Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

A hybrid genetic-simulated annealing algorithm for multiple traveling salesman problems

F. Smaili[a*]

[a]*Mechanical Engineering Department, College of Engineering, University of Bisha, Bisha 61922, Saudi Arabia*

C H R O N I C L E

Article history:
Received: October 23, 2023
Received in revised format: March 2, 2024
Accepted: April 4, 2024
Available online: April 4, 2024

Keywords:
MTSP
Genetic algorithm
Simulated annealing
Hybrid algorithm
Non-dominated front
Statistical Analyses

A B S T R A C T

The Multiple Traveling Salesman Problem (MTSP) was able to model and solve various theoretical and real-life applications. This problem is one of the many difficult issues that have no perfect solution yet. In this paper, on the one hand genetic algorithms with different combinations of operators and simulated annealing were used to solve the MTSP. On the other hand, the genetic algorithm with the combination of operators that gave the best solutions of the MTSP was hybridized with a Simulated Annealing algorithm. The simulation results showed that the hybrid algorithm significantly outperforms most of the comparable methods in obtaining the best-fitness solutions compared to the other methods in most of the test cases. In addition, by scaling the fitness function according to the amplitude of tours, it was obvious that the non-dominated front obtained by the hybrid algorithm was better than the non-dominated front obtained by the other algorithms.

1. Introduction

Travelling Salesman Problem (TSP) is a well-studied problem in combinatorial optimization (Shmoys et al., 1985).There exists no polynomial order algorithm for TSPs, which are NP-hard (Shmoys et al., 1985; Lin & Kernighan, 1973). Thus, it is possible that the worst-case running time of any algorithm for the TSP increases exponentially with the number of cities (Lin & Kernighan, 1973). Knowing the cities and the distance between them, the goal of the TSP problem is to minimize the distance of a path traveled by a salesman who visits this set of cities exactly once, starting and ending in the same city. Application of TSP is found in many areas such as logistics and transportation, semiconductor manufacturing, design of hardware devices and radio electronic systems and computer networks (Filip & Otakar, 2011; Laporte, 1992).

The Multiple Travelling Salesman Problem (MTSP) is an extension of the famous TSP where each city is visited by exactly one of m salesmen (Reinelt, 2003). A salesman starts and ends at the depot (which is one of the n cities). Minimizing the total distance of all paths taken by the salesmen and the difference between the longest and shortest sub-tour are the goals of the MTSP. Many real-life problems can be modeled as the MTSP, such as path planning (Yu et al., 2002), hot rolling scheduling problem (Tang et al., 2000), distribution of emergence materials problem (Ming et al., 2014), UAVs planning problem (Ann et al., 2015). Most existing literature on the MTSPs focuses on a single-objective, or considers the objectives from two separate perspectives: for example, minimizing the total distance travelled and minimizing the longest travel distance of a salesman. The second objective is a condition for balancing the sub tours, the workload among salesmen and service time of each customer in practical situations. However, reducing only the total distance will result in highly imbalanced sub-tours where one salesman serves most cities whereas each of the other salesmen serves one of the closer cities to the depot. If we only consider the balance between sub-tours, this will unnecessarily increase the total travel distances. Therefore, optimizing the total distance and the balance between sub tours are two conflicting goals which cannot be considered separately. In the first part of this paper, the total distance of the salesman is taken as an objective function.

* Corresponding author.
E-mail address: fsmaili@ub.edu.sa (F. Smaili)

doi: 10.5267/dsl.2024.4.001

In the second part, the difference between the longest route and the shortest route is used as an objective function to determine a better non-dominated front.

The MTSP has been tackled using many different approaches; most of them are nature-inspired algorithms such as Genetic Algorithms (GA), Ant Colony Optimization (ACO) and Gravitational Emulation Local Search (GELS). In addition, nearest-neighbor based search algorithms such as 2-Opt, were frequently used to improve the quality of solutions obtained by other algorithms. These methods proved to be efficient and provide reasonable solutions when implemented on their own. To the best of our knowledge, no approach combining the above-mentioned algorithms has been applied to solve the MTSP. However, there are some disadvantages in using traditional GA to solve MTSP, such as poor search-ability and low convergence accuracy. At the same time, the hybrid algorithm has also received the attention of many experts and scholars. On this basis, we combined GA and SA and proposed an improved genetic simulated annealing algorithm for solving MTSP, improving the local search capabilities and the convergence of the GA.

The remainder of this paper is organized as follows: Section 2 illustrates some basic definitions and concepts, such as distance matrix and MTSP modeling. Section 3 presents the description of the different used algorithms. In Section 4, some experiments and a discussion are performed to show the effectiveness of algorithms and models. Finally, some basic concluding remarks are discussed in Section 5.

2. Related Work

To solve MTSP, all practical algorithms can be classified into exact algorithms and heuristics. The choice of the algorithm depends on the size of the problem, so to get an optimal solution for small-scale problems, exact algorithms are suitable. For large-scale problems and due to the NP-hard nature of the MTSP, heuristic algorithms are more popularly employed. For problems with less than 100 cities, the cutting-planes algorithm was used to optimally solve MTSP Laporte & Nobert, 1980). Various mutation and crossover operators for the MTSPs for the GA were designed and compared elsewhere (Li et al., 2013). The results showed that proper operator design could boost the performance. An estimation of distribution algorithm (EDA) with a gradient search was used to solve the MTSP in which an objective function was set as the weighted sum of the total travelling costs of all salesmen and the highest travelling cost of any single salesman (Shim et al., 2012). In the latter reference, the authors considered minimizing the longest cost to balance the workload between salesmen. The same authors designed an insert, swap, and two-operator's algorithm for enhancing the capability to escape from local optimum points (Yousefikhoshbakht et al., 2013). In another method, five local search schemes were introduced to improve the result without much increased time complexity (Soylu, 2015). An effective evolutionary algorithm, reinforced by a post-optimization procedure based on path-relinking (PR), is used to deal with a bi-objective multiple travelling salesman problems with profits (Labadie et al., 2014). Meanwhile, other evolutionary algorithms based on swarm intelligence are used to solve the MTSP problem. A new acceleration particle swarm optimization was constructed to solve the MTSP (Qiang & Kang, 2014), which can effectively overcome the premature convergence. Several multi-objective ACSs are proposed to tackle MTSP from a bi-criteria perspective that require minimizing the total cost of travelled sub-tours while achieving balanced sub-tours (Necula et al., 2015).

Moreover, the two-phase heuristic algorithm (TPHA) which combined K-means and modified GA was used for solving MTSP subject to the workload balance (Xu et al., 2018). Based on a genetic algorithm, two new local operators Branch and Bound and cross-elimination were effectively combined to find high-quality solutions within a short time for study of MTSP (Lo et al., 2018). Another MTSP study in which a hybrid algorithm developed, integrating ACO, 2-Opt based GA (AC2OptGA) and showed results that are outperforming other state-of-art techniques (Harrath et al., 2019). In addition, suggesting an evolutionary NSGA-II algorithm, which effectively jumps from the local optimum, has been addressed to a bi-objective MTSP model (Shuai et al., 2019). A comparative study of various GA crossover operators for MTSP can be found in (Al-Omeer & Ahmed, 2019). Inspired by the existing works mentioned above, the work done in this paper further improves the convergence of using GA to solve the MTSP with a hybrid of simulating operators.

3. Modeling of the MTSP

Based on the number of depots, the MTSP problem can be divided into single-depot multiple TSP (SD-MTSP) and multi-depot multiple TSP (MD-MTSP), the former means that the salesmen start from the same starting depot and the latter means that the salesmen start from different starting depot. This paper mainly explores the two objectives SD-MTSP, where the two objectives are conflicting. Usually, the problem can be described as follows: there is a set of n number of cities and m number of salesmen, the set is expressed as $D = \{0, 1, 2, \ldots N\}$ and $V = \{1, 2, \ldots, m\}$. Each salesman departs from the same depot, takes a tour route and returns to the original starting city. Here we use c_{ij} for the distance between cities i and j, and x_{ijk} for salesman k from cities i to j. The distance between cities is represented by matrix, known as distance matrix $D = (D_{ij})$, i, j = 1, 2,…, n, Furthermore, we can get the distance matrix C as follows:

$$D = \begin{bmatrix} D_{11} & \dots & D_{1i} & \dots & D_{1j} & \dots & D_{1(N+1)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ D_{i1} & \dots & D_{ii} & \dots & D_{ij} & \dots & D_{i(N+1)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ D_{j1} & \dots & D_{ij} & \dots & C_{jj} & \dots & D_{j(N+1)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ D_{(N+1)1} & \dots & D_{(N+1)i} & \dots & D_{(N+1)j} & \dots & D_{(N+1)(N+1)} \end{bmatrix}$$

Usually, the diagonal elements of the distance matrix D are zeros, c_{ij}, represents the distance of travel from i^{th} to j^{th} city. The distance D is said to be symmetric when $D_{ij} = D_{ji}$, ∀ (i, j) ∈ E and asymmetric otherwise. Each city will be visited exactly once (except the starting point). Ideally, the total travel distance is minimized while the travel distance between salesmen is as close as possible. The mathematical model is as follows:

$$\min \quad F = (f_1, f_2) \tag{1}$$

$$f_1 = \sum_{k=1}^{m} \sum_{i=0}^{n} \sum_{j=0}^{n} C_{ij} X_{ijk} \tag{2}$$

$$f_2 = \max_{1 \le k \le m} \sum_{i=0}^{n} \sum_{j=0}^{n} C_{ij} X_{ijk} - \min_{1 \le k \le m} \sum_{i=0}^{n} \sum_{j=0}^{n} C_{ij} X_{ijk} \tag{3}$$

where

$$X_{ijk} = \begin{cases} 1, & salesman\ k\ passes\ from\ city\ i\ to\ city\ j \\ 0, & else \end{cases} \tag{4}$$

subject to

$$\begin{cases} \sum_{k=1}^{m} \sum_{i=0}^{n} X_{ijk} = 1; & \forall j = 1, \dots, n \\ \sum_{k=1}^{m} \sum_{j=1}^{n} X_{ijk} = 1; & \forall i = 1, \dots, n \\ \sum_{k=1}^{m} \sum_{i=1}^{n} X_{i0k} = m; & \\ \sum_{k=1}^{m} \sum_{j=1}^{n} X_{0jk} = m; & \\ \sum_{i \in S} \sum_{j \in S} X_{ijk} \ge 1; & \forall k \in V, \forall S \sqsubseteq \complement. \end{cases} \tag{5}$$

Eq. (2) and Eq. (3) respectively represent two objective functions: the total distance of the salesman and the difference between the longest route and the shortest route. What we need to do is to minimize both and . Eq. (5) represents the constraint: all salesmen start from the same starting city 1. It is required that, except for the starting city, there is one and only one salesman in each city passing through, and all salesmen return to their starting city. The final solution doesn't generate sub tours.

4. Proposed algorithm

On the one hand, Genetic Algorithm for different methods of selection operator, crossover operator and mutation operator and simulated annealing is crucial for some heuristics algorithms to solve the NP-complete problems such as MTSP. On the other hand, the combination of some algorithm strategies is more advantageous than one algorithm strategy. In addition, different algorithm strategies usually take different effects for the same problem. As a result, and in the first part, double methods of crossover and mutation strategies such as single point crossover, two-point crossover, random swap mutation and reverse swap mutation are used with different combinations in solving MTSP. In the second part, the simulated annealing algorithm is also used in solving MTSP. Finally, in the third part, we propose a hybrid Genetic-Simulated Annealing Algorithm.

4.1. Genetic Algorithm

A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution from about 50 years ago (Bremermann et al., 1965). This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. Their application to combinatorial optimization problems has only recently become an actual research topic. In recent years, many articles and books on evolutionary optimization of NP-hard problems have been published, in very different application domains such as computer aided design, cryptanalysis, identification of systems, medicine, microelectronics, pattern recognition, production planning, robotics, telecommunications (Bodenhofer, 2004). In 1975 Holland (1992) introduced genetic algorithms with a problem search space represented as a collection of individuals. Representing character strings, these individuals are often called chromosomes. The goal of using a genetic algorithm is to find the individual from the search space with the best "genetic material". The part of the search space to be examined is called the population and the quality of an individual is measured with an evaluation function (fitness function).

First, the initial population is chosen and the quality of this population is determined. Then, at each iteration, the parents are selected from the population. These parents produce children, which are added to the population. For all the newly created individuals of the resulting population, there is a probability close to zero that they will "mutate". In order to bring the population back to its initial size, certain individuals are removed from the population according to a selection criterion. Then an iteration of the algorithm is called generation. The crossover operator and the mutation operator are operators that define the child production process and the mutation process respectively. Crossover and mutation perform different parts in the genetic algorithm. The crossover should increase the average quality of the population. The mutation is necessary to explore new states and helps the algorithm to avoid local optima.

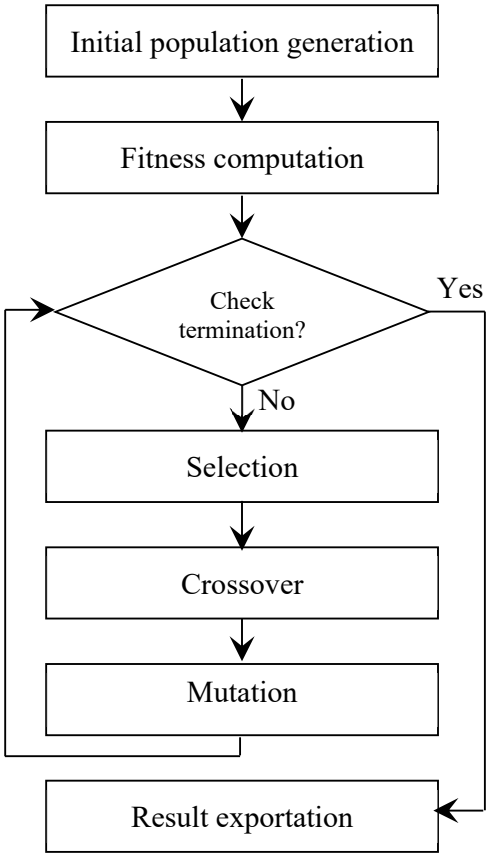

Fig. 1. A flowchart of the genetic algorithm

By choosing an appropriate crossover and mutation operators, the probability that the genetic algorithm results in an almost optimal solution in a reasonable number of iterations is increased. There can be various criteria for stopping algorithm. For example, if it is possible to define previously the number of iterations needed. But the stopping criteria should normally take into account the relationship between the average objective function versus the objective function of the best individual, the uniformity of the population, as well as not producing an increase in the objective function of the best individual during a fixed number of cycles. A large amount of research on the use of GA to resolve TSP or MTSP has been carried out successfully with satisfactory results.For the MTSP, the fitness value equals the total travel distance of all salesmen. Therefore, the optimal solution is found when the fitness value is minimized. The whole process of the proposed algorithm is illustrated in Fig. 1. Termination criteria can be defined as a fixed number of generations, a fixed execution time, or an

end time if no significant improvement can be made over previous generations. The steps of the genetic algorithm are presented in Algorithm 1.

4.1.1. Chromosome Representation

A several ways were used to encode an MTSP solution into a chromosome, containing in the same chromosome (Tang et al., 2000), two chromosomes (Malmborg, 1996), two-part chromosome (Carter & Ragsdale, 2006), Multi-Chromosome Technique (Singh et al., 2018) etc. With the abundance of solution space, experimental results have demonstrated that the two-part chromosome performs the best in terms of quality of solution and convergence speed. The two-part chromosome technique reduces redundant solutions (Albayrak & Allahverdi, 2011; Brown et al., 2007; Carter & Ragsdale, 2006). However, in this technique for the MTSP, regarding the first part of the chromosome there are n! possible permutations. The second part of the chromosome represents a positive vector of integers (k_1, k_2,…, k_m), such that the sum of their components must be equal to n. Therefore, the two-part chromosome representation is adopted in this work. In this method, MTSP solution is represented by two parts of a chromosome of length $n + m$, where n is the number of cities and m is the number of salesmen used in MTSP. The first part of the chromosome of length n represents a permutation (tour), of n cities, in which each gene takes integer value ranging from 1 to n. Second part of the chromosome of length m gives the number of cities assigned to each salesman. In this example, the first salesman will visit 2 cities as indicated in the second part of the chromosome.

Since all salesmen must depart from and return to the depot (V_0), then this city is outside the chromosome to minimize memory space. Fig. 2 shows a chromosome representing the MTSP solution (where $n = 10$ and $m = 3$), in which n cities are represented by natural numbers between 0 and $n-1$, and 0 represents the central city. Among them, the chromosome consists of two parts: the first part is the arrangement of $n-1$ natural numbers; the second part is m−1 break points, which divides the first part into m groups, each group is represented by a salesman. As well as Fig. 3 shows the details of calculation of total distance, thus the total distance of n cites is divided over three salesmen, the result of the division indicates the position i of the end of the part of route of salesman number 1, the position i of the end of the part of route of the second salesman is indicated by two thirds of the total distance and so on.

In order to more intuitively display the route of each salesman represented by the chromosome code, we assume that the relative positions of the 10 cities are as shown in the left of Fig. 4, and the solution corresponding to the chromosome in Fig. 3 is as shown in the right of Fig. 4.

Algorithm 1

Pseudocode of Genetic Algorithm

```
1.  PS;  // Population Size
2.  NER; // Number of Elite Routes
3. Initialization: Randomly generate a population.
4. Estimation of Fitness & Arrangement: Estimate fitness of individuals and sort them with
fitness value.
5.  Crossover Population Method (population)
6.  CP: Crossover Population; // generate new population
7.  for i from 0 to NER do
8.      CP← population;
9.  for i from NER to PS do
10   chromosome1 ← Call Tournament selection Method (population).chromosome(0);
11   chromosome2 ← Call Tournament selection Method (population).chromosome(0);
12.        CP.chromosome(i) ←  Call Single Point Crossover Method(chromosome1,
chromosome2);
13. return  CP ;
14.  Mutate Population Method (MutatePopulation )
15.  MP: Mutate Population; // generate new population
16.  for i from NER to PS do
17.      MP. chromosome (i) ← Call Random Swap Mutation Method(chromosome);
18.  return MP;
19.  Evolve Population Method (Evolve Population )
20.  return Call  Mutate Population Method (Call Crossover Population Method (Evolve Population));
```

From this, the visiting cities' permutation of the first salesman is 0 (Depot)→ 2→ 8→ 0 (Depot). The visiting cities' combination of the second salesman is 0 (Depot)→ 1→ 7→ 4→ 6→ 0 (Depot). Finally, the visited cities' combination of the third salesman is 0 (Depot) → 3→ 5→ 9→ 0 (Depot).

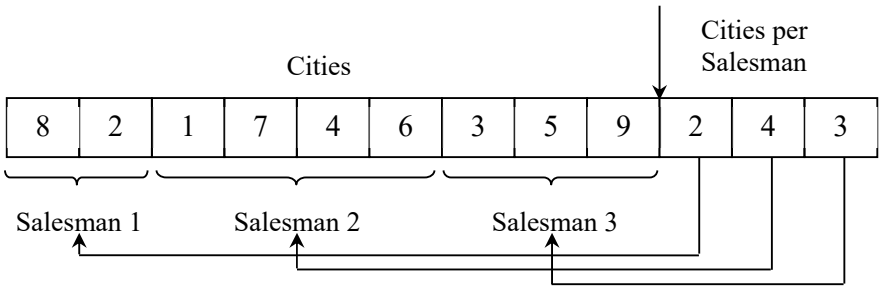

Fig. 2. Two parts chromosome technique for a 10 city MTSP with three salesmen

4.1.2. Parameter Selection

4.1.2.1. Population capacity

According to the schema theorem, with the increase of the population size, the genetic operators will deal with more patterns, and we will have a greater chance of finding the optimal solution (Altenberg, 1995). However, with the increase in the number of schemas, the computational load will be higher and the efficiency of the genetic algorithm will be lower.

4.1.2.2. Crossover probability

With higher crossover probability, the new structures in the population will appear faster. If the crossover probability is too high, the better structures cannot be well retained, so the loss rate of those good genes will accelerate. On the contrary, with lower crossover probability, the individuals will hardly exchange their genes, so the evolution velocity will be slower (Eiben et al., 1999).

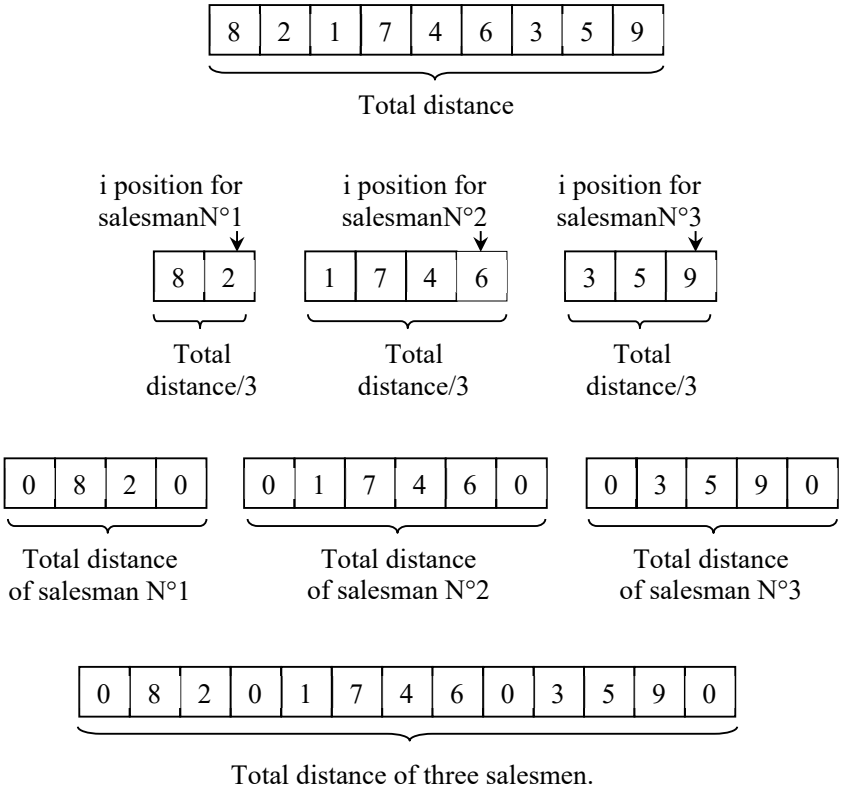

Fig. 3. Example of calculation of total distance of three salesmen for 10 cities MTSP.

4.1.2.3. Mutation probability

With lower mutation probability, the gene will be less likely to change, so the next generation will have less genetic variation. If mutation probability is too high, the genetic algorithm will turn to a random search (Bagchi & Pal, 2011).

4.1.3. Genetic Operators

In GA, Genetic operators are essential to evolve the chromosomes. They significantly influence the search ability and convergence speed. However, the choice of proper genetic operators is essential to prevent premature convergence, that is to say the solution is stuck in a local optimum.

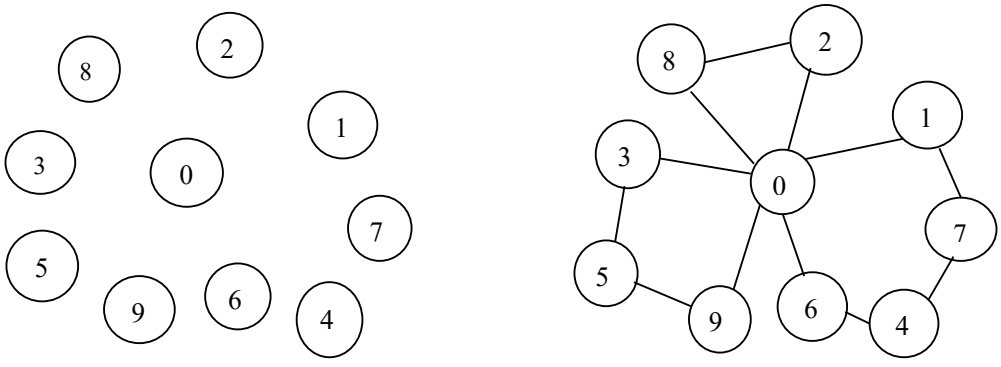

Fig. 4. The relative position of the city and one of its solutions

4.1.3.1. Selection Operator

Selection is the process of choosing two parents from the population for crossing. The purpose of selection is to emphasize fitter individuals in the population in hopes that their offsprings have higher fitness. Chromosomes are selected from the initial population to be parents for reproduction. Selection is a method that randomly picks chromosomes out of the population according to their evaluation function. The higher the fitness function, the more chance an individual has to be selected (Deepa & Sivanandam, 2010). The roulette wheel selection method is widely used as a selection operator. In this method, the expectation of each individual being selected is proportional to its fitness value. If the fitness value of each individual is greatly different, the probability of the best individual being selected will increase exponentially, that means, the survival chance of the best individual will be much greater than the worst individual. As the probability of the worst individual being selected decreases, the diversity of the population will also decline rapidly. The steps of the roulette wheel selection method are presented in Algorithm 2.

Algorithm 2
Roulette Wheel selection Method(population)

```
1.RWS: Roulette Wheel selection; // size of population
2.Generate Roulette Wheel population; // new population of RWSsize
3. for i from 0 to RWS do
4.Roulette Wheel population ← select best chromosome from population;
5.sort Roulette Wheel population;
6. return Roulette Wheel population;
```

Tournament Selection is a very popular method and in order to enhance the diversity of the population, we adopt the tournament selection method. There is a brief instruction about the method: randomly select a certain number of individuals from the current population, compare their fitness values and select two individuals with the largest fitness values to be parents for the next generation, create the next generation of individuals, and repeat the process until a new population meets the conditions. But the tournament selection method may converge prematurely to a local optimum. The steps of the tournament selection method are presented in Algorithm 3.

Algorithm 3
Pseudocode of tournament selection Method

```
1.TS: tournament selection;// size of population
2.Generate Tournament population; // new population of TS size
3. for i from 0 to TS do
4.    Tournament population ← select randomly chromosome from population;
5.sort Tournament population;
6. return Tournament population;
```

4.1.3.2. Crossover Operator

Parent-centric and mean-centric operators are two main approaches of crossover development. The first approach generates offspring near each of the parents whereas the second approach generates offspring solutions near the centroid of the parents, which is close to the mean of the participating parents. The most famous crossover operators are described in the following paragraph (Mool, 2016). In this paper, we discuss two classic examples.

- Single point crossover:

This operator detects one crossover point at random position before splitting parents at this crossover point thereby producing offspring by exchanging tails (Holland, 1975). The range of the common crossover probability is [0.2; 1.0].

- N-point crossover:

This operator is a generalization of the single point crossover (Eshelman, 1997). The *n* crossover points are picked randomly from the parent chromosomes, after which the genes in between points are swapped between the parents organisms. In this paper, we adopt the path representation method. If we continue to use the single or two-point crossover method, there will be an illegal route, which means each city may appear twice or more in this route, so the final solution cannot be considered as a feasible solution. In order to solve this problem, we adopt the order insert crossover method. There is a brief instruction about the method: randomly select single or two crossover positions, select the cities before the single crossover positions or between the two crossover positions from parent p1 or p2 by referring to crossover rate (if the random number is higher than the crossover rate, select p1 and vice versa), save the relative order of the cities in rest parent, and generate the child o, which is a new individual. For example (shown in Fig. 5) take 7 cities 1,2,3,…..,7. We already have parents p_1,p_2:

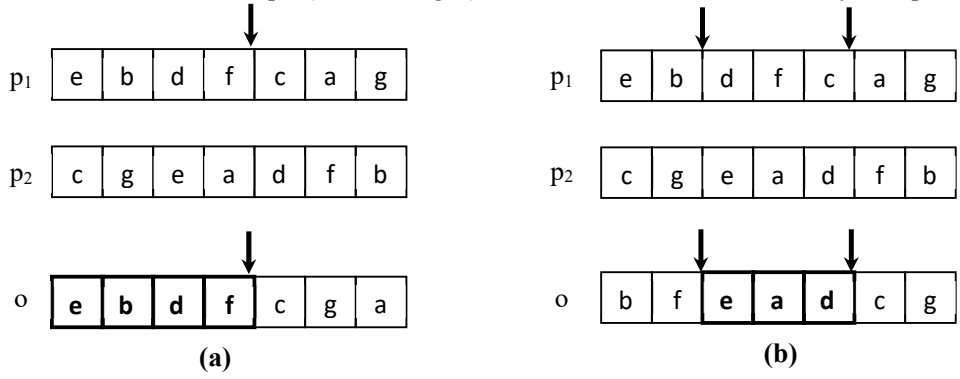

Fig. 5. Example of crossover operations: (a) the single point crossover and the order insert crossover method and (b) the two-point crossover and order insert crossover method

For part (a), the random number selected superior to crossover rate and randomly selected single crossover position number 4 (the fourth gen 4) from p1, the cities before the single crossover position should remain unchanged. For other rest positions in o, if the cities in p_2 never exist in o, insert them one by one according to the order of p_2.

For part (b), the random number selected is inferior to crossover rate and randomly select two crossover positions 3,5 (the third gene e and the fifth gene d) from p2, the cities between the two crossover positions should remain unchanged. For other rest positions in o, if the cities in p_1 never exist in o, insert them one by one according to the order of p1. The steps of the single Point crossover method and the two-point crossover method are presented in Algorithm 4 and Algorithm 5, respectively.

Algorithm 4
Pseudocode of Single Point Crossover Method

```
1.CR :Crossover Rate;
2.CC : Crossover Chromosome;
3. if (Random Number < CR)
4.select chromosome2;
5. else
6.select chromosome1;
7. end if
8. RP : Random Position (0 to n) for selected chromosome;
9. for i from 0 to RP do
10.   CC ← selected chromosome;
11. for i from RP+1 to n do
12.CC ← non selected chromosome for gene never exist in CC one by one;
13. return CC;
```

Algorithm 5
Pseudocode of Two-Point Crossover Method

```
1. CR : Crossover Rate;
2. CC : Crossover Chromosome;
3. if (Random Number < CR)
4. select chromosome2;
5. else
6. select chromosome1;
7. end if
8. RP1 : Random Position (0 to n) for selected chromosome;
9. RP2 : Random Position (0 to n) for selected chromosome;
10. if ( RP1< RP2)
11.for i from RP1 to RP2 do
12.CC ← selected chromosome;
13.for i from 0 to RP1-1 do
14.for j from RP2+1 to n do
15.CC ← non selected chromosome for gene never exist in Crossover
chromosome one by one;
16.endif
17.return CC;
```

4.1.3.3. Mutation Operator

Various studies on varieties of mutation techniques have been carried out to improve the performance of FA in recent years (Tang & Tseng, 2013). To change the genes of the offspring and to increase the diversity of the population is the main purpose of mutation operation.To avoid premature convergence, the mutation process enables GAs to jump out of local or sub-optimal solutions (Mooi, 2016).

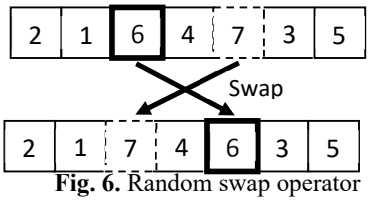

Fig. 6. Random swap operator

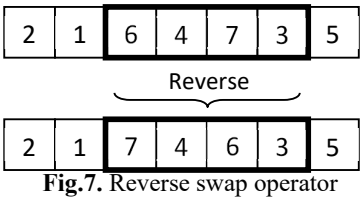

Fig.7. Reverse swap operator

Algorithm 6
Pseudocode of Random Swap Mutation Method

```
1. MR : Mutation Rate;
2. RP : Random Position (0 to n) for chromosome;
3. if (Random Number < MR)
4. for i from 0 to n do
5. if (i == RP)
6. Switch the gene pattern in the chromosome;
7.endif
8. endif
9.return chromosome;
```

Algorithm 7
Pseudocode of Random Reverse Swap Mutation Method

```
1. MR : Mutation Rate;
2. RP1 : Random Position (0 to n) for chromosome;
3. RP2 : Random Position (0 to n) for chromosome;
4. if (Random Number < MR and RP1< RP2 and RP2< MR)
5. Reverse gene pattern in chromosome from RP1 to RP2
8. endif
9.return chromosome;
```

Regarding the first part of the chromosome, there are two mutation operators, the random swap operator and the reverse swap operator. For the random swap operator, we select two distinct random positions (G_i and G_j), where $i \neq j$, then we swap the genes in these two positions (Banzhaf, 1990; Beed et al., 2017). For the reverse swap operator, we select two separate random positions to define the segment, and then we reverse the position of the genes inside the segment (Grefenstette, 2013).

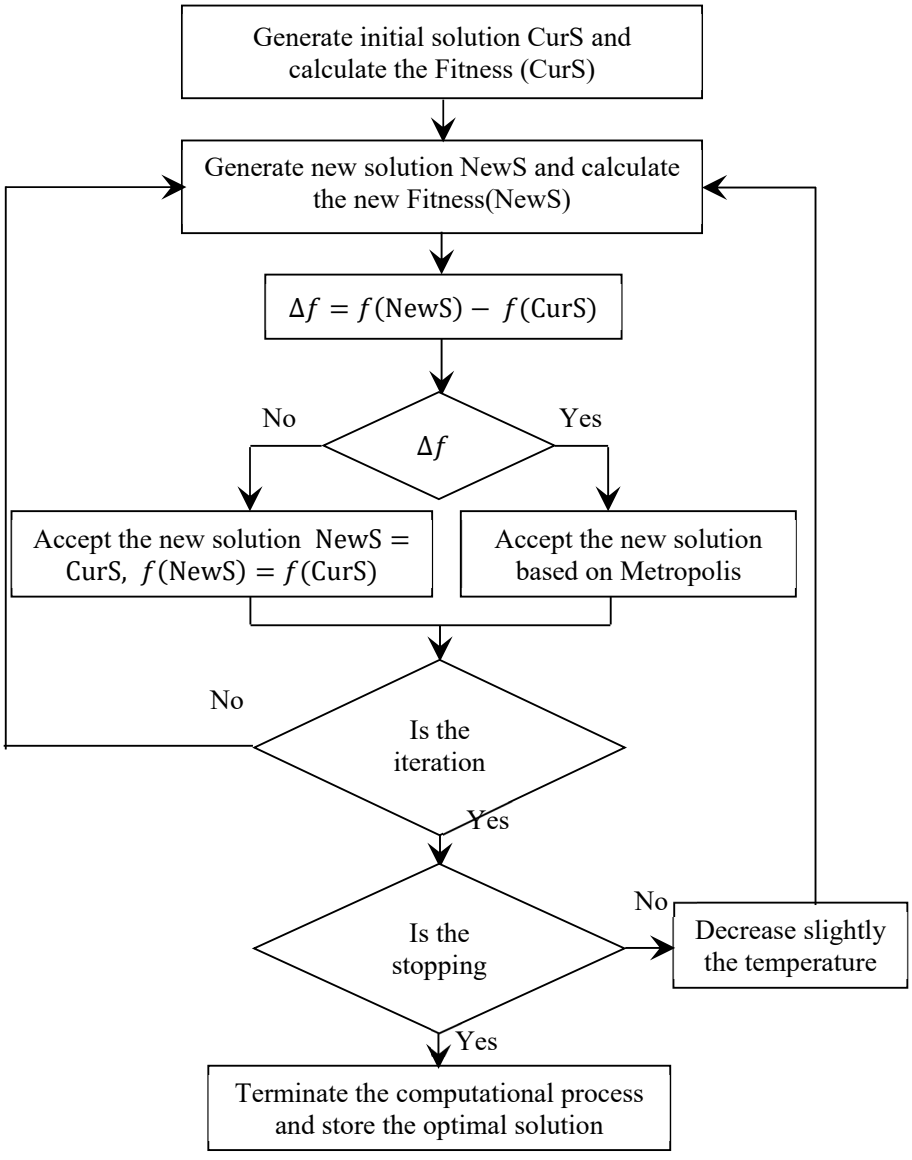

Fig. 8. A flowchart of the simulated annealing algorithm

In this paper, the path representation is adopted for encoding, but traditional simple mutation will lead to a duplication or loss of cities. So we adopt the mutation operators mentioned before. Because the random swap and the reverse swap methods only exchange pre-existing values, it will never create a list which has missing or duplicate values when compared to the original (Beed et al., 2017). Examples in Fig. 6 and Fig. 7, respectively showed a brief instruction about these operators. The steps of the random swap mutation method and the Random Reverse Swap Mutation Method are presented in Algorithm 6 and Algorithm 7, respectively.

4.2. Simulated annealing Algorithm

Simulated annealing (SA) is an iterative search method inspired by the annealing of metals (Mooi, 2016; Banzhaf, 1990). The algorithm performs a stochastic search in the partial state space starting with an initial solution and armed with adequate perturbation and evaluation functions. With probability controlled by a parameter called temperature (T), uphill moves are sometimes accepted. The probability of acceptance of uphill moves decreases as T decreases. By increasing the temperature, the search becomes more and more random, while at low temperature the search becomes almost greedy. At zero temperature, the search becomes totally greedy (Kirkpatrick et al., 1983; Černý, 1985). The basic principle of the algorithm is the Metropolis procedure, which simulates the annealing process at a given temperature T (Metropolis et al., 1953). This procedure is named after the scientist who devised a similar scheme to simulate a collection of atoms in equilibrium at a given temperature.

Algorithm 8

Pseudocode of Simulated annealing Algorithm

initial temperature T_0, minimum temperature T_{min}, probability of temperature drop;
1. Generating an initial solution CurS;
2. BestS← CurS;
3. Computing the value of the Fitness function $f(\text{CurS})$ and $f(x_{best})$;
4. i ← 0;
5. $T_i \leftarrow T_0$;
6. while $T_i > T_{min}$ do
7. Δf ← f(NewS) – f(BestS);
8. if Δf < 0 then
9. BestS ← NewS;
10. endif
11. if Δf > 0 then
12. $p \leftarrow e^{\frac{\Delta f}{T}}$;
13. if c ← random[0,1] ≥ p then
13. BestS ← NewS;
14. else
15. BestS ← BestS;
16. endif
17. endif
18. i ← i + 1;
19. return chromosome;

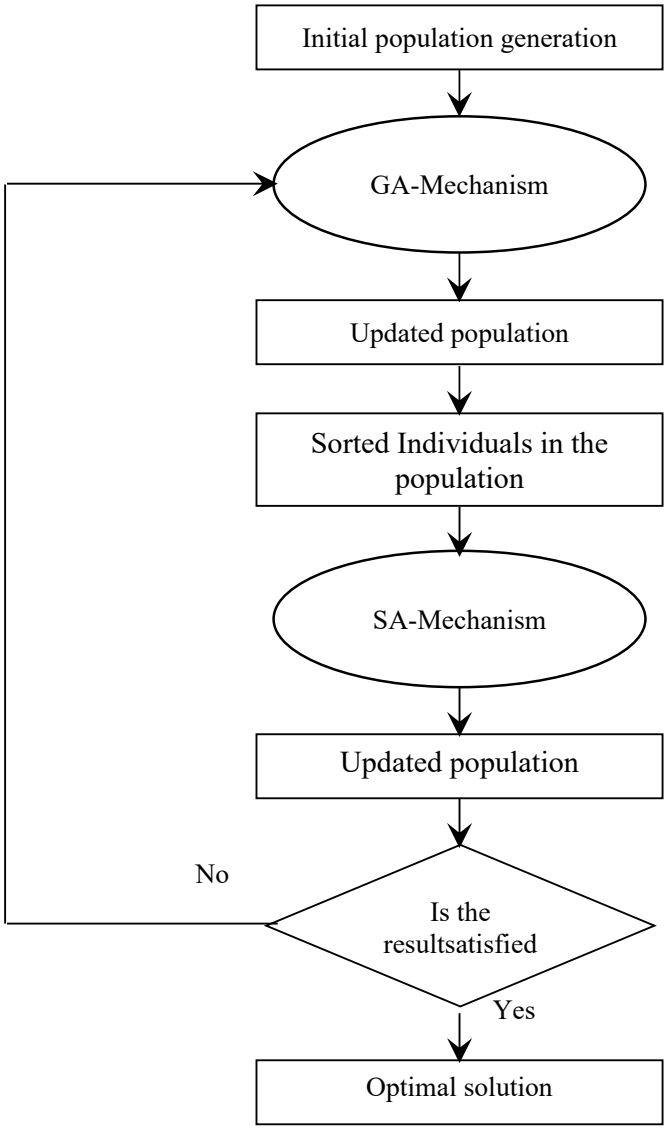

Fig. 9. A flowchart of the hybrid algorithm

Temperature is initialized to a value T_0 at the beginning of the procedure, and is slowly reduced; the parameter α is used to achieve this cooling. The Metropolis procedure uses the procedure Neighbor to generate a local neighbor NewS of any given solution S. The function Fitness returns the fitness of a given solution S. If the fitness of the new solution NewS is better than the fitness of the current solution Sc, then the new solution is accepted, and we do so by setting CurS = NewS. If the fitness of the new solution is better than the best solution (BestS) seen thus far, then we also replace BestS by NewS. If the new solution has a lower fitness in comparison to the original solution CurS, Metropolis will accept the new solution on a probabilistic basis. A random number is generated in the range 0 to 1. If this random number is smaller than , where Δfitness = fitness(NewS) - fitness(CurS), and T is the current temperature, the inferior solution is accepted. This criterion for accepting the new solution is known as the Metropolis criterion. From Fig. 8, we can see that the Metropolis procedure is very important for SA to find the optimal solution. The SA algorithm needs to start from a high temperature (T). However, if this initial value of T is too high, it causes a waste of processing time. The initial temperature value should be such that it allows virtually all proposed uphill or downhill moves to be accepted. The temperature parameter is initialized using the procedure described in (Wong & Liu, 1986). The parameter α for updating the temperature is user specified. In our implementation α takes the value of 0.05. At each updated value of the temperature, a number of state transitions are made so as to reach the probabilistic steady state. The perturb mechanisms employed are similar to the mutation operators in GA. The stopping criterion is when the final $T < 0.001$. The steps of the SA algorithm are shown as in Fig. 9.

4.3. Hybrid algorithm

SA operator is a local search operator (Eglese, 1990), which searches for the optimal solution more effectively due to its high local search ability. By generating some neighborhood solution in some mechanism, the near-optimal feasible solution can be incessantly updated until obtaining the optimal solution. For this reason and in order to produce a good solution, a GA combined with SA operator is presented. The motivation behind the development of this algorithm is to take advantage of the high convergence rate of SA on GA for MTSP. The complete process of the proposed algorithm is shown in Fig. 9. For solving MTSP with *m* cities, the hybrid approach uses the same population of GA which is randomly generated. Hybrid GA-SA algorithm is explained in Algorithm 9. Individuals of population obtained from GA are sorted in descending order of fitness, and fittest individuals are delivered to SA operators. The SA processes will be used to improve the results by using the nearest solution technique. If no improvement results are obtained in the ten consecutive iterations, then the best memorized population from SA will be moved to the GA to repeat the above process.

Algorithm 9
Pseudocode of Hybrid algorithm

1. **Initialization:** Randomly generate a population**.**
2. **Estimation of Fitness & Arrangement:** Estimate fitness of individuals and sort them with fitness value.
3. **GA-Mechanism:** Apply the classical genetic algorithm on the individuals of the population and produce new population.
   A. **Selection:** Employ proportionate selection to build a mating pool.
   B. **Crossover:** Employ order-based crossover to create new offspring.
   C. **Mutation:** Perform mutation on new offspring via scamble operator.
4. **SA-Mechanism:** Apply the SA on best individuals to update their position.
5. **Repeat:** Start from step 2 until termination criterion is not satisfied.

5. Experiments

This section first introduces bi-objective symmetric MTSP instances and corresponding parameter settings in Sec. 5.1. Then experiments results and the statistical analysis are implemented in Sec. 5.2.

5.1. Datasets and parameters

For the sake of contrastive analysis, the problem set contains 5Bi-objective symmetric MTSP instances with the numbers of cities ranging from 10 to 100. The benchmark details are listed in Table 1. All selected parameters values for GA and SA have been chosen empirically and are listed in Table 2. All experiments were implemented in Java on a CPU Intel Core i5-8250 with 1.8 GHz and 8 GB of RAM. The different algorithms used in this paper are described in Table 3.

Table 1
Parameters of the benchmark problems

Instances	No of cities	Number of salesmen
GSA10	10	
GSA30	30	
GSA50	50	3 / 5 /7
GSA70	70	
GSA100	100	

Table 2
Parameters for GA and Simulated Annealing Algorithm

Parameters	Values
Population Size	500
Random Swap Rate	25%
Reverse Swap Rate	50%
Crossover Rate	40%
Initial temperature	999
Minimum Temperature	0.99
Rate of Cooling	0.05

Table 3
Description of used algorithm

Algorithm	Description
GASCSM	Genetic algorithm with single point crossover and swap mutation.
GASCRSM	Genetic algorithm with single point crossover and reverse swap mutation.
GATCSM	Genetic algorithm with two point crossover and swap mutation.
GATCRSM	Genetic algorithm with two point crossover and swap reverse mutation.
SA	Simulated Annealing.
HGSA	Hybrid Genetic-Simulated Annealing Algorithm.

5.2 Results and analysis

In this section, we present experimental results and carry out statistical analyses. All experiments are implemented in the same environment to enable fair comparisons between all algorithms. Four genetic algorithms with different combinations of crossover and mutation operators are formulated. Also a simulated annealing and one of the best efficient of four algorithms previously described is hybridized with simulated annealing algorithm. This experiment aims to compare the performance of the operators under different numbers of salesmen. Therefore, we set the maximum number of generations to a fixed value, which is used as a termination criterion. In order to wipe off the computational fluctuation, all results in our experiments are averaged over 40 times.

5.2.1. Experimental Results

The results are reported in Table 4. Hybrid Genetic-Simulated Annealing Algorithm obtains the best-fitness solution compared to other settings in most of the test cases. The fitness is defined as the sum of the salesmen's path distance, which is the smaller the better. The average fitness values are improved by 19.5%, 17.5%, and 20.57% with 3, 5 and 7 salesmen respectively compared to the results of GASCSM. From the experiment, we observed that HGSA works poorly with a small ratio of the number of cities. That means that for each salesman visiting, it is more likely to be a short path. We estimate that this type of path can reach a local optimum during the GA stage because the fitness function is the total distance of all salesmen. Hence, since the number of cities is low, applying HGSA will not be helpful, as the overall optimization of MTSP requires an exchange of cities between salesmen.

To stimulate HGSA, a threshold for HGSA can be introduced. When the fitness improvement obtained by HGSA is below a threshold (k%), it will be disabled for some cycles. This will help save time and avoid creating troubles to find a better solution by exchanging cities. The cooling operators rate for simulated annealing and HGSA algorithms can also be settled to different problems. If the application is urgent, HGSA can be turned off to get acceptable results in the fastest way. If the computational resources and the time are adequate, the application is critical in terms of cost, the use of both operators will bring the best benefit to the result.

Regarding the multi-objective MTSP problem, we changed the objective function (fitness function) of all algorithms to plot the graphs of the variation of the difference between the longest sub-tour and the shortest sub-tour (amplitude of tours) as a function of total distance of tours. In all the following figures, the abscissa represents the total distance of tours and the ordinate represents the amplitude of tours. In Fig. 10, it is obvious that the non-dominated front obtained by HGSA is better than the non-dominated front edge obtained by all other algorithms. The former solution tends to have less total distance and lower difference. Compared with the other five algorithms, the solution from our algorithm can dominate most of the solutions obtained by the other methods, especially the instance GSA100-m7. In GSA30-m7, our result is to show a clear advantage in the objective of minimizing the total distance, but not very good in minimizing the balance.

As shown in GSA100-m7, combining HGSA and SA still performs better than the other algorithms at the same number of iterations. In Figure 8, the results of three instances of the genetic algorithm with different operators have relatively poor performance and it is obvious that our algorithm HGSA can get a better non-dominated front.

In Fig. 11, our algorithm HGSA also gets a better non-dominated front, especially in the GSA100-m3 example, where our non-dominated solution set reflects a good diversity. Since HGSA is based on the search of shortest distance accurately for each iteration for genetic algorithm and for each rate of cooling for simulated annealing algorithm, our results do not perform well in the extreme case of minimizing the balance in some instances. To see the effect of the number of salesmen and the number of cities on the algorithm results and by comparing the results of Fig. 10 and Fig. 11, it appears that our algorithm is efficient in increasing the number of cities as it's not affected by the number of salesmen. In general, the non-dominated front derived from our algorithm is closer to the origin than the non-dominated solution set obtained by any of the other five algorithms, the results we obtain simultaneously are of good diversity and can provide more options to decision makers.

.

Table 4
Fitness comparison with different algorithm and number of salesmen

Number of salesmen		3				5				7			
Problem	**Algorithm**	**Best**	**Avg**	**Worse**	**Time (s)**	**Best**	**Avg**	**Worse**	**Time (s)**	**Best**	**Avg**	**Worse**	**Time (s)**
GSA10	GASCSM	1840	2026	2310	0.261	2250	2396	2510	0.239	2710	2905	3320	0.27
	GASCRSM	1840	2122	2360	0.192	2290	2475	2670	0.244	2770	2917	3280	0.275
	GATCSM	1860	2034	2280	0.189	2250	2428	2670	0.221	2750	2872	3060	0.258
	GATCRSM	1890	2084	2390	0.204	2310	2493	2640	0.214	2710	3043	3300	0.254
	SA	1780	1823	1850	1.949	2210	2230	2250	4.1	2710	2710	2710	5.027
	HGSA	**1780**	**1807**	**1840**	7.297	**2210**	**2230**	**2250**	3.736	**2710**	**2710**	**2710**	19.793
GSA30	GASCSM	5087	5429	5730	0.754	5594	5871	6225	1.264	6021	6287	6476	1.682
	GASCRSM	4903	5509	6187	0.777	5389	5866	6907	1.179	5362	6153	7040	1.696
	GATCSM	5251	5427	5682	0.66	5384	5871	6139	1.126	6043	6266	6556	1.543
	GATCRSM	4462	5152	6267	0.744	5206	5868	6769	1.14	5558	6137	6832	1.568
	SA	4364	4551	4695	13.711	4941	5081	5180	21.57	5380	5463	5517	35.199
	HGSA	**4271**	**4345**	**4416**	247.665	**4778**	**4895**	**4970**	85.881	**5287**	**5349**	**5414**	123.285
GSA50	GASCSM	7351	7564	7957	2.99	7058	7823	8248	6.83	7837	8305	9653	7.846
	GASCRSM	6688	7501	8447	2.95	7199	7789	8905	4.919	7666	8172	9294	9.305
	GATCSM	7033	7584	7946	2.847	7677	7909	8261	4.496	7729	8243	8578	8.416
	GATCRSM	6910	7315	9005	3.757	7743	8398	9028	6.291	7378	8082	9250	10.303
	SA	6158	6409	6555	62.387	6520	6694	6882	115.533	6948	7075	7221	160.366
	HGSA	**5900**	**6061**	**6180**	975.754	**6473**	**6585**	**6720**	784.127	**6437**	**6569**	**6689**	431.028
GSA70	GASCSM	10069	10386	10766	8.634	10185	10790	11222	15.689	10737	11134	11600	28.702
	GASCRSM	9669	10501	11624	8.703	9954	11037	11975	16.497	10379	11359	12490	22.22
	GATCSM	10046	10371	10701	10.272	10506	10878	11336	18.245	10368	11142	11597	20.391
	GATCRSM	9530	10499	11386	8.005	10290	11054	12133	14.18	9857	10726	12386	20.088
	SA	8419	8813	9127	195.212	9213	9361	9431	347.013	9198	9689	9982	512.763
	HGSA	**8401**	**8578**	**8718**	2830.079	**9010**	**9186**	**9350**	1898.746	**9170**	**9345**	**9410**	2005.148
GSA100	GASCSM	16087	16656	17145	43.038	16284	16915	17390	75.276	16795	17521	19614	86.741
	GASCRSM	15769	16776	18096	35.126	15653	16583	18757	73.611	15512	17032	19345	91.061
	GATCSM	15985	16594	16945	39.749	16274	17005	17404	69.093	16621	17339	18036	96.137
	GATCRSM	15895	16533	17795	31.233	15973	17040	18664	69.863	16403	17461	18799	96.992
	SA	13618	13871	14212	813.488	14163	14569	14897	1427.573	14372	14818	15083	2041.302
	HGSA	**12949**	**13321**	**13583**	51842.682	**13430**	**14094**	**14260**	5159.875	**13858**	**14446**	**14800**	7693.246

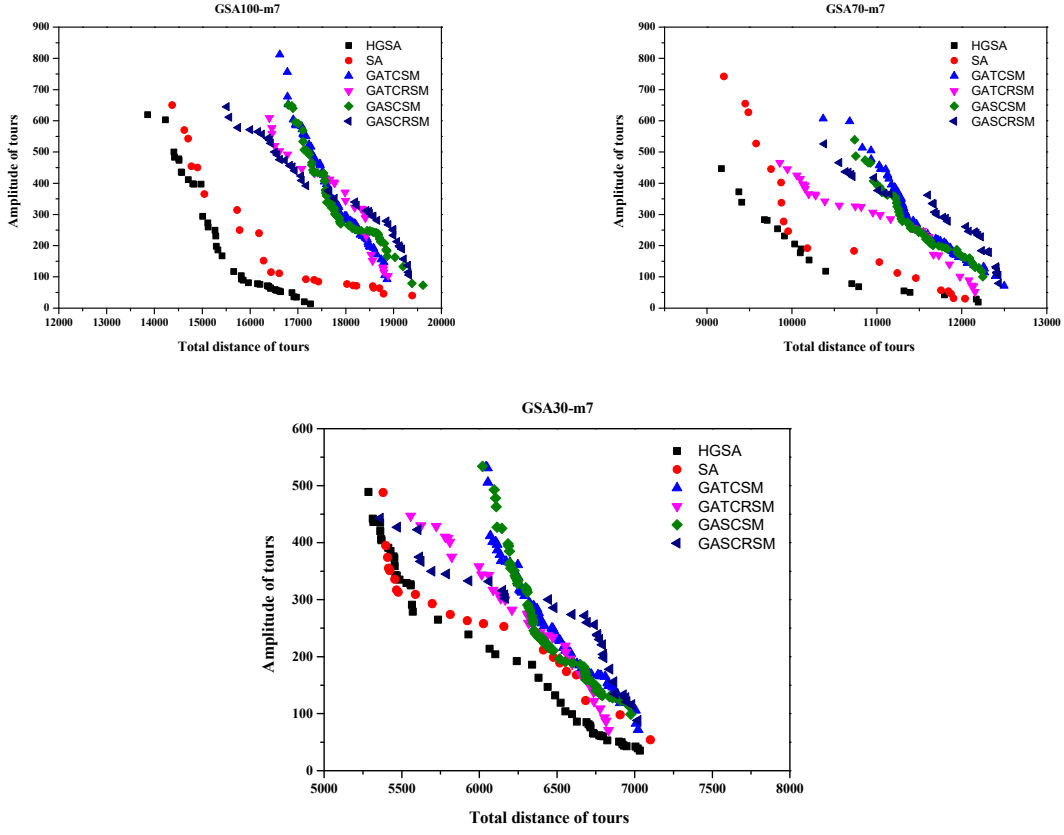

Fig. 10. Non-dominated fronts for GSA100-m7, GSA70-m7 and GSA30-m7 instances

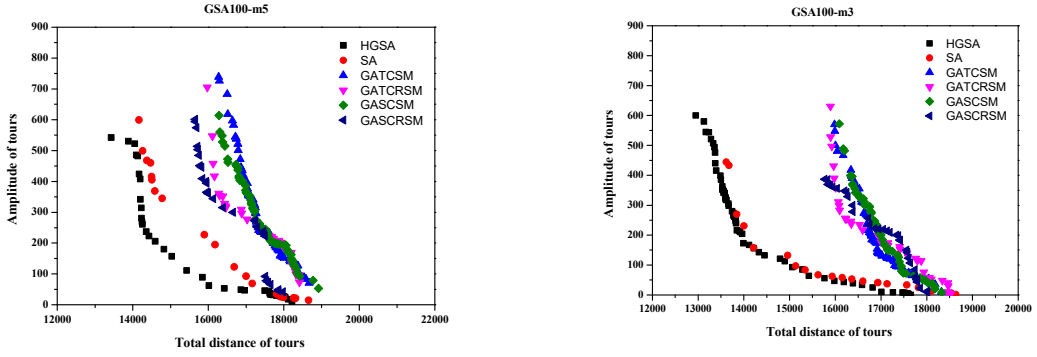

Fig. 11. Non-dominated fronts for GSA100-m5 and GSA100-m3 instances

5.2.2 Statistical Analyses

In order to verify that the HGSA algorithm is statistically superior to other algorithms, this section performs statistical analyses according to (García et al., 2009). Table 5 shows the average value of fitness in different scales of instances. Table 6 gives the rankings of different algorithms on the various datasets based on results in Table 5.

In Table 5, symbols G1 – G9 denote six instances taken in these statistical analyses.The values in Table 6 indicate the ranking results of six algorithms from best to worst based on the average fitness value (the sum of the salesmen's path distance) summarized in Table 5 of three instances GSA50, GSA70 and GSA100. R represents the average of all rankings on nine datasets for each algorithm. For example, the ranking R of HGSA can be calculated as, R = (1 + 1 +1 + 1+ 1 + 1) /9= 1.

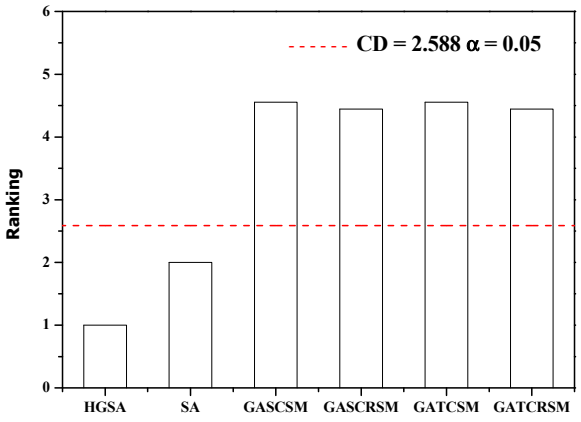

Fig. 12. The Bonferroni-Dunn's graph corresponding to results of Table 6. The horizontal line represents the value which equals to the sum of the ranking of control algorithm (i.e., HGSA) and the corresponding CD. Thos bars which exceed this line are the associated to an algorithm with worse performance than HGSA

Table 5

The comparison of measure of fitness value obtained by algorithms in different instances. From left to right, these instances are GSA50-m3, GSA50-m5, GSA50-m7, GSA70-m3, GSA70-m5, GSA70-m7, GSA100-m3, GSA100-m5, and GSA100-m7

Algorithm	G1	G2	G3	G4	G5	G6	G7	G8	G9
GASCSM	7564	7823	8305	10386	10790	11134	16656	16915	17521
GASCRSM	7501	7789	8172	10501	11037	11359	16776	16583	17032
GATCSM	7584	7909	8243	10371	10878	11142	16594	17005	17339
GATCRSM	7315	8398	8082	10499	11054	10726	16533	17040	17461
SA	6409	6694	7075	8813	9361	9689	13871	14569	14818
HGSA	6061	6585	6569	8578	9186	9345	13321	14094	14446

First, by Eq. (6), the statistics can be made in Friedman test (i.e. χ_F^2). R_j represents the rank of diverse algorithms. N and k express the number of datasets and algorithms, respectively. According to Eq. (6), the χ_F^2 of Table 6 is calculated to be 32.128. The degree of freedom of Table 6 can be obtained from the records in Chi-square table, i.e., k - 1 =5 and $\chi_{0.05}^2$= 11.071. Because 32.128 > 11.071, within the confidence interval of 95%, these algorithms in Table 6 show significant differences.

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{6}$$

Second, according to the significant differences among these algorithms, the Bonferroni-Dunn's test can be utilized to prove the specific distinction between two algorithms.This rule, whether the difference value between two algorithms in ranking is greater than the critical difference, is used as the evaluation criterion (denoted as CD). For a multiple comparison, α and q_α are the confidence level and threshold obtained by checking the Z table, respectively.Therefore, we can conclude that $q_{0:05}$ = 2.935 (where P = k (k - 1) / 2 = 15) for Table 6. According to Eq. (7), we can get the critical values at the 95% confidence levels, i.e., $CD_{0,05}$= 2.588.

$$CD_\alpha = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{7}$$

Since the performance of two algorithms is obviously different and that ranking difference is larger than CD_α, it can be conclude that HGSA is better than GASCSM, GASCRSM, GATCSM, GATCRSM and SA with α = 0,05 (95% confidence) based on Fig. 12.

$$Z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6N}}} \tag{3}$$

Finally, Holm's and Hochberg's methods are used to further compare the differences between two algorithms. To compare algorithms i and j, the statistic is calculated (denoted as zvalue) by Eq.(8). Generally, the ranking result is listed in reverse order. More Specifically, according on the z values, searching the normal distribution table can gain an unadjusted p (expressed as Up). From BD_{pi}= min{vi;1}, Bonferroni-Dunn p (expressed as BD_p) can be computed, where v_i= (k - 1)U_{pi}. From H_{pi}= min{vi;1}, Holm p (denoted as H_p) can be computed, where v_i= max{(k - j)U_{pj}: 1 ≤ j ≤ i}. From HB_{pi}=max{(k - j)U_{pj} : 1 ≥ j ≥ i}, Hochberg p (expressed as HBp) can be computed.

Table 6
The ranking obtained based on Table 3. The value means the ranking result of these algorithms in each instance.

Algorithm	G1	G2	G3	G4	G5	G6	G7	G8	G9	Ranking R
GASCSM	5	4	6	4	3	4	5	4	6	4.555
GASCRSM	4	3	4	6	5	6	6	3	3	4.444
GATCSM	6	5	5	3	4	5	4	5	4	4.555
GATCRSM	3	6	3	5	6	3	3	6	5	4.444
SA	2	2	2	2	2	2	2	2	2	2
HGSA	1	1	1	1	1	1	1	1	1	1

According to Holm's and Hochberg's procedures, Table 7 reports the statistical results of the data in Table 6.We can get the value of z by Eq. (8). Through searching and comparing the value of z and the value in the normal distribution table, we can obtain the probabilistic error estimation of a comparison (i.e., p-value).unadjusted p in Table 7 is p-valued squared. However, it does not consider the remaining comparisons, when p-value is in multiple comparisons. Adjusted p-value (APVs) considers multiple tests and can be used directly as the sumptive p-value in the multiple algorithm comparison range.Bonferroni-Dunn p (BDp), Holm p (Hp) and Hochberg p (HBp) represent the three calculated APVs. According to such a comparison, we can conclude that HGSA is better than other comparison algorithms at the 95% confidence level.

Table 7
The p-value on datasets G1 - G9 (HGSA is the control algorithm), which reports statistical results of Table 4.

HGSA vs.	z	Unadjusted p	Bonferroni-Dunn p	Holm p	Hochberg p
GASCSM	4.03	0	0	0	0.2417
GATCSM	4.03	0	0	0	0.2417
GASCRSM	3.90	0.000099	0.000495	0.000297	0.2417
GATCRSM	3.90	0.000099	0.000495	0.000297	0.2417
SA	1.13	0.2417	1	0.2417	0.2417

6. Conclusion

In this paper, a novel MTSP solving hybrid algorithm, called Hybrid Genetic-Simulated Annealing Algorithm (HGSA) has been proposed and developed. The crossover and mutation operators have been successfully deployed to generate four algorithms of the genetic algorithm. Then one of the best efficient of four algorithms previously described was hybridized with a simulated annealing algorithm. We also compared the performance of HGSA with the other proposed algorithms.

The Simulated Annealing operator increased the diversity of individuals, which could enhance the exploration space and avoid falling into the local optimum. According to such comparisons in the statistical analyses, it is easy to reach the conclusion that HGSA is better than other comparison algorithms at the 95% confidence level.

Also, the total travel distance and the difference between the longest sub-tour and the shortest one represented two conflicting objectives. Our algorithm (HGSA) was tested and compared with other five algorithms. The comparison results showed that the efficiency of diversity was realized for both of the two objective functions of our algorithm.

For future work, the proposed method can be further improved by using novel crossover and mutation operators that are not classic. Furthermore, the operators can be applied to other variations of MTSP problems, like multiple depots MTSP to find a better solution efficiently.

Acknowledgment

The authors are thankful to the Deanship of Graduate Studies and Scientific Research at University of Bisha for supporting this work through the Fast-Track Research Support Program.

Conflicts of interest

Author declares no conflicts of interest.

References

Albayrak, M., & Allahverdi, N. (2011). Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms. *Expert Systems with Applications, 38*(3), 1313-1320.

Al-Omeer, M. A., & Ahmed, Z. H. (2019, April). Comparative study of crossover operators for the MTSP. *In 2019 International Conference on Computer and Information Sciences (ICCIS) (pp. 1-6). IEEE*.

Altenberg, L. (1995). The schema theorem and Price's theorem. *In Foundations of genetic algorithms (Vol. 3, pp. 23-49). Elsevier*.

Ann, S., Kim, Y., & Ahn, J. (2015). Area allocation algorithm for multiple UAVs area coverage based on clustering and graph method. *IFAC-Papers OnLine, 48*(9), 204-209.

Bagchi, P., & Pal, S. (2011, April). Controlling crossover probability in case of a genetic algorithm. *In International Conference on Advances in Information Technology and Mobile Communication (pp. 287-290). Berlin, Heidelberg: Springer Berlin Heidelberg*.

Banzhaf, W. (1990). The "molecular" traveling salesman. *Biological Cybernetics, 64*(1), 7-14.

Beed, R. S., Sarkar, S., Roy, A., & Chatterjee, S. (2017, December). A study of the genetic algorithm parameters for solving multi-objective travelling salesman problem. *In 2017 International conference on information technology (ICIT) (pp. 23-29). IEEE*.

Bodenhofer, U. (2004). Genetic algorithms: theory and applications.

Bremermann, H. J., Rogson, M., & Salaff, S. (1965). Search by evolution. *Biophysics and Cybernetic Systems, 157-167*.

Brown, E. C., Ragsdale, C. T., & Carter, A. E. (2007). A grouping genetic algorithm for the multiple traveling salesperson problem. *International Journal of Information Technology & Decision Making, 6*(02), 333-347.

Carter, A. E., & Ragsdale, C. T. (2006). A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European journal of operational research, 175*(1), 246-257.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications, 45,* 41-51.

Deepa, S. N., & Sivanandam, S. N. (2010). Introduction to genetic algorithms. *Springer*.

Eglese, R. W. (1990). Simulated annealing: a tool for operational research. *European journal of operational research, 46*(3), 271-281.

Eiben, Á. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on evolutionary computation*, *3*(2), 124-141.

Eshelman, L. J. (1997). Crossover operator biases: Exploiting the population distribution. *In Proceedings of the 7th International Conference on Genetic Algorithms (pp. 354-361)*.

Filip, E., & Otakar, M. (2011). The travelling salesman problem and its application in logistic practice. *WSEAS Transactions on Business and Economics, 8*(4), 163-173.

García, S., Molina, D., Lozano, M., & Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics, 15*, 617-644.

Grefenstette, J. J. (Ed.). (2013). Genetic algorithms and their applications: *proceedings of the second international conference on genetic algorithms. Psychology Press*.

Harrath, Y., Salman, A. F., Alqaddoumi, A., Hasan, H., & Radhi, A. (2019). A novel hybrid approach for solving the multiple traveling salesmen problem. *Arab Journal of Basic and applied sciences, 26*(1), 103-112.

Holland, J. H. (1975). Adption in natural and artifieial system.

Holland, J. H. (1992). Genetic algorithms. *Scientific american, 267*(1), 66-73.

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. science, *220*(4598), 671-680.

Labadie, N., Melechovsky, J., & Prins, C. (2014). An evolutionary algorithm with path relinking for a bi-objective multiple traveling salesman problem with profits. Applications of Multi-Criteria and Game Theory Approaches: *Manufacturing and Logistics, 195-223*.

Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research, 59*(2), 231-247.

Laporte, G., & Nobert, Y. (1980). A cutting planes algorithm for the m-salesmen problem. *Journal of the Operational Research society, 31,* 1017-1023.

Li, J., Sun, Q., Zhou, M., & Dai, X. (2013, October). A new multiple traveling salesman problem and its genetic algorithm-based solution. *In 2013 IEEE international conference on systems, man, and cybernetics (pp. 627-632). IEEE*.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research, 21(2), 498-516*.

Lo, K. M., Yi, W. Y., Wong, P. K., Leung, K. S., Leung, Y., & Mak, S. T. (2018). A genetic algorithm with new local operators for multiple traveling salesman problems. *International Journal of Computational Intelligence Systems, 11*(1), 692-705.

Malmborg, C. J. (1996). A genetic algorithm for service level based vehicle scheduling. *European journal of operational research, 93*(1), 121-134.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics, 21*(6), 1087-1092.

Ming, L. I. U., & Pei-yong, Z. H. A. N. G. (2014). New hybrid genetic algorithm for solving the multiple traveling salesman problem: an example of distribution of emergence materials. *Journal of Systems & Management, 23*(2), 247.

Mooi, L. S. (2016). Crossover and mutation operators of real coded genetic algorithms for global optimization problems.

Necula, R., Breaban, M., & Raschip, M. (2015, November). Tackling the bi-criteria facet of multiple traveling salesman problem with ant colony systems. *In 2015 IEEE 27th international conference on tools with artificial intelligence (ICTAI) (pp. 873-880). IEEE*.

Qiang, N., & Kang, F. J. (2014). A hybrid particle swarm optimization for solving vehicle routing problem with stochastic demands. *Advanced Materials Research, 971,* 1467-1472.

Reinelt, G. (2003). The traveling salesman: computational solutions for TSP applications *(Vol. 840). Springer*.

Shim, V. A., Tan, K. C., & Tan, K. K. (2012, June). A hybrid estimation of distribution algorithm for solving the multi-objective multiple traveling salesman problem. *In 2012 IEEE congress on evolutionary computation (pp. 1-8). IEEE*.

Shmoys, D. B., Lenstra, J. K., Kan, A. R., & Lawler, E. L. (Eds.). (1985). The traveling salesman problem (Vol. 12). *John Wiley & Sons, Incorporated*.

Shuai, Y., Yunfeng, S., & Kai, Z. (2019). An effective method for solving multiple travelling salesman problem based on NSGA-II. *Systems Science & Control Engineering, 7*(2), 108-116.

Singh, D. R., Singh, M. K., Singh, T., & Prasad, R. (2018). Genetic algorithm for solving multiple traveling salesmen problem using a new crossover and population generation. *Computación y Sistemas, 22*(2), 491-503.

Soylu, B. (2015). A general variable neighborhood search heuristic for multiple traveling salesmen problem. *Computers & Industrial Engineering, 90,* 390-401.

Tang, L., Liu, J., Rong, A., & Yang, Z. (2000). A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *European Journal of Operational Research, 124*(2), 267-282.

Tang, P. H., & Tseng, M. H. (2013). Adaptive directed mutation for real-coded genetic algorithms. *Applied Soft Computing, 13*(1), 600-614.

Wong, D. F., & Liu, C. L. (1986, June). A new algorithm for floorplan design. *In 23rd ACM/IEEE Design Automation Conference (pp. 101-107). IEEE*.

Xu, X., Yuan, H., Liptrott, M., & Trovati, M. (2018). Two phase heuristic algorithm for the multiple-travelling salesman problem. *Soft Computing, 22,* 6567-6581.

Yousefikhoshbakht, M., Didehvar, F., & Rahmati, F. (2013). Modification of the ant colony optimization for solving the multiple traveling salesman problem. *Romanian Journal of Information Science and Technology, 16*(1), 65-80.

Yu, Z., Jinhai, L., Guochang, G., Rubo, Z., & Haiyan, Y. (2002, June). An implementation of evolutionary computation for path planning of cooperative mobile robots. *In Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No. 02EX527) (Vol. 3, pp. 1798-1802). IEEE*.