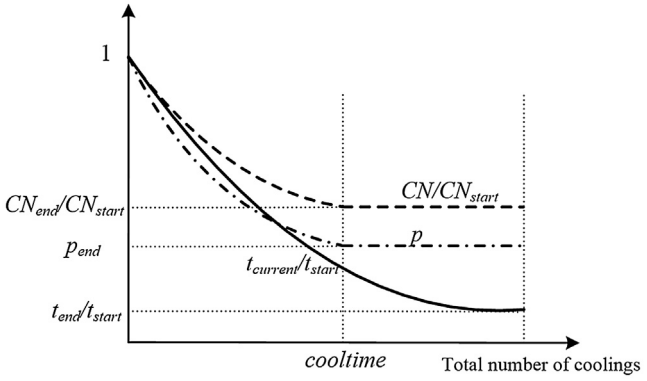


**Fig. 5.** Dynamic parameters of AHSATS-D-CM.

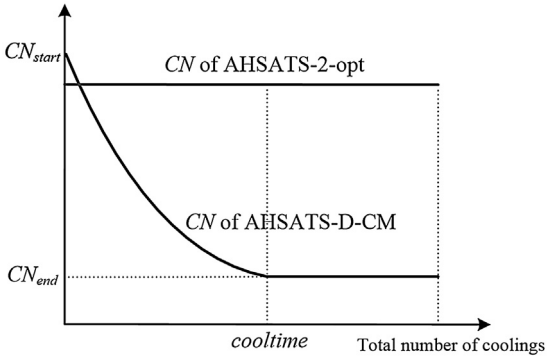


**Fig. 6.** Comparison of the number of candidate solutions generated between AHSATS-2-opt and AHSATS-D-CM.

every time, as Fig. 5 shows. The two values also decrease at an exponential rate via multiplication by the decrease coefficients $p_{cool}$ and $CN_{cool}$, respectively. Both of them stop decreasing when $p_{end}$ and $CN_{end}$ are reached. This decrease is performed *cooltime* times. The comparison between the *CN*s of AHSATS-2-opt and AHSATS-D-CM during the search process is shown in Fig. 6. We can conclude from Fig. 6 that, compared with AHSATS-2-opt, AHSATS-D-CM generates fewer neighbor solutions and requires fewer calculations of tour length differences.

The structure of AHSATS-D-CM is given in Table 3 and Fig. 7.

### 3.3. The adaptive parameters

For almost all algorithms, parameters must be adjusted to adapt to different numerical examples. It is time-consuming to adjust the parameters for different numerical examples every time an algorithm is executed, especially for algorithms in which parameters have a significant influence on the solution quality. We focus on proposing a set of adaptive parameters that can be adjusted automatically according to the characteristics of a specific numerical example. We have analyzed several characteristics of the benchmarks in TSPLIB and their influence on the parameters in our algorithms. Based on this analysis, a set of adaptive parameters is determined by testing benchmarks from TSPLIB.

#### 3.3.1. Four scalars representing the characteristics of numerical examples

Before the adaptive parameters are proposed, we first define four scalars that can reflect several characteristics of the benchmarks. These scalars will be used in the following parameter settings.

**Table 2**
Algorithm 1: AHSATS-2-opt.

Input the data of a numerical example
Initialize the parameters of the algorithm: $t_{start}$, $t_{end}$, $t_{cool}$, $CN$, $TL$ (length of tabu list), $Elen$ and set $t_{current} = t_{start}$, $it = 0$, $X_{current} = X_0$, $E_{current} = f(X_0)$, $X_{best} = X_0$, $E_{best} = f(X_0)$
**Do while** $t_{current} > t_{end}$
  **Do while** $it < Elen$
    Use the neighbor structure (2-opt) to generate $CN$ candidate solutions $\{X_1, X_2, \ldots, X_{CN}\}$ of $X_{current}$'s neighbors.
    Calculate $\{\Delta f_1, \Delta f_2, \ldots, \Delta f_{CN}\}$ ($\Delta f_i = f(X_i) - E_{current}$) and record the subscript $opt$, where $\Delta f_{opt} = \min\{\Delta f_1, \Delta f_2, \ldots, \Delta f_{CN}\}$.
    **If** $E_{current} + \Delta f_{opt} < E_{best}$ (the aspiration criterion)
      $X_{current} = X_{opt}$
      $E_{current} = E_{current} + \Delta f_{opt}$
      $X_{best} = X_{opt}$
      $E_{best} = E_{current} + \Delta f_{opt}$
      Update the tabu list
    **Else**
      **Do while** the mutation from $X_{current}$ to $X_{opt}$ is in tabu list
        Select the suboptimal solution as $X_{opt}$ from $\{X_1, X_2, \ldots, X_{CN}\}$
      **End do**
      **If** $\Delta f_{opt} < 0$ || $\rho < rand$ (see formula 10)
        $X_{current} = X_{opt}$
        $E_{current} = E_{current} + \Delta f_{opt}$
        Update the tabu list
      **End if**
    **End if**
    $it = it + 1$
  **End do**
  $t_{current} = t_{current} \times t_{cool}$
**End do**
Output $X_{best}$ and $E_{best}$.

1) $N$: number of cities.

The number of cities reflects the scale of a numerical example. The parameters in most algorithms depend on the scale of numerical examples, and our algorithms are no exception.

2) $$\alpha = \frac{|a|}{A},\ a = \left\{ l \in A \mid |l| \le \max_j \min_i (|l_{ij}|),\ l_{ij} \in A; i, j \in V \right\} \quad (6)$$

If the variable between the vertical bars (| |) represents a set (such as $a$ and $A$), it receives the number of elements in the set; if the variable between the bars represents an arc (such as $l$), it receives the length of the arc.

The scalar $\alpha$ is used in the dynamic neighborhood structure. It determines the value of $p_{end}$, which must be large enough (compared to the ratio $\alpha$) to ensure that every vertex (i.e. city) has a sufficient probability of being selected to generate neighbor solutions.

3) $$\beta = \sum_{l \in K} |l|,\ K = \left\{ l \mid |l| = \min_i (|l_{ij}|),\ l_{ij} \in A; i, j \in V \right\} \quad (7)$$

where $|l|$ is the length of arc $l$.

The scalar $\beta$, which is the sum of the lengths of the shortest arcs linking all the vertexes, is used to determine the acceptance probability $\rho$ in simulated annealing. In general, the initial temperature $t_{start}$ and the final temperature $t_{end}$ in simulated annealing

**Table 3**
Algorithm 2: AHSATS-D-CM.

Input the data of a numerical example
Initialize the parameters of the algorithm: $t_{start}$, $t_{end}$, $t_{cool}$, $CN_{start}$, $TL$, $CN_{cool}$, $p_{cool}$, $Elen$ and set $p$=1, $CN$=$CN_{start}$, $t_{current}$ = $t_{start}$, $it$=0, $X_{current}$ = $X_0$, $E_{current}$ = $f(X_0)$, $X_{best}$ = $X_0$, $E_{best}$ = $f(X_0)$
**Do while** $t_{current} > t_{end}$
  **Do while** $it < Elen$
    Use the dynamic neighborhood structure to generate $CN$ candidate solutions $\{X_1, X_2, \ldots, X_{CN}\}$ of $X_{current}$'s neighbors.
    Calculate $\{\Delta f_1, \Delta f_2, \ldots, \Delta f_{CN}\}$ ($\Delta f_i = f(X_i)$ - $E_{current}$) and record the subscript $opt$, where $\Delta f_{opt} = \min\{\Delta f_1, \Delta f_2, \ldots, \Delta f_{CN}\}$.
    **If** $E_{current} + \Delta f_{opt} < E_{best}$
      $X_{current} = X_{opt}$
      $E_{current} = E_{current} + \Delta f_{opt}$
      $X_{best} = X_{opt}$
      $E_{best} = E_{current} + \Delta f_{opt}$
      Update the tabu list
    **Else**
      **Do while** the mutation from $X_{current}$ to $X_{opt}$ is in tabu list
        Select the suboptimal solution as $X_{opt}$ from $\{X_1, X_2, \ldots, X_{CN}\}$
      **End do**
      **If** $\Delta f_{opt} < 0$ || $\rho < rand$ (see formula 10)
        $X_{current} = X_{opt}$
        $E_{current} = E_{current} + \Delta f_{opt}$
        Update the tabu list
      **End if**
    **End if**
    $it = it + 1$
  **End do**
  $t_{current} = t_{current} \times t_{cool}$
  $CN = \max(CN \times CN_{cool}, CN_{end})$
  $p = \max(p \times p_{cool}, p_{end})$
**End do**
Output $X_{best}$ and $E_{best}$.

should be adjusted according to a specific numerical example, and the acceptance probability only depends on the current temperature. However, in our algorithms, $t_{start}$ and $t_{end}$ are stipulated to be constants for convenience; thus we must adjust the acceptance probability $\rho$ according to different numerical examples.

$$4)\ \gamma = \frac{N \cdot std(|l|)}{\sum_{l \in K} |l|},\ K = \left\{ l \,||l| = \min_i(|l_{ij}|), |l_{ij} \in A; i, j \in V \right\} \quad (8)$$

where $std()$ calculates the standard deviation.

The scalar $\gamma$ can reflect the degree of irregularity and dispersion of a specific numerical example. The degree of irregularity and dispersion of the numerical example increases with the scalar $\gamma$ increasing. The degree of irregularity and dispersion has a significant influence on the following parameters: number of candidate solutions $CN$ (including $CN_{start}$ and $CN_{end}$), epoch length $Elen$, and acceptance probability $\rho$.

We apply benchmarks pr226 and ts225 from TSPLIB to demonstrate the relationship between the parameters ($CN$ and $Elen$) and the degree of irregularity and dispersion. Figs. 8 and 9 show distribution diagrams for pr226 and ts225, respectively. It is evident

**Table 4**
Experimental results of ts225 and pr226 under different settings of parameters (*CN* and *Elen*).

| Instances | $Error_{avg}$ | | | | |
|---|---|---|---|---|---|
| | *CN* = 53 *Elen* = 20746 | *CN* = 105 *Elen* = 10373 | *CN* = 210 *Elen* = 5187 | *CN* = 420 *Elen* = 2593 | *CN* = 840 *Elen* = 1297 |
| ts225 ($\gamma = 0$) | 0.416 | 0.357 | 0.233 | **0.124** | 0.126 |
| pr226 ($\gamma = 1.42$) | 0.747 | **0.415** | 1.259 | 1.214 | 2.280 |

**Table 5**
Setting of parameters of AHSATS-2-opt.

| notations | definition | value |
|---|---|---|
| $t_{start}$ | The initial temperature | 50 |
| $t_{cool}$ | The cooling coefficient of temperature | 0.99 |
| $t_{end}$ | The final temperature | 0.15 |
| *Elen* | Epoch length | Formula (9) |
| $\rho$ | The probability of accepting a newly generated solution | Formula (10) |
| *CN* | The number of candidate solutions | Formula (11) |
| *TL* | The length of tabu list | Formula (12) |

**Table 6**
Setting of parameters of AHSATS-D-CM.

| notations | Definition | value |
|---|---|---|
| $t_{start}$ | The initial temperature | 40 |
| $t_{cool}$ | The cooling coefficient of temperature | 0.99 |
| $t_{end}$ | The final temperature | 0.15 |
| *Elen* | Epoch length | Formula (13) |
| $\rho$ | The probability of accepting a newly generated solution | Formula (10) |
| *cooltime* | The number of decreases of *p* and *CN* | Formula (14) |
| *p* | The percentage of *PA* that accounts for *A* | – |
| $p_{start}$ | Initial value of percentage *p* | 1 |
| $p_{end}$ | Final value of percentage *p* | Formula (15) |
| $p_{cool}$ | Decrease coefficient of *p* | Formula (16) |
| *CN* | The number of candidate solutions | – |
| $CN_{start}$ | Initial number of candidate solutions | Formula (17) |
| $CN_{end}$ | Final number of candidate solutions | Formula (18) |
| $CN_{cool}$ | Decrease coefficient of *CN* | Formula (19) |
| *TL* | The length of tabu list | Formula (20) |

that pr226 has a high degree of irregularity and dispersion, whereas ts225 has a significantly lower degree of irregularity and dispersion. We employ AHSATS-2-opt to conduct an experiment showing the relationship between the numerical example's degree of irregularity and dispersion and the setting of parameters *CN* and *Elen*. We generate five pairs of *CN* and *Elen* parameters. Using each pair of parameters, we conduct six repeated experiments. The experimental results are listed in Table 4, where $error_{avg}$ (i.e. (mean length − optimum) × 100/optimum) denotes the percentage gap between the average tour length and the optimal tour length reported in TSPLIB.

The benchmarks ts225 and pr226 have almost the same scale, but the experiment shows that the two benchmarks have significant differences that result from the settings of parameters *CN* and *Elen*: pr226 produces the minimum average error when *Elen* = 10373 and *CN* = 105, while *Elen* = 2593 and *CN* = 420 are optimal for ts225 (the bold values shown in Table 4). The optimal parameters for the two benchmarks have a gap of four times. Thus, we can conclude that the degree of irregularity and dispersion can significantly influence the setting of parameters *CN* and *Elen*.

### 3.3.2. Parameters of AHSATS-2-opt

The initial temperature ($t_{start}$), final temperature ($t_{end}$), and cooling coefficient ($t_{cool}$) are all defined as constants for all numerical examples. In contrast, the SA epoch length (*Elen*), the number of candidate solutions (*CN*), the length of tabu list (*TL*), and the acceptance probability ($\rho$) of a newly generated solution ($X_{opt}$), which is generated by a tabu search, depend on the data of a specific numerical example. The parameter settings are listed in Table 5.

$$Elen = \left[\frac{5600N^{0.4}(1.27+\gamma^{4.11})[4.72\times10^{-11}(\gamma+0.1)+N^{-2.81}]}{(22.10+\gamma^{4.11})(1.42\times10^{-11}+N^{-2.81})}\right] \quad (9)$$

where the square brackets "[]" receive the rounding integer.

$$\rho = \begin{cases} \dfrac{-2.46\cdot N\cdot[f(X_{opt})-f(X_{current})]}{t_{current}\cdot\beta\cdot(3.7+\gamma^{1.1})}, f(X_{opt})-f(X_{current})>0 \\ 1\ , f(X_{opt})-f(X_{current})\le 0 \end{cases} \quad (10)$$

where $X_{opt}$ is newly generated by the tabu search, $X_{current}$ is the current solution, $t_{current}$ is the current temperature, and the function "$f()$" calculates the tour length.

In [21], the authors have already defined the parameter $\rho$ similarly:

$$\rho = e^{-[F(c^{best})-F(c^{old})/t_{current}]\times(10\times N/OPT)}.$$

However, the optimal tour length (*OPT*) is used in their formula. In typical instances, the optimal tour length is unknown, and thus their formula had limitations. In this paper, we modify the formula in Ref. [21], and employ the scalars $\beta$ and $\gamma$ instead, which can be easily calculated using the data from a specific numerical example.

$$CN = \left[\frac{2800N^{1.1}}{Elen}\right] \quad (11)$$

$$TL = \left[\frac{Elen^{0.6}}{3.5}\right] \quad (12)$$

### 3.3.3. Parameters of AHSATS-D-CM

In addition to the parameters defined in AHSATS-2-opt, extra parameters must be set for AHSATS-D-CM: the number of decreases for *p* and *CN* (*cooltime*), initial value of percentage *p* ($p_{start}$), final value of percentage *p* ($p_{end}$), decrease coefficient of *p* ($p_{cool}$), initial number of solution candidates ($CN_{start}$), final number of solution candidates ($CN_{end}$), and the decrease coefficient of *CN* ($CN_{cool}$). The parameter settings are listed in Table 6.

$$Elen = \left[\frac{5556N^{0.4}(1.28+\gamma^{4.11})[4.72\times10^{-11}(\gamma+0.1)+N^{-2.81}]}{(24.72+\gamma^{4.11})(1.42\times10^{-11}+N^{-2.81})}\right] \quad (13)$$

$$cooltime = \left[\frac{(121950+2.75N^{2.18})\cdot\log_{t_{cool}}^{t_{end}/t_{start}}}{121950+N^{2.18})}\right] \quad (14)$$

$$p_{end} = \min[\max(3\cdot\alpha, 0.1), 1] \quad (15)$$

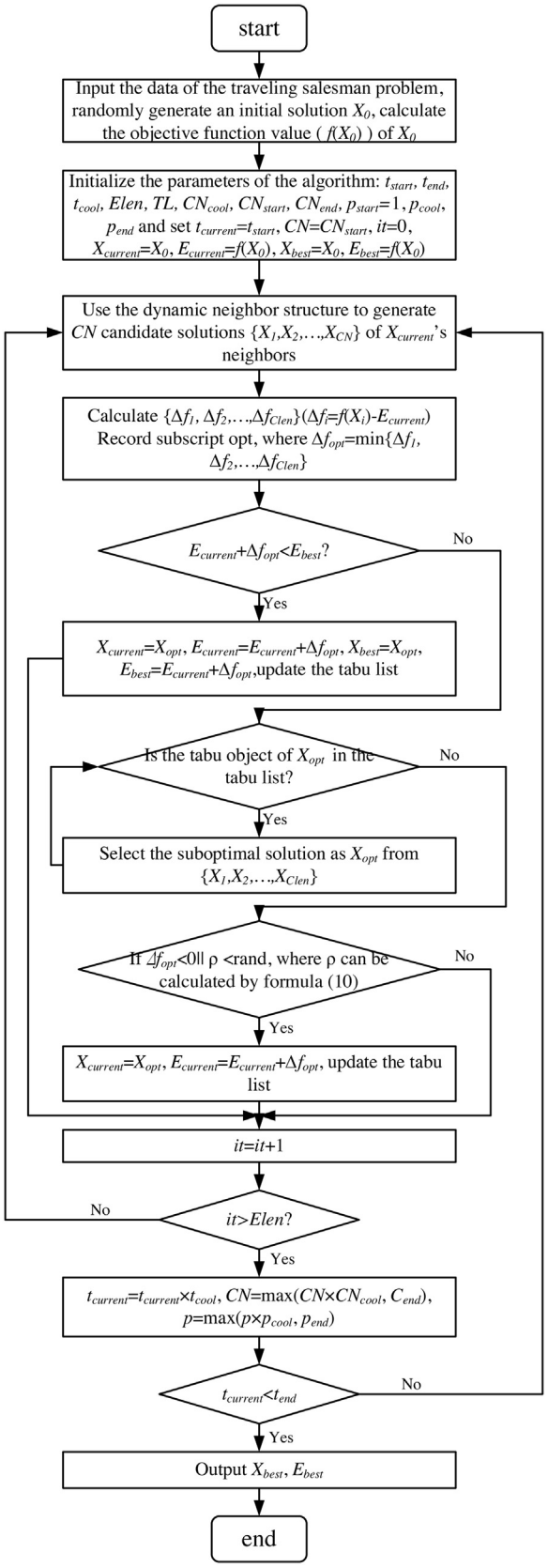


**Fig. 7.** Flow chart for AHSATS-D-CM.

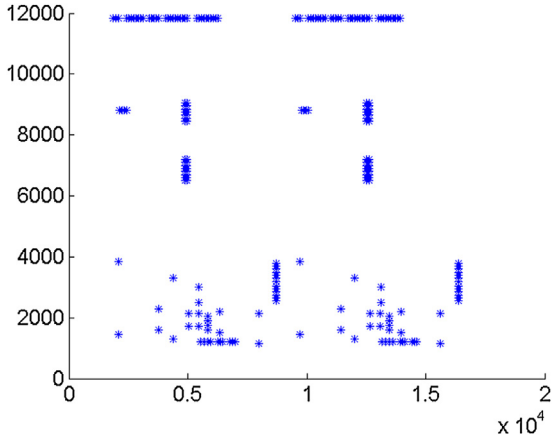


**Fig. 8.** Distribution diagram for pr226.

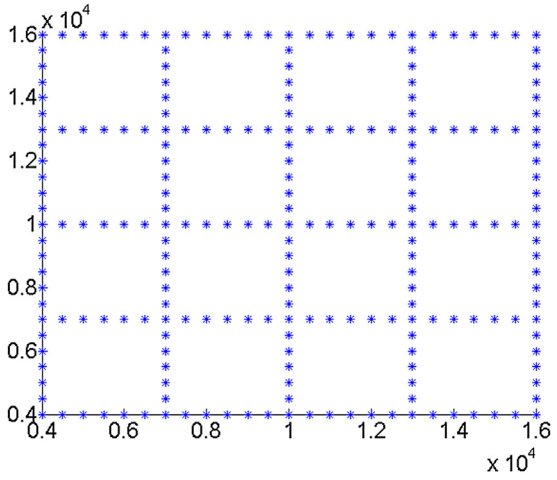


**Fig. 9.** Distribution diagram for ts225.

$p_{end}$ is defined in Formula (15) to ensure that every vertex (i.e. city) has a sufficient probability of being selected to generate neighbor solutions, so that the algorithm has global search capacity.

$$p_{cool} = p_{end}^{\frac{1}{cooltime}} \tag{16}$$

$$CN_{start} = \left\lceil \frac{2500N^{1.1}}{Elen} \right\rceil \tag{17}$$

$$CN_{end} = \left\lceil CN_{start}\ \frac{1538 + 1.35N^{1.14}}{1538 + N^{1.14}} + p_{end} - 1 \right\rceil \tag{18}$$

$$CN_{cool} = \left(\frac{CN_{start}}{CN_{end}}\right)^{\frac{1}{cootime}} \tag{19}$$

$$TL = \left\lceil \frac{Elen^{0.6}}{3.5} \right\rceil \tag{20}$$

### 3.3.4. Analysis of time complexity for AHSATS-2-opt and AHSATS-D-CM

We analyze the time complexity of the two algorithms AHSATS-2-opt and AHSATS-D-CM based on the adaptive parameters. Time complexities of the two algorithms depend on the epoch length (*Elen*), the number of candidate solutions (*CN*), and the length of tabu list (*TL*). The epoch length (*Elen*) determines number of iterations, *CN* determines the number of neighborhood solution generations and the number of calculations of objective function

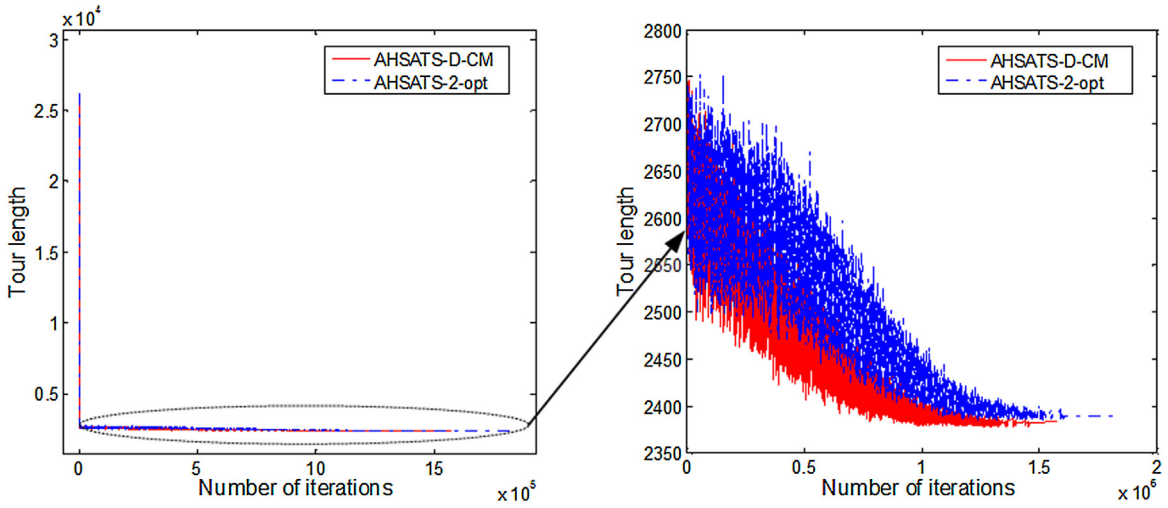


**Fig. 10.** Tour length changes resulting from increased iterations of AHSATS-2-opt and AHSATS-D-CM algorithms (gil262).

values, and *TL* determines the number of examinations of tabued solutions.

$$O(\text{AHSATS-2-opt}) = O(OLN \times Elen \times (CN + TL)) \tag{20}$$

where the *OLN* is the number of outside loops (number of decreases of temperature) of simulated annealing.

$$OLN = \frac{\ln t_{end} - \ln t_{start}}{\ln t_{cool}} \tag{21}$$

Since $t_{start}$, $t_{cool}$, and $t_{end}$ are constants, *OLN* is a constant.

$$\begin{aligned} &O(\text{AHSATS-2-opt}) = O(Elen \times (CN + TL)) \\ &= O(Elen \times CN + Elen \times TL) \\ &= O(N^{1.1} + N^{0.64}) \\ &= O(N^{1.1}) \end{aligned} \tag{22}$$

*Elen*, *CN*, and *TL* retain their previous definitions.

$$\begin{aligned} &O(\text{AHSATS-D-CM}) = O\left(Elen \times \sum_{k=1}^{OLN} \min\left(CN_{start} \times {CN_{cool}}^{k}, CN_{end}\right) + Elen \times TL\right) \\ &= O\left(CN_{start} \times Elen \times \sum_{k=1}^{OLN} \min\left({CN_{cool}}^{k}, \frac{1538 + 1.35N^{1.14}}{1538 + N^{1.14}} + p_{end} - 1\right) + Elen \times TL\right) \\ &= O(N^{1.1} + N^{0.64}) \\ &= O(N^{1.1}) \end{aligned} \tag{23}$$

The complexities of the two algorithms are only $O(N^{1.1})$ based on this set of adaptive parameters. The complexity of the algorithm is reasonable for a large scale TSP. Note that although the time complexities of the two algorithms are equal, AHSATS-D-CM has fewer iterations than AHSATS-2-opt.

## 4. Computational experiments

### 4.1. Numerical examples

Our algorithms, AHSATS-D-CM and AHSATS-2-opt, tested 64 benchmarks from TSPLIB, containing from 51 to 18,512 cities.

### 4.2. Running conditions

The benchmarks were tested with software implemented in C++ on a 3.10 GHz Windows 8 PC with 16GB RAM.

### 4.3. Setting of initial solutions

To investigate the influence of initial solutions on the quality of the solutions obtained by the two algorithms, we generated initial solutions using three different methods. The first is the nearest neighbor heuristic (NNH), in which the salesman starts from a city, then travels to the nearest city one by one until all cities are visited, and finally returns to the starting city. The second is random generation, which randomly generates the traversal order. The third, farthest heuristic (FH), is designed to maximize the tour length; the salesman starts from a city, then travels to the farthest city one by one until all cities are visited, and finally returns to the starting city. Each method generated two initial solutions, and each initial solution was optimized by AHSATS-2-opt and AHSATS-D-CM, for a total of six trials. In typical instances, the nearest neighbor heuristic obtains the shortest tour, random generation obtains a medium-length tour, and the farthest heuristic obtains the longest tour. If the initial solution has an influence on the execution of the two algorithms, the experimental results will show significant differences in the quality of the three groups of final solutions which are derived from the three groups of initial solutions generated by the three methods.

### 4.4. Computational results

The experimental results are listed in Table 7. Columns "OPT", "Best", "Avg", "$\text{Error}_{\text{best}}$", "$\text{Error}_{\text{avg}}$" and "T" respectively denote the optimal tour length reported in TSPLIB, the tour length of the best solution among the six trials using our proposed hybrid algorithms, average tour length of the six trials, percentage gap between the shortest tour length and the optimal tour length reported in TSPLIB, percentage gap between the average tour length and the optimal tour length, and the average CPU time for all six trials. Both algorithms can obtain satisfactory solutions within a reasonable computational time. AHSATS-D-CM (AHSATS-2-opt) can always find the optimal solutions reported in TSPLIB

**Table 7**
Running results of six trials for 64 benchmarks.

| Instances | OPT | AHSATS-D-CM | | | | | AHSATS-2-opt | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | best | avg | $Error_{best}$ | $Error_{avg}$ | T(s) | best | avg | $Error_{best}$ | $Error_{avg}$ | T(s) |
| eil51 | 426 | 426 | 426.00 | **0.000** | **0.000** | **2.729** | 426 | 426.00 | **0.000** | **0.000** | 4.901 |
| berlin52 | 7542 | 7542 | 7542.00 | **0.000** | **0.000** | **4.645** | 7542 | 7542.00 | **0.000** | **0.000** | 5.102 |
| ncit60 | 6400 | 6400 | 6400.00 | **0.000** | **0.000** | **2.850** | 6400 | 6400.00 | **0.000** | **0.000** | 4.770 |
| st70 | 675 | 675 | 675.00 | **0.000** | **0.000** | **4.101** | 675 | 675.00 | **0.000** | **0.000** | 6.592 |
| eil76 | 538 | 538 | 538.00 | **0.000** | **0.000** | **4.271** | 538 | 538.00 | **0.000** | **0.000** | 7.696 |
| pr76 | 108159 | 108159 | 108159.00 | **0.000** | **0.000** | **5.597** | 108159 | 108159.00 | **0.000** | **0.000** | 7.316 |
| rat99 | 1211 | 1211 | 1211.00 | **0.000** | **0.000** | **3.966** | 1211 | 1211.00 | **0.000** | **0.000** | 9.900 |
| rd100 | 7910 | 7910 | 7910.00 | **0.000** | **0.000** | **4.636** | 7910 | 7910.00 | **0.000** | **0.000** | 9.703 |
| kroA100 | 21282 | 21282 | 21282.00 | **0.000** | **0.000** | **4.740** | 21282 | 21284.33 | **0.000** | 0.011 | 10.001 |
| kroB100 | 22141 | 22141 | 22160.50 | **0.000** | **0.088** | **5.078** | 22141 | 22160.67 | **0.000** | 0.089 | 10.048 |
| kroC100 | 20749 | 20749 | 20749.00 | **0.000** | **0.000** | **4.403** | 20749 | 20749.33 | **0.000** | 0.002 | 10.328 |
| kroD100 | 21294 | 21294 | 21294.00 | **0.000** | **0.000** | **4.901** | 21294 | 21294.00 | **0.000** | **0.000** | 9.643 |
| kroE100 | 22068 | 22068 | 22113.17 | **0.000** | **0.205** | **5.270** | 22076 | 22125.50 | 0.036 | 0.261 | 10.082 |
| eil101 | 629 | 629 | 629.00 | **0.000** | **0.000** | **5.669** | 629 | 629.00 | **0.000** | **0.000** | 10.060 |
| lin105 | 14379 | 14379 | 14379.00 | **0.000** | **0.000** | **5.498** | 14379 | 14382.67 | **0.000** | 0.026 | 11.114 |
| pr107 | 44303 | 44303 | 44303.00 | **0.000** | **0.000** | **4.526** | 44303 | 44303.00 | **0.000** | **0.000** | 10.943 |
| pr124 | 59030 | 59030 | 59037.67 | **0.000** | 0.013 | **5.647** | 59030 | 59030.00 | **0.000** | **0.000** | 12.616 |
| bier127 | 118282 | 118282 | 118291.50 | **0.000** | **0.008** | **15.154** | 118282 | 118322.17 | **0.000** | 0.034 | 17.575 |
| ch130 | 6110 | 6110 | 6113.17 | **0.000** | **0.052** | **5.239** | 6110 | 6119.83 | **0.000** | 0.161 | 12.563 |
| pr136 | 96772 | 96785 | 96946.83 | **0.013** | **0.181** | **5.521** | 96920 | 96984.17 | 0.153 | 0.219 | 13.315 |
| pr144 | 58537 | 58537 | 58581.17 | **0.000** | 0.075 | **6.216** | 58537 | 58563.50 | **0.000** | **0.045** | 13.836 |
| ch150 | 6528 | 6528 | 6550.00 | **0.000** | **0.337** | **6.273** | 6549 | 6553.33 | 0.322 | 0.388 | 15.604 |
| kroA150 | 26524 | 26524 | 26536.00 | **0.000** | **0.045** | **6.260** | 26525 | 26548.50 | 0.004 | 0.092 | 15.902 |
| kroB150 | 26130 | 26132 | 26156.67 | **0.008** | **0.102** | **6.687** | 26135 | 26173.33 | 0.019 | 0.166 | 15.768 |
| pr152 | 73682 | 73682 | 73754.67 | **0.000** | **0.099** | **6.655** | 73682 | 73825.00 | **0.000** | 0.194 | 16.021 |
| u159 | 42080 | 42080 | 42238.33 | **0.000** | 0.376 | **7.656** | 42080 | 42186.83 | **0.000** | **0.254** | 15.681 |
| rat195 | 2323 | 2328 | 2328.50 | 0.215 | **0.237** | **7.979** | 2323 | 2330.00 | **0.000** | 0.301 | 20.278 |
| d198 | 15780 | 15791 | 15800.33 | **0.070** | **0.129** | 57.808 | 15798 | 15805.33 | 0.114 | 0.161 | **56.943** |
| kroA200 | 29368 | 29383 | 29498.00 | 0.051 | 0.443 | **8.520** | 29368 | 29466.33 | **0.000** | **0.335** | 20.655 |
| kroB200 | 29437 | 29437 | 29469.33 | **0.000** | **0.110** | **8.148** | 29445 | 29488.50 | 0.027 | 0.175 | 20.419 |
| ts225 | 126643 | 126643 | 126704.67 | **0.000** | **0.049** | **8.858** | 126643 | 126773.33 | **0.000** | 0.103 | 22.492 |
| pr226 | 80369 | 80369 | 80577.67 | **0.000** | **0.260** | **24.144** | 80467 | 80702.67 | 0.122 | 0.415 | 35.330 |
| gil262 | 2378 | 2378 | 2380.67 | **0.000** | **0.112** | **10.639** | 2379 | 2384.17 | 0.042 | 0.259 | 28.041 |
| pr264 | 49135 | 49135 | 49135.00 | **0.000** | **0.000** | **14.823** | 49135 | 49142.50 | **0.000** | 0.015 | 28.169 |
| pr299 | 48191 | 48191 | 48251.83 | **0.000** | **0.126** | **12.061** | 48191 | 48255.83 | **0.000** | 0.135 | 31.685 |
| lin318 | 42029 | 42175 | 42301.83 | 0.347 | **0.649** | **12.529** | 42148 | 42375.50 | **0.283** | 0.824 | 34.083 |
| rd400 | 15281 | 15294 | 15339.33 | **0.085** | **0.382** | **15.539** | 15308 | 15363.33 | 0.177 | 0.539 | 43.153 |
| fl417 | 11861 | 11867 | 11897.83 | **0.051** | **0.311** | **70.618** | 11872 | 11908.50 | 0.093 | 0.400 | 96.808 |
| pr439 | 107217 | 107393 | 108039.17 | **0.164** | **0.767** | **27.253** | 107552 | 108808.83 | 0.312 | 1.485 | 47.664 |
| pcb442 | 50778 | 50791 | 50979.33 | **0.026** | **0.396** | **19.573** | 50891 | 51060.00 | 0.223 | 0.555 | 48.789 |
| u574 | 36905 | 36966 | 37056.67 | **0.165** | **0.411** | **23.539** | 37062 | 37215.83 | 0.425 | 0.842 | 64.493 |
| rat575 | 6773 | 6791 | 6812.50 | **0.266** | **0.583** | **24.201** | 6807 | 6822.17 | 0.502 | 0.726 | 60.184 |
| u724 | 41910 | 42131 | 42173.17 | **0.527** | **0.628** | **32.272** | 42199 | 42261.33 | 0.690 | 0.838 | 82.079 |
| rat783 | 8806 | 8838 | 8865.33 | **0.363** | **0.674** | **35.613** | 8878 | 8900.00 | 0.818 | 1.067 | 86.248 |
| pr1002 | 259045 | 260540 | 261631.17 | **0.577** | **0.998** | **47.886** | 261538 | 262182.17 | 0.962 | 1.211 | 120.524 |
| pcb1173 | 56892 | 57092 | 57261.33 | **0.352** | **0.649** | **58.969** | 57400 | 57550.50 | 0.893 | 1.157 | 129.962 |
| d1291 | 50801 | 51187 | 51349.50 | **0.760** | **1.080** | 197.472 | 51238 | 51672.33 | 0.860 | 1.715 | **179.793** |
| rl1304 | 252948 | 254967 | 255948.50 | **0.798** | **1.186** | **64.243** | 255582 | 257798.00 | 1.041 | 1.917 | 151.194 |
| rl1323 | 270199 | 271092 | 272013.33 | **0.330** | **0.671** | **68.257** | 272881 | 273648.00 | 0.993 | 1.276 | 153.785 |
| fl1400 | 20127 | 20227 | 20481.33 | **0.497** | **1.760** | **527.347** | 20354 | 20570.00 | 1.128 | 2.201 | 616.704 |
| fl1577 | 22249 | 22279 | 22513.17 | 0.135 | **1.187** | **81.542** | 22270 | 22579.67 | **0.094** | 1.486 | 187.295 |
| d1655 | 62128 | 62823 | 62950.33 | **1.119** | **1.324** | 235.178 | 62968 | 63101.00 | 1.352 | 1.566 | **211.284** |
| vm1748 | 336556 | 339669 | 340478.50 | **0.925** | **1.165** | **99.985** | 339695 | 341282.67 | 0.933 | 1.404 | 204.661 |
| rl1889 | 316536 | 319798 | 322133.83 | **1.031** | **1.768** | **101.979** | 321533 | 322752.17 | 1.579 | 1.964 | 233.492 |
| d2103 | 80450 | 80603 | 81392.33 | **0.190** | **1.171** | 299.784 | 80920 | 81705.33 | 0.584 | 1.560 | **269.169** |
| u2319 | 234256 | 235274 | 235391.33 | **0.435** | **0.485** | **132.388** | 235749 | 235894.17 | 0.637 | 0.699 | 297.043 |
| pr2392 | 378032 | 382922 | 383621.67 | **1.294** | **1.479** | **133.704** | 383429 | 384559.17 | 1.428 | 1.727 | 285.931 |
| pcb3038 | 137694 | 138870 | 139185.67 | **0.854** | **1.083** | **168.122** | 139116 | 139329.33 | 1.033 | 1.188 | 366.544 |
| fnl4461 | 182566 | 184394 | 184625.00 | **1.001** | **1.128** | **275.655** | 185488 | 185626.50 | 1.601 | 1.676 | 551.400 |
| rl5934 | 556045 | 564590 | 566692.33 | **1.537** | **1.915** | **379.243** | 565346 | 567442.83 | 1.673 | 2.050 | 823.702 |
| pla7397 | 23260728 | 23745871 | 23935651.17 | **2.086** | **2.902** | **661.815** | 23979062 | 24028928.17 | 3.088 | 3.303 | 1109.000 |
| usa13509 | 19982859 | 20304083 | 20359728.67 | **1.607** | **1.886** | **1119.905** | 20389514 | 20432729.67 | 2.035 | 2.251 | 2188.455 |
| brd14051 | 469385 | 475279 | 478872.33 | **1.256** | **2.021** | **1230.269** | 480059 | 481100.50 | 2.274 | 2.496 | 2367.851 |
| d18512 | 645238 | 656621 | 659364.83 | **1.764** | **2.189** | 3328.670 | 660289 | 664009.33 | 2.333 | 2.909 | **2969.112** |
| Average | | | | 0.327 | 0.561 | | | | 0.483 | 0.732 | |

Bold numbers are the more excellent values obtained by the two algorithms.

for 15 (12) out of the 64 benchmarks, and optimal solutions for 30 (26) benchmarks can be found in six trials. The average $Error_{best}$ and average $Error_{avg}$ are 0.327 (0.483) and 0.561 (0.732) for AHSATS-D-CM (AHSATS-2-opt), respectively. From the comparison between AHSATS-D-CM and AHSATS-2-opt, we can see that AHSATS-D-CM is more accurate than AHSATS-2-opt. Although AHSATS-D-CM does not have a significant advantage over AHSATS-D-CM when the scale is relatively small, the superiority of AHSATS-D-CM over AHSATS-2-opt becomes more significant when the scale increases. Moreover, for most of the benchmarks,

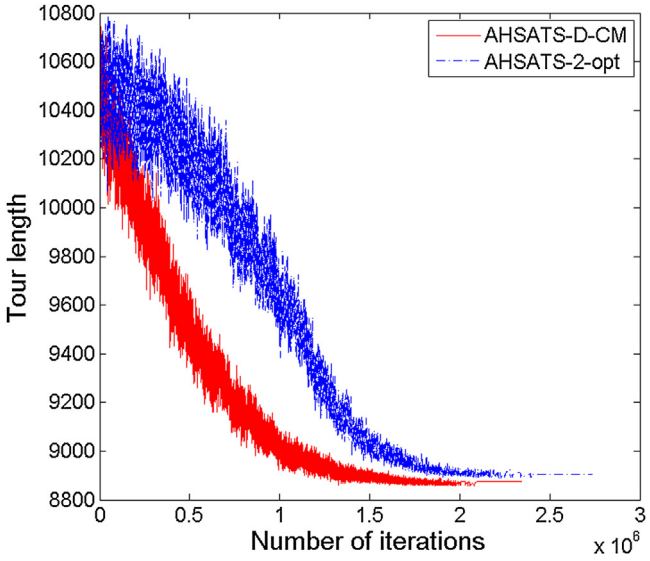


**Fig. 11.** Tour length changes resulting from increased iterations of AHSATS-2-opt and AHSATS-D-CM algorithms (rat783).

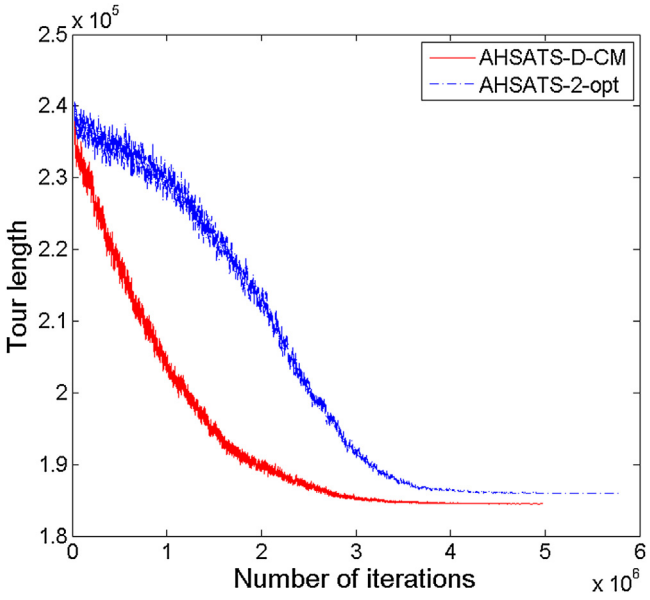


**Fig. 12.** Tour length changes resulting from increased iterations of AHSATS-2-opt and AHSATS-D-CM algorithms (fnl4461).

AHSATS-D-CM requires much less computation time compared to AHSATS-2-opt.

Figs. 10–12 show the changes in tour length for instances gil262, rat783, and fnl4461 that occur when the number of iterations increases for both algorithms. It is evident that both AHSATS-2-opt and AHSATS-D-CM exhibit a fast decrease rate and a clear convergence process. However, the number of iterations executed by AHSATS-D-CM is fewer than that of AHSATS-2-opt. AHSATS-D-CM's convergence rate is faster than that of AHSATS-2-opt.

Finally, in Table 8, the columns "NNH," "random," and "FH" respectively denote the three methods of generating initial solutions (nearest neighbor heuristic, random generation, and farthest heuristic). The rows "average" and "NM" denote the average gap and number of minimum gaps achieved based on the three groups of initial solutions. This table shows the relationship between the initial solutions and the results of the trials. For AHSATS-D-CM (AHSATS-2-opt), the average gaps are 0.519, 0.580, and 0.583 (0.765, 0.714, and 0.718) and the number of minimum gaps achieved are 35, 32, and 31 (30, 32, and 32), respectively, for the three groups of initial solutions. There is no significant difference in the results based on the three groups of initial solutions. Thus, we cannot conclude that the initial solutions' quality has an influence on the final solutions obtained by the two algorithms.

From the analysis above, we can draw the following conclusions:

1. The hybrid simulated annealing – tabu search algorithms have a fast decrease rate and a clear convergence process.
2. The quality of the initial solutions has almost no influence on the results of the hybrid algorithms.
3. The dynamic neighborhood structure is more accurate and efficient than the classical 2-opt because it can simultaneously reduce the number of candidate solutions that are generated, reduce the total number of iterations by improving the probability of acceptance, and ensure the quality of the solutions during the iterative process.
4. The adaptive parameters are appropriate for almost all numerical examples.

### *4.5. Comparisons of the proposed algorithm with improved traditional meta-heuristic algorithms in recent works*

In this subsection, we compare our algorithm AHSATS-D-CM with seven recent improved algorithms based on traditional meta-heuristic strategies—SA, TS, GA, ACO, and PSO. These include the Improved Ant Colony Optimization Algorithm (IACO) [36], Genetic Algorithm with Greedy Sub Tour Mutation (GA-GSTM) [29], Genetic Simulated Annealing Ant Colony System with Particle Swarm Optimization (GSAACPSO) [49], Adaptive Simulated Annealing Algorithm with Greedy Search (ASA-GS) [21], Hybrid Genetic Algorithm (HGA) [31], Ant Colony Extended (ACE) [37], and hybrid algorithm of Particle Swarm Optimization, Ant Colony Optimization and 3-Opt (PSO-ACO-3Opt) [75]. The results are listed in Table 9.

From the table, we can see that the quality of solutions obtained by our algorithm AHSATS-D-CM exceeds those reported in other studies for most tested benchmarks in terms of the best gap ($error_{best}$) and average gap ($error_{avg}$). AHSATS-D-C obtains no worse solutions than those obtained by IACO, GA-GSTM, GSAACPSO, and PSO-ACO-3Opt for all benchmarks tested. ASA-GS, HGA, and ACE found better solutions for only a few benchmarks. The proposed AHSATS-D-CM is obviously more accurate than the algorithms selected for comparison.

### *4.6. Comparisons of the proposed algorithm with other algorithms*

Table 10 lists the rankings of 59 algorithms from the DIMACS implementation challenge (http://www.research.att.com/~dsj/chtsp/). These include tour construction algorithms, simple local search algorithms, Lin-Kernighan implementations and variants, repeated local search algorithms, algorithms that use Chained Lin-Kernighan as a subroutine, and metaheuristics for solving 10 benchmarks (pr1002, pcb1173, d1291, rl1304, rl1323, fl1400, fl1577, rl1889, d2103, and pr2392). For a detailed explanation of the table contents, please refer to Ref. [65].

AHSATS-D-CM and AHSATS-2-opt are ranked in the 9th and 13th places, respectively, among the 59 algorithms. Both of the algorithms are ranked higher than all tour construction heuristics and all simple local search algorithms. Only a few Lin-Kernighan-based Variable Neighborhood Search implementations and one

**Table 8**
Experimental results of different initial solutions generated by three methods.

| Instances | AHSATS-D-CM | | | | | | AHSATS-2-opt | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NNH | | random | | FH | | NNH | | random | | FH | |
| | average | $error_{avg}$ | average | $error_{avg}$ | average | $error_{avg}$ | average | $error_{avg}$ | average | $error_{avg}$ | average | $error_{avg}$ |
| eil51 | 426.0 | **0.000** | 426.0 | **0.000** | 426.0 | **0.000** | 426.0 | **0.000** | 426.0 | **0.000** | 426.0 | **0.000** |
| berlin52 | 7542.0 | **0.000** | 7542.0 | **0.000** | 7542.0 | **0.000** | 7542.0 | **0.000** | 7542.0 | **0.000** | 7542.0 | **0.000** |
| ncit60 | 6400.0 | **0.000** | 6400.0 | **0.000** | 6400.0 | **0.000** | 6400.0 | **0.000** | 6400.0 | **0.000** | 6400.0 | **0.000** |
| st70 | 675.0 | **0.000** | 675.0 | **0.000** | 675.0 | **0.000** | 675.0 | **0.000** | 675.0 | **0.000** | 675.0 | **0.000** |
| eil76 | 538.0 | **0.000** | 538.0 | **0.000** | 538.0 | **0.000** | 538.0 | **0.000** | 538.0 | **0.000** | 538.0 | **0.000** |
| pr76 | 108159.0 | **0.000** | 108159.0 | **0.000** | 108159.0 | **0.000** | 108159.0 | **0.000** | 108159.0 | **0.000** | 108159.0 | **0.000** |
| rat99 | 1211.0 | **0.000** | 1211.0 | **0.000** | 1211.0 | **0.000** | 1211.0 | **0.000** | 1211.0 | **0.000** | 1211.0 | **0.000** |
| rd100 | 7910.0 | **0.000** | 7910.0 | **0.000** | 7910.0 | **0.000** | 7910.0 | **0.000** | 7910.0 | **0.000** | 7910.0 | **0.000** |
| kroA100 | 21282.0 | **0.000** | 21282.0 | **0.000** | 21282.0 | **0.000** | 21282.0 | **0.000** | 21282.0 | **0.000** | 21289.0 | 0.033 |
| kroB100 | 22141.0 | **0.000** | 22141.0 | **0.000** | 22199.5 | 0.264 | 22200.0 | 0.266 | 22141.0 | **0.000** | 22141.0 | **0.000** |
| kroC100 | 20749.0 | **0.000** | 20749.0 | **0.000** | 20749.0 | **0.000** | 20749.0 | **0.000** | 20750.0 | 0.005 | 20749.0 | **0.000** |
| kroD100 | 21294.0 | **0.000** | 21294.0 | **0.000** | 21294.0 | **0.000** | 21294.0 | **0.000** | 21294.0 | 0.000 | 21294.0 | **0.000** |
| kroE100 | 22114.5 | 0.211 | 22094.5 | **0.120** | 22130.5 | 0.283 | 22147.5 | 0.360 | 22110.5 | **0.193** | 22118.5 | 0.229 |
| eil101 | 629.0 | **0.000** | 629.0 | **0.000** | 629.0 | **0.000** | 629.0 | **0.000** | 629.0 | **0.000** | 629.0 | **0.000** |
| lin105 | 14379.0 | **0.000** | 14379.0 | **0.000** | 14379.0 | **0.000** | 14379.0 | **0.000** | 14379.0 | **0.000** | 14390.0 | 0.077 |
| pr107 | 44303.0 | **0.000** | 44303.0 | **0.000** | 44303.0 | **0.000** | 44303.0 | **0.000** | 44303.0 | **0.000** | 44303.0 | **0.000** |
| pr124 | 59030.0 | **0.000** | 59053.0 | 0.039 | 59030.0 | **0.000** | 59030.0 | **0.000** | 59030.0 | **0.000** | 59030.0 | **0.000** |
| bier127 | 118295.0 | 0.011 | 118297.5 | 0.013 | 118282.0 | **0.000** | 118297.5 | **0.013** | 118316.5 | 0.029 | 118352.5 | 0.060 |
| ch130 | 6115.0 | 0.082 | 6112.0 | **0.033** | 6112.5 | 0.041 | 6125.5 | 0.254 | 6121.5 | 0.188 | 6112.5 | **0.041** |
| pr136 | 96919.5 | 0.152 | 97020.0 | 0.256 | 96901.0 | **0.133** | 96920.0 | **0.153** | 97067.5 | 0.305 | 96965.0 | 0.199 |
| pr144 | 58563.5 | **0.045** | 58590.0 | 0.091 | 58590.0 | 0.091 | 58563.5 | **0.045** | 58563.5 | **0.045** | 58563.5 | **0.045** |
| ch150 | 6552.5 | 0.375 | 6553.5 | 0.391 | 6544.0 | **0.245** | 6552.5 | 0.375 | 6556.0 | 0.429 | 6551.5 | **0.360** |
| kroA150 | 26530.5 | 0.025 | 26524.5 | **0.002** | 26553.0 | 0.109 | 26568.0 | 0.166 | 26525.0 | **0.004** | 26552.5 | 0.107 |
| kroB150 | 26144.5 | **0.055** | 26174.5 | 0.170 | 26151.0 | 0.080 | 26177.5 | 0.182 | 26183.5 | 0.205 | 26159.0 | **0.111** |
| pr152 | 73750.0 | 0.092 | 73682.0 | **0.000** | 73832.0 | 0.204 | 73767.5 | **0.116** | 73827.5 | 0.197 | 73880.0 | 0.269 |
| u159 | 42239.0 | 0.378 | 42238.0 | **0.375** | 42238.0 | **0.375** | 42238.0 | 0.375 | 42242.5 | 0.386 | 42080.0 | **0.000** |
| rat195 | 2328.0 | **0.215** | 2328.0 | **0.215** | 2329.5 | 0.280 | 2332.5 | 0.409 | 2331.5 | 0.366 | 2326.0 | **0.129** |
| d198 | 15801.0 | 0.133 | 15798.5 | **0.117** | 15801.5 | 0.136 | 15811.5 | 0.200 | 15802.0 | **0.139** | 15802.5 | 0.143 |
| kroA200 | 29456.5 | 0.301 | 29616.0 | 0.844 | 29421.5 | **0.182** | 29429.0 | **0.208** | 29525.5 | 0.536 | 29444.5 | 0.260 |
| kroB200 | 29482.0 | 0.153 | 29461.5 | **0.083** | 29464.5 | 0.093 | 29469.0 | **0.109** | 29517.0 | 0.272 | 29479.5 | 0.144 |
| ts225 | 126713.0 | 0.055 | 126700.5 | **0.045** | 126700.5 | **0.045** | 126719.5 | 0.060 | 126713.0 | **0.055** | 126887.5 | 0.193 |
| pr226 | 80496.5 | 0.159 | 80816.5 | 0.557 | 80420.0 | **0.063** | 80753.0 | 0.478 | 80673.5 | **0.379** | 80681.5 | 0.389 |
| gil262 | 2378.5 | **0.021** | 2380.5 | 0.105 | 2383.0 | 0.210 | 2379.0 | **0.042** | 2383.0 | 0.210 | 2390.5 | 0.526 |
| pr264 | 49135.0 | **0.000** | 49135.0 | **0.000** | 49135.0 | 0.000 | 49135.0 | **0.000** | 49135.0 | **0.000** | 49157.5 | 0.046 |
| pr299 | 48248.0 | 0.118 | 48275.5 | 0.175 | 48232.0 | **0.085** | 48250.0 | 0.122 | 48309.0 | 0.245 | 48208.5 | **0.036** |
| lin318 | 42267.5 | **0.567** | 42282.5 | 0.603 | 42355.5 | 0.777 | 42376.0 | 0.826 | 42407.0 | 0.899 | 42343.5 | **0.748** |
| rd400 | 15297.0 | **0.105** | 15363.5 | 0.540 | 15357.5 | 0.501 | 15340.5 | **0.389** | 15357.0 | 0.497 | 15392.5 | 0.730 |
| fl417 | 11901.5 | 0.341 | 11922.5 | 0.519 | 11869.5 | **0.072** | 11904.0 | 0.363 | 11923.5 | 0.527 | 11898.0 | **0.312** |
| pr439 | 107992.0 | 0.723 | 108138.0 | 0.859 | 107987.5 | **0.719** | 109208.5 | 1.857 | 107606.5 | **0.363** | 109611.5 | 2.233 |
| pcb442 | 50949.5 | **0.338** | 51028.0 | 0.492 | 50960.5 | 0.359 | 51109.0 | 0.652 | 51044.0 | 0.524 | 51027.0 | **0.490** |
| u574 | 37017.5 | **0.305** | 37072.5 | 0.454 | 37080.0 | 0.474 | 37257.5 | 0.955 | 37150.5 | **0.665** | 37239.5 | 0.906 |
| rat575 | 6797.0 | **0.354** | 6819.0 | 0.679 | 6821.5 | 0.716 | 6820.0 | **0.694** | 6823.5 | 0.746 | 6823.0 | 0.738 |
| u724 | 42164.0 | 0.606 | 42160.0 | **0.597** | 42195.5 | 0.681 | 42286.5 | 0.898 | 42278.0 | 0.878 | 42219.5 | **0.738** |
| rat783 | 8863.0 | 0.647 | 8875.0 | 0.784 | 8858.0 | **0.591** | 8887.5 | **0.926** | 8906.0 | 1.136 | 8906.5 | 1.141 |
| pr1002 | 261325.0 | **0.880** | 262018.0 | 1.148 | 261550.5 | 0.967 | 262134.0 | 1.192 | 262185.0 | **1.212** | 262227.5 | 1.229 |
| pcb1173 | 57351.5 | 0.808 | 57234.0 | 0.601 | 57198.5 | **0.539** | 57536.0 | 1.132 | 57650.0 | 1.332 | 57465.5 | **1.008** |
| d1291 | 51380.5 | 1.141 | 51456.0 | 1.289 | 51212.0 | **0.809** | 51480.0 | **1.337** | 51807.0 | 1.980 | 51730.0 | 1.829 |
| rl1304 | 255280.0 | **0.922** | 256353.0 | 1.346 | 256212.5 | 1.291 | 259480.5 | 2.583 | 256737.5 | **1.498** | 257176.0 | 1.671 |
| rl1323 | 271651.0 | **0.537** | 272202.0 | 0.741 | 272187.0 | 0.736 | 274361.5 | 1.541 | 273196.5 | **1.109** | 273386.0 | 1.180 |
| fl1400 | 20443.5 | 1.573 | 20436.5 | **1.538** | 20564.0 | 2.171 | 20722.5 | 2.959 | 20544.5 | 2.074 | 20443.0 | **1.570** |
| fl1577 | 22468.0 | 0.984 | 22465.5 | **0.973** | 22606.0 | 1.605 | 22701.0 | 2.032 | 22450.5 | **0.906** | 22587.5 | 1.521 |
| d1655 | 63059.0 | 1.499 | 62843.0 | **1.151** | 62949.0 | 1.321 | 63162.0 | 1.664 | 62985.0 | **1.379** | 63156.0 | 1.655 |
| vm1748 | 341426.5 | 1.447 | 340045.0 | 1.037 | 339964.0 | **1.013** | 340150.5 | **1.068** | 342950.0 | 1.900 | 340747.5 | 1.245 |
| rl1889 | 320867.0 | **1.368** | 321351.0 | 1.521 | 324183.5 | 2.416 | 323007.0 | 2.044 | 322098.0 | **1.757** | 323151.5 | 2.090 |
| d2103 | 81105.5 | **0.815** | 81162.5 | 0.886 | 81909.0 | 1.814 | 81631.5 | 1.469 | 81973.5 | 1.894 | 81511.0 | **1.319** |
| u2319 | 235470.5 | 0.518 | 235306.5 | **0.448** | 235397.0 | 0.487 | 235962.5 | 0.728 | 235807.0 | **0.662** | 235913.0 | 0.707 |
| pr2392 | 384038.5 | 1.589 | 383133.5 | **1.349** | 383693.0 | 1.497 | 384458.0 | **1.700** | 384671.5 | 1.756 | 384548.0 | 1.724 |
| pcb3038 | 139182.0 | 1.081 | 139323.0 | 1.183 | 139052.0 | **0.986** | 139310.0 | 1.174 | 139334.0 | **1.191** | 139344.0 | 1.198 |
| fnl4461 | 184836.0 | 1.243 | 184494.5 | **1.056** | 184544.5 | 1.084 | 185660.0 | 1.695 | 185720.5 | 1.728 | 185499.0 | **1.607** |
| rl5934 | 566100.5 | **1.808** | 567279.5 | 2.020 | 566697.0 | 1.916 | 569663.5 | 2.449 | 566453.5 | 1.872 | 566211.5 | **1.828** |
| pla7397 | 23794414.0 | **2.294** | 24068214.5 | 3.471 | 23944325.0 | 2.939 | 24046177.5 | 3.377 | 23993299.5 | **3.149** | 24047307.5 | 3.382 |
| usa13509 | 20334728.0 | **1.761** | 20377160.0 | 1.973 | 20367298.0 | 1.924 | 20419600.5 | 2.186 | 20467951.5 | 2.428 | 20410637.0 | **2.141** |
| brd14051 | 479844.5 | 2.228 | 478973.0 | 2.043 | 477799.5 | **1.793** | 481400.5 | 2.560 | 481321.0 | 2.543 | 480580.0 | **2.385** |
| d18512 | 659263.5 | **2.174** | 659464.5 | 2.205 | 659366.5 | 2.190 | 661781.5 | **2.564** | 664105.0 | 2.924 | 666141.5 | 3.240 |
| Average | | 0.519 | | 0.580 | | 0.583 | | 0.765 | | 0.714 | | 0.718 |
| NM | | 35 | | 32 | | 31 | | 30 | | 32 | | 32 |

Bold numbers are the minimum average errors using the three methods "NNH", "random" and "FH" for each of the two hybrid algorithms, AHSATS-D-CM and AHSATS-2-opt.

metaheuristic algorithm (HBMO) are ranked higher than our algorithms, because our algorithms only use one mutation method while LK and HBMO incorporate various mutation structures. There are two other metaheuristic algorithms (GA-EAC [30] and PHGA [24]) that should be ranked higher than our algorithms. They are not in Table 10 because they have not been tested against the ten

**Table 9**
Comparison with meta-heuristics reported in literature.

| Instances | $Error_{best}$(%) | | | | | | | | $Error_{avg}$(%) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IACO | GA-GSTM | GSAACPSO | ASA-GS | HGA | ACE | PSO-ACO-3Opt | AHSATS-D-CM | IACO | GA-GSTM | GSAACPSO | ASA-GS | HGA | ACE | PSO-ACO-3Opt | AHSATS-D-CM |
| eil51 | **0.000** | | 0.235 | 0.674 | 0.674 | **0.000** | **0.000** | **0.000** | 0.763 | | 0.298 | 0.674 | 0.749 | 0.192 | 0.106 | **0.000** |
| berlin52 | **0.000** | **0.000** | **0.000** | 0.031 | 0.031 | **0.000** | **0.000** | **0.000** | 0.438 | **0.000** | **0.000** | 0.031 | 0.031 | 0.014 | 0.016 | **0.000** |
| ncit64 | | | **0.000** | | | | | **0.000** | | | **0.000** | | | | | **0.000** |
| st70 | **0.000** | | | 0.313 | 0.313 | **0.000** | 0.148 | **0.000** | 1.311 | | | 0.313 | 0.354 | 0.210 | 0.474 | **0.000** |
| eil76 | **0.000** | | **0.000** | 1.184 | 1.184 | **0.000** | **0.000** | **0.000** | 1.199 | | 0.409 | 1.184 | 1.498 | 0.058 | 0.056 | **0.000** |
| pr76 | | | | **0.000** | **0.000** | **0.000** | | **0.000** | | | | **0.000** | 0.090 | 0.085 | | **0.000** |
| rat99 | | | | 0.680 | 0.680 | **0.000** | **0.000** | **0.000** | | | | 0.701 | 0.904 | 0.189 | 0.278 | **0.000** |
| rd100 | | | **0.000** | 0.005 | 0.005 | | | **0.000** | | | 0.981 | 0.005 | 0.044 | | | **0.000** |
| kroA100 | **0.000** | **0.000** | **0.000** | 0.016 | 0.016 | **0.000** | 0.089 | **0.000** | 0.664 | 1.184 | 0.416 | 0.016 | 0.143 | 0.078 | 0.766 | **0.000** |
| kroB100 | | | **0.000** | – | | | | **0.000** | | | 0.641 | – | | | | **0.088** |
| kroC100 | | | **0.000** | 0.009 | 0.008 | | | **0.000** | | | 0.626 | 0.009 | 0.305 | | | **0.000** |
| kroD100 | | | 0.070 | 0.001 | 0.001 | | | **0.000** | | | 1.533 | 0.033 | 0.238 | | | **0.000** |
| kroE100 | | | **0.000** | 0.174 | | | | **0.000** | | | 0.523 | **0.201** | | | | 0.205 |
| eil101 | | | 0.159 | 1.783 | 1.782 | **0.000** | 0.318 | **0.000** | | | 0.990 | 1.831 | 2.515 | 0.734 | 0.588 | **0.000** |
| lin105 | **0.000** | | **0.000** | 0.028 | 0.028 | **0.000** | **0.000** | **0.000** | 0.651 | | 0.190 | 0.028 | 0.305 | 0.045 | 0.001 | **0.000** |
| pr107 | | | | – | **0.000** | | | **0.000** | | | | – | 0.087 | | | **0.000** |
| pr124 | | | | 0.001 | 0.001 | | | **0.000** | | | | **0.001** | 0.109 | | | 0.013 |
| bier127 | | | **0.000** | 0.010 | | | | **0.000** | | | 0.964 | 0.057 | | | | **0.008** |
| ch130 | | | 0.507 | 0.012 | 0.012 | **0.000** | | **0.000** | | | 1.565 | 0.182 | 0.332 | 0.719 | | **0.052** |
| pr136 | | | | 0.201 | 0.014 | | | **0.013** | | | | 0.317 | 0.256 | | | **0.181** |
| pr144 | **0.000** | **0.000** | | – | **0.000** | | | **0.000** | 0.246 | 1.081 | | 0.015 | **0.000** | | | 0.075 |
| ch150 | | 0.460 | **0.000** | 0.044 | 0.044 | **0.000** | 0.153 | **0.000** | | 0.636 | 0.547 | **0.181** | 0.455 | 0.337 | 0.551 | 0.337 |
| kroA150 | | | **0.000** | 0.003 | 0.003 | | | **0.000** | | | 1.415 | 0.055 | 0.278 | | | **0.045** |
| kroB150 | | 0.964 | **0.000** | 0.041 | **0.000** | | | 0.008 | | 1.762 | 1.218 | 0.184 | **0.788** | | | 0.102 |
| pr152 | | 0.770 | | 0.002 | 0.002 | **0.000** | | **0.000** | | 1.620 | | **0.017** | 0.114 | 0.115 | | 0.099 |
| u159 | | | | 0.744 | | **0.000** | | **0.000** | | | | 0.758 | | **0.285** | | 0.376 |
| rat195 | | 0.603 | | 0.957 | 1.043 | | | **0.215** | | 1.843 | | 1.078 | 1.421 | | | **0.237** |
| d198 | | 0.387 | | 0.321 | 0.741 | **0.000** | | 0.070 | | 1.219 | | 0.414 | 1.161 | 0.211 | | **0.129** |
| kroA200 | | 0.868 | 0.051 | 0.148 | **0.005** | | 0.341 | 0.051 | | 1.543 | 1.262 | **0.240** | 0.309 | | 0.947 | 0.443 |
| kroB200 | | | 0.353 | 0.228 | 0.046 | | | **0.000** | | | 2.032 | 0.259 | 0.497 | | | **0.110** |
| ts225 | | 0.253 | | 0.002 | 1.184 | | | **0.000** | | 0.499 | | **0.002** | 1.305 | | | 0.049 |
| pr226 | | 0.724 | | 0.215 | 0.083 | | | **0.000** | | 1.529 | | 0.396 | **0.206** | | | 0.260 |
| gil262 | | | | 0.658 | | **0.000** | | **0.000** | | | | 0.867 | | 0.507 | | **0.112** |
| pr264 | | | | **0.000** | 0.033 | | | **0.000** | | | | 0.008 | 0.058 | | | **0.000** |
| pr299 | | 1.233 | | 0.162 | 2.638 | | | **0.000** | | 1.233 | | 0.281 | 3.251 | | | **0.126** |
| lin318 | | 0.983 | 1.090 | 0.661 | 1.416 | | | **0.347** | | 3.310 | 2.317 | 0.844 | 2.018 | | | **0.649** |
| rd400 | | | | 0.456 | 5.030 | | | **0.085** | | | | 0.974 | 5.647 | | | **0.382** |
| fl417 | | | | 0.669 | | | | **0.051** | | | | 1.541 | | | | **0.311** |
| pr439 | | | | 2.614 | 2.755 | | | **0.164** | | | | 2.806 | 3.724 | | | **0.767** |
| pcb442 | | 2.050 | | 0.563 | 4.128 | | | **0.026** | | 2.776 | | 0.967 | 4.408 | | | **0.396** |
| u574 | | | | 0.887 | | | | **0.165** | | | | 1.259 | | | | **0.411** |
| rat575 | | | 1.742 | 1.463 | | | | **0.266** | | | 2.375 | 1.946 | | | | **0.583** |
| u724 | | | | 0.870 | | | | **0.527** | | | | 1.337 | | | | **0.628** |
| rat783 | | | 2.067 | 1.685 | | 0.704 | | **0.363** | | | 3.103 | 2.001 | 1.484 | | | **0.674** |
| pr1002 | | | | 1.724 | | | | **0.577** | | | | 2.019 | | | | **0.998** |
| pcb1173 | | | | 1.527 | | | | **0.352** | | | | 1.632 | | | | **0.649** |
| d1291 | | | | 1.870 | | | | **0.760** | | | | 2.857 | | | | **1.080** |
| rl1323 | | | 2.755 | 0.653 | | | | **0.330** | | | 3.694 | 1.201 | | | | **0.671** |
| fl1400 | | | 2.315 | 2.586 | | 0.999 | | **0.497** | | | 6.075 | 3.255 | 1.811 | | | **1.760** |
| d1655 | | | 3.256 | 2.427 | | | | **1.119** | | | 5.622 | 3.264 | | | | **1.324** |
| vm1748 | | | | 1.747 | | | | **0.925** | | | | 2.185 | | | | **1.165** |
| u2319 | | | | 0.896 | | | | **0.435** | | | | 1.062 | | | | **0.485** |
| pcb3038 | | | | 2.214 | | | | **0.854** | | | | 2.577 | | | | **1.083** |
| fnl4461 | | | | 2.405 | | | | **1.001** | | | | 2.653 | | | | **1.128** |
| rl5934 | | | | 2.938 | | | | **1.537** | | | | 3.487 | | | | **1.915** |
| pla7397 | | | | 3.616 | | | | **2.086** | | | | 3.894 | | | | **2.902** |
| usa13509 | | | | 4.087 | | | | **1.607** | | | | 4.145 | | | | **1.886** |
| brd14051 | | | | 3.459 | | | | **1.256** | | | | 3.702 | | | | **2.021** |
| d18512 | | | | 3.544 | | | | **1.764** | | | | 3.752 | | | | **2.189** |

Bold numbers represent the optimal values among those obtained by all different algorithms in terms of average error and best error.

benchmarks. Note that a fair comparison in terms of computational efficiency is difficult, because computational speed is affected by the compiler and the hardware that are used. Therefore, the comparisons against the algorithms from the literature were performed only terms of solution quality

## 5. Conclusion

This paper develops an adaptive hybrid meta-heuristic algorithm that combines simulated annealing and tabu search algorithms with a dynamic neighborhood structure to solve the Traveling Salesman Problem (TSP). The experimental results demonstrated that the proposed algorithm can obtain satisfactory solutions within a reasonable time. Moreover, the proposed algorithm can overcome the individual disadvantages of simulated annealing and tabu search. The new hybrid algorithm provides a fast decrease rate and a clear convergence process, and is independent on initial solutions. Furthermore, compared with the classical 2-opt neighborhood, the dynamic neighborhood can significantly reduce computational time by decreasing the number of calculations and simultaneously improve solution quality. Finally, the

**Table 10**
Ranking of the algorithms based on the average quality (%).

| Ranks. | Algorithms | Error(%) | Ranks | Algorithms | Error (%) |
|---|---|---|---|---|---|
| 1 | HBMO [43] | 0 | 31 | 3opt-B [68] | 3.635 |
| 2 | Tour Merging [15] | 0.004 | 32 | Concorde-Lin-Kernighan [15] | 3.916 |
| 3 | ILK-NYYY [71] | 0.009 | 33 | 4-Hyperopt [12] | 4.062 |
| 4 | Iterated Lin – Kernighan by Johnson [12] | 0.041 | 34 | Applegate Lin Kernighan [15] | 4.308 |
| 5 | Keld Helsgaun's Multi-Trial Variant on Lin-Kernighan [14] | 0.064 | 35 | 3-Hyperopt [12] | 4.559 |
| 6 | Iterated-3-Opt by Johnson [12] | 0.413 | 36 | GENIUS [70] | 4.685 |
| 7 | Augmented-LK [71] | 0.42 | 37 | Tabu-search-2opt-2opt [22] | 4.83 |
| 8 | Applegate-Cook-Rohe Chained Lin–Kernighan [15] | 0.567 | 38 | Memetic SOM [74] | 5.775 |
| 9 | **AHSATS-D-CM** | 0.596 | 39 | 2.5opt-B [68] | 5.93 |
| 10 | Iterated Lin–Kernighan by Neto [15] | 0.642 | 40 | 2opt-J [12] | 6.113 |
| 11 | Multi-Level Lin-Kernighan Implementations [73] | 0.743 | 41 | 2-Hyperopt [12] | 6.218 |
| 12 | Concorde Chain Lin-Kernighan [15] | 0.88 | 42 | Held Karp Christofides [71] | 6.496 |
| 13 | **AHSATS-2-opt** | 0.956 | 43 | 2opt-B [68] | 6.83 |
| 14 | Helsgaun Lin-Kernighan [14] | 0.969 | 44 | GENI [70] | 8.05 |
| 15 | Tabu-search-SC-DB [22] | 1.034 | 45 | CCA [71] | 10.43 |
| 16 | Expanding Neighborhood Search GRASP [65] | 1.181 | 46 | Clarke–Wright[69] | 11.26 |
| 17 | Lin-Kernighan-with-HK-Christo-starts [71] | 1.202 | 47 | 2opt-C [15] | 14.56 |
| 18 | Tabu-search-LK-DB [22] | 1.241 | 48 | FI [68] | 16.64 |
| 19 | Variable-Neighborhood-Search-Using-3-Hyperopt [71] | 1.292 | 49 | RI [5] | 17.47 |
| 20 | Balas-Simonetti-Dynamic-Programming-Heuristic [71] | 1.308 | 50 | Boruvka [15] | 17.76 |
| 21 | Johnson Lin–Kernighan [12] | 1.41 | 51 | CHCI [68] | 17.96 |
| 22 | ASA-GS [21] | 1.672 | 52 | C-Greedy [15] | 18.23 |
| 23 | Stem–Cycle Ejection Chain [71] | 1.702 | 53 | B-Greedy [71] | 18.708 |
| 24 | LK-NYYY [71] | 1.706 | 54 | Q-Boruvka [15] | 19.6 |
| 25 | Neto Lin–Kernighan [72] | 1.804 | 55 | NN [15] | 24.85 |
| 26 | Variable-Neighborhood-Search-Using-2-Hyperopt [71] | 1.993 | 56 | Spacefilling [71] | 47.545 |
| 27 | Tabu-search-Stem − Cycle −SC [22] | 2.034 | 57 | Best–Way Strip [71] | 49.405 |
| 28 | Tabu-search-LK-LK [22] | 2.2 | 58 | FRP [68] | 74.36 |
| 29 | Tabu-search-2opt-Double Bridge [22] | 3.11 | 59 | Strip [71] | 75.565 |
| 30 | 3opt-J [12] | 3.5 | | | |

adaptive parameters are appropriate for solving almost all benchmark test examples.

## 6. Future research

There are several areas for future research. First, we anticipate that the dynamic neighborhood structure will be adaptable to other evolutionary algorithms as a local search strategy or a mutation operator. These evolutionary algorithms may include genetic algorithms, ant colony optimization, particle swarm optimization, and honey bee mating optimization. Furthermore, the idea of the dynamic neighborhood is available for other neighborhoods, such as 3-opt, 4-opt, and even variable neighborhoods. In other future works, we plan to apply the dynamic neighborhood concept to other types of local search problems. Finally, our algorithms can be processed in parallel; therefore we will develop parallel algorithms based on the proposed algorithms in order to solve numerical examples with larger scales.

## References


[1] H. Papadimitriou Christos, The Euclidean travelling salesman problem is NP-complete, Theor. Comput. Sci. 4 (3) (1977) 237–244.
[2] Osman Ibrahim Hassan, Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem, Ann. Oper. Res. 41 (4) (1993) 421–451.
[3] Ketan Savla, Emilio Frazzoli, Francesco Bullo, Traveling salesperson problems for the Dubins vehicle, IEEE Trans. Autom. Control 53 (6) (2008) 1378–1391.
[4] Yannis Marinakis, Magdalene Marinaki, Georgios Dounias, A hybrid particle swarm optimization algorithm for the vehicle routing problem, Eng. Appl. Artif. Intell. 23 (4) (2010) 463–472.
[5] Marinakis Yannis, Multiple phase neighborhood search-GRASP for the capacitated vehicle routing problem, Expert Syst. Appl. 39 (8) (2012) 6807–6815.
[6] Qu Yuan, F. Bard Jonathan, A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment, Comput. Oper. Res. 39 (10) (2012) 2439–2456.
[7] Mladenović Nenad, Dragan Urošević, Aleksandar Ilić, A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem, Eur. J. Oper. Res. 220 (1) (2012) 270–285.
[8] Keng Hoo Chuah, Jon C. Yingling, Routing for a just-in-time supply pickup and delivery system, Transp. Sci. 39 (3) (2005) 328–339.
[9] Rami Musa, Jean-Paul Arnaout, Hosang Jung, Ant colony optimization algorithm to solve for the transportation problem of cross-docking network, Comput. Ind. Eng. 59 (2) (2010) 85–92.
[10] S. Meysam Mousavi, Reza Tavakkoli-Moghaddam, A hybrid simulated annealing algorithm for location and routing scheduling problems with cross-docking in the supply chain, J. Manuf. Syst. 32 (2) (2013) 335–347.
[11] Zhu Xiaowei, et al., A complexity model for sequence planning in mixed-model assembly lines, J. Manuf. Syst. 31 (2) (2012) 121–130.
[12] David A. Johnson, Lyle A. McGeoch, The traveling salesman problem: a case study in local optimization, Local search in combinatorial optimization 1 (1997) 215–310.
[13] Shen Lin, W. Kernighan Brian, An effective heuristic algorithm for the traveling-salesman problem, Oper. Res. 21 (2) (1973) 498–516.
[14] Helsgaun Keld, An effective implementation of the Lin–Kernighan traveling salesman heuristic, Eur. J. Oper. Res. 126 (1) (2000) 106–130.
[15] Applegate David, William Cook, André Rohe, Chained Lin-Kernighan for large traveling salesman problems, INFORMS J. Comput. 15 (1) (2003) 82–92.
[16] Helsgaun Keld, General k-opt submoves for the Lin–Kernighan TSP heuristic, Math. Program. Comput. 1 (2–3) (2009) 119–163.

[17] Daniel Karapetyan, Gregory Gutin, Lin–Kernighan heuristic adaptations for the generalized traveling salesman problem, Eur. J. Oper. Res. 208 (3) (2011) 221–232.
[18] Reinelt Gerhard, The Traveling Salesman: Computational Solutions for TSP Applications, Springer-Verlag, 1994, 2016.
[19] Yong Chen, Pan Zhang, Optimized annealing of traveling salesman problem from the nth-nearest-neighbor distribution, Phys. A: Stat. Mech. Appl. 371 (2) (2006) 627–632.
[20] Zicheng Wang, Xiutang Geng, Zehui Shao, An effective simulated annealing algorithm for solving the traveling salesman problem, J. Comput. Theor. Nanosci. 6 (7) (2009) 1680–1686.
[21] Geng Xiutang, et al., Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search, Appl. Soft Comput. 11 (4) (2011) 3680–3689.
[22] Martin Zachariasen, Martin Dam, Tabu search on the geometric traveling salesman problem, Meta-Heuristics. Springer, US, 1996, pp. 571–587.
[23] Knox John, Tabu search performance on the symmetric traveling salesman problem, Comput. Oper. Res. 21 (8) (1994) 867–876.
[24] Hung Nguyen Dinh, et al., Implementation of an effective hybrid GA for large-scale traveling salesman problems, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 37 (1) (2007) 92–99.
[25] Huai-Kuang Tsai, et al., An evolutionary algorithm for large traveling salesman problems, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 34 (4) (2004) 1718–1729.
[26] Sushil J. Louis, Gong Li, Case injected genetic algorithms for traveling salesman problems, Inf. Sci. 122 (2) (2000) 201–225.
[27] Marinakis Yannis, Athanasios Migdalas, M. Panos Pardalos, A hybrid genetic—GRASP algorithm using lagrangean relaxation for the traveling salesman problem, J. Comb. Optim. 10 (4) (2005) 311–326.
[28] Baraglia Ranieri, Jose Ignacio Hidalgo, Raffaele Perego, A hybrid heuristic for the traveling salesman problem, IEEE Trans. Evol. Comput. 5 (6) (2001) 613–622.
[29] Albayrak Murat, Novruz Allahverdi, Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms, Expert Syst. Appl. 38 (3) (2011) 1313–1320.
[30] Yuichi Nagata, Shigenobu Kobayashi, A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem, Inf. J. Comput. 25 (2) (2013) 346–363.
[31] Wang, Yong The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem, Comput. Ind. Eng. 70 (2014) 124–133.
[32] Dorigo Marco, Luca Maria Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.
[33] Amr Badr, Ahmed Fahmy, A proof of convergence for Ant algorithms, Inf. Sci. 160 (1) (2004) 267–279.
[34] Shu-Chuan Chu, John F. Roddick, Jeng-Shyang Pan, Ant colony system with communication strategies, Information Sciences 167 (1) (2004) 63–76.
[35] Anzuo Liu, Guishi Deng, Shimin Shan, Mean-Contribution Ant System: an Improved Version of Ant Colony Optimization for Traveling Salesman Problem. Simulated Evolution and Learning, Springer, Berlin Heidelberg, 2006, pp. 489–496.
[36] Zar Chi Su Su Hlaing, May Aye Khine, Solving traveling salesman problem by using improved ant colony optimization algorithm, Int. J. Inf. Educ. Technol. 1 (5) (2011) 404.
[37] B. Escario Jose, Juan F. Jimenez, Jose M. Giron-Sierra, Ant colony extended: experiments on the travelling salesman problem, Expert Syst. Appl. 42 (1) (2015) 390–410.
[38] Xiangyong Li, et al., A Hybrid Discrete Particle Swarm Optimization for the Traveling Salesman Problem Simulated Evolution and Learning, Springer, Berlin Heidelberg, 2006, pp. 181–188.
[39] H. Shi Xiaohu, et al., Particle swarm optimization-based algorithms for TSP and generalized TSP, Inf. Process. Lett. 103 (5) (2007) 169–176.
[40] Yan Wang, et al., A novel quantum swarm evolutionary algorithm and its applications, Neurocomputing 70 (4) (2007) 633–640.
[41] Yu-xi Gao, Yan-min Wang, Zhi-li Pei, An improved particle swarm optimisation for solving generalised travelling salesman problem, Int. J. Comput. Sci. Math. 3 (4) (2012) 385–393.
[42] Yannis Marinakis, Magdalene Marinaki, A hybrid honey bees mating optimization algorithm for the probabilistic traveling salesman problem, in: Evolutionary Computation, 2009, CEC'09. IEEE Congress on. IEEE, 2009.
[43] Yannis Marinakis, Magdalene Marinaki, Georgios Dounias, Honey bees mating optimization algorithm for the Euclidean traveling salesman problem, Inf. Sci. 181 (20) (2011) 4684–4698.
[44] Yanping Bai, Wendong Zhang, Zhen Jin, An new self-organizing maps strategy for solving the traveling salesman problem, Chaos Solitons Fractals 28 (4) (2006) 1082–1089.
[45] A.S. Masutti Thiago, N. Leandro de Castro, A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem, Inf. Sci. 179 (10) (2009) 1454–1468.
[46] Abdolhamid Modares, Samerkae Somhom, Takao Enkawa, A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems, Int. Trans. Oper. Res. 6 (6) (1999) 591–606.
[47] Siqueira Paulo Henrique, Maria Teresinha Arns Steiner, Sérgio Scheer, A new approach to solve the traveling salesman problem, Neurocomputing 70 (4) (2007) 1013–1021.
[48] Cheng-Fa Tsai, Chun-Wei Tsai, Ching-Chang Tseng, A new hybrid heuristic approach for solving large traveling salesman problem, Inf. Sci. 166 (1) (2004) 67–81.
[49] Shyi-Ming Chen, Chih-Yao Chien, Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques, Expert Syst. Appl. 38 (12) (2011) 14439–14450.
[50] Shyi-Ming Chen, Chih-Yao Chien, Parallelized genetic ant colony systems for solving the traveling salesman problem, Expert Syst. Appl. 38 (4) (2011) 3873–3883.
[51] Gaifang Dong, William W. Guo, Kevin Tickle, Solving the traveling salesman problem using cooperative genetic ant systems, Expert Syst. Appl. 39 (5) (2012) 5006–5011.
[52] Mauro Dell'Amico, Marco Trubian, Applying tabu search to the job-shop scheduling problem, Ann. Oper. Res. 41 (3) (1993) 231–252.
[53] Kolonko Michael, Some new results on simulated annealing applied to the job shop scheduling problem, Eur. J. Oper. Res. 113 (1) (1999) 123–136.
[54] Chao Yong Zhang, et al., A very fast TS/SA algorithm for the job shop scheduling problem, Comput. Oper. Res. 35 (1) (2008) 282–294.
[55] Patrick R. McMullen, Gregory V. Frazier, A simulated annealing approach to mixed-model sequencing with multiple objectives on a just-in-time line, IIE Trans. 32 (8) (2000) 679–686.
[56] Patrick R. McMullen, G.V. Frazier, Using simulated annealing to solve a multiobjective assembly line balancing problem with parallel workstations, Int. J. Prod. Res. 36 (10) (1998) 2717–2741.
[57] Patrick R. McMullen, JIT sequencing for mixed-model assembly lines with setups using tabu search, Prod. Plann. Control 9 (5) (1998) 504–510.
[58] Marvin A. Arostegui, N. Sukran Kadipasaoglu, M. Basheer Khumawala, An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems, Int. J. Prod. Econ. 103 (2) (2006) 742–754.
[59] Scott Kirkpatrick, Mario P. Vecchi, Optimization by simmulated annealing, Science 220 (4598) (1983) 671–680.
[60] Černý Vladimír, Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm, J. Optim. Theory Appl. 45 (1) (1985) 41–51.
[61] Malek Miroslaw, et al., Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem, Ann. Oper. Res. 21 (1) (1989) 59–84.
[62] Joshua W. Pepper, Bruce L. Golden, A. Wasil Edward, Solving the traveling salesman problem with annealing-based heuristics: a computational study, IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. 32 (1) (2002) 72–77.
[63] Glover Fred, Tabu search-part I, ORSA J. Comput. 1 (3) (1989) 190–206.
[64] Glover Fred, Tabu search—part II, ORSA J. Comput. 2 (1) (1990) 4–32.
[65] Yannis Marinakis, Athanasios Migdalas, Panos M. Pardalos, Expanding neighborhood GRASP for the traveling salesman problem, Comput. Optim. Appl. 32 (3) (2005) 231–257.
[66] Metropolis Nicholas, et al., Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (6) (1953) 1087–1092.
[67] David S. Johnson, Lyle A. McGeoch, The traveling salesman problem: a case study in local optimization, Local search in combinatorial optimization 1 (1997) 215–310.
[68] Bentley Jon Jouis, Fast algorithms for geometric traveling salesman problems, ORSA J. Comput. 4 (4) (1992) 387–411.
[69] G.U. Clarke, John W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Oper. Res. 12 (4) (1964) 568–581.
[70] Michel Gendreau, Alain Hertz, Gilbert Laporte, New insertion and postoptimization procedures for the traveling salesman problem, Oper. Res. 40 (6) (1992) 1086–1094.
[71] S. Johnson David, A. Lyle McGeoch, Experimental Analysis of Heuristics for the STSP. The Traveling Salesman Problem and Its Variations, Springer, US, 2007, pp. 369–443.
[72] M. Neto David, Efficient Cluster Compensation for Lin-kernighan Heuristics, Diss University of Toronto, 1999.
[73] Walshaw Chris, A multilevel approach to the travelling salesman problem, Oper. Res. 50 (5) (2002) 862–877.
[74] Jean-Charles Créput, Abderrafiãa Koukam, A memetic neural network for the Euclidean traveling salesman problem, Neurocomputing 72 (4) (2009) 1250–1264.
[75] Mahi Mostafa, Ömer Kaan Baykan, Halife Kodaz, A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem, Appl. Soft Comput. 30 (2015) 484–490.
[76] K. Ahuja Ravindra, L. Thomas Magnanti, B. James Orlin, Network Flows. No. MIT-WP-2059-88, ALFRED P SLOAN SCHOOL OF MANAGEMENT, CAMBRIDGE MA, 1988.
[77] Mesut Gündüz, Mustafa Servet Kiran, Eren Özceylan, A hierarchic approach based on swarm intelligence to solve the traveling salesman problem, Turk. J. Electr. Eng. Comput. Sci. 23 (1) (2015) 103–117.
[78] Yong Wang, Hybrid Max?Min ant system with four vertices and three lines inequality for traveling salesman problem, Soft Comput. 19 (3) (2015) 585–596.
[79] Munlin Mud-Armeen, Mana Anantathanavit, Hybrid K-means and particle swarm optimization for symmetric traveling salesman problem, in: IEEE 10th Conference on Industrial Electronics and Applications (ICIEA), IEEE, 2015, p. 2015.