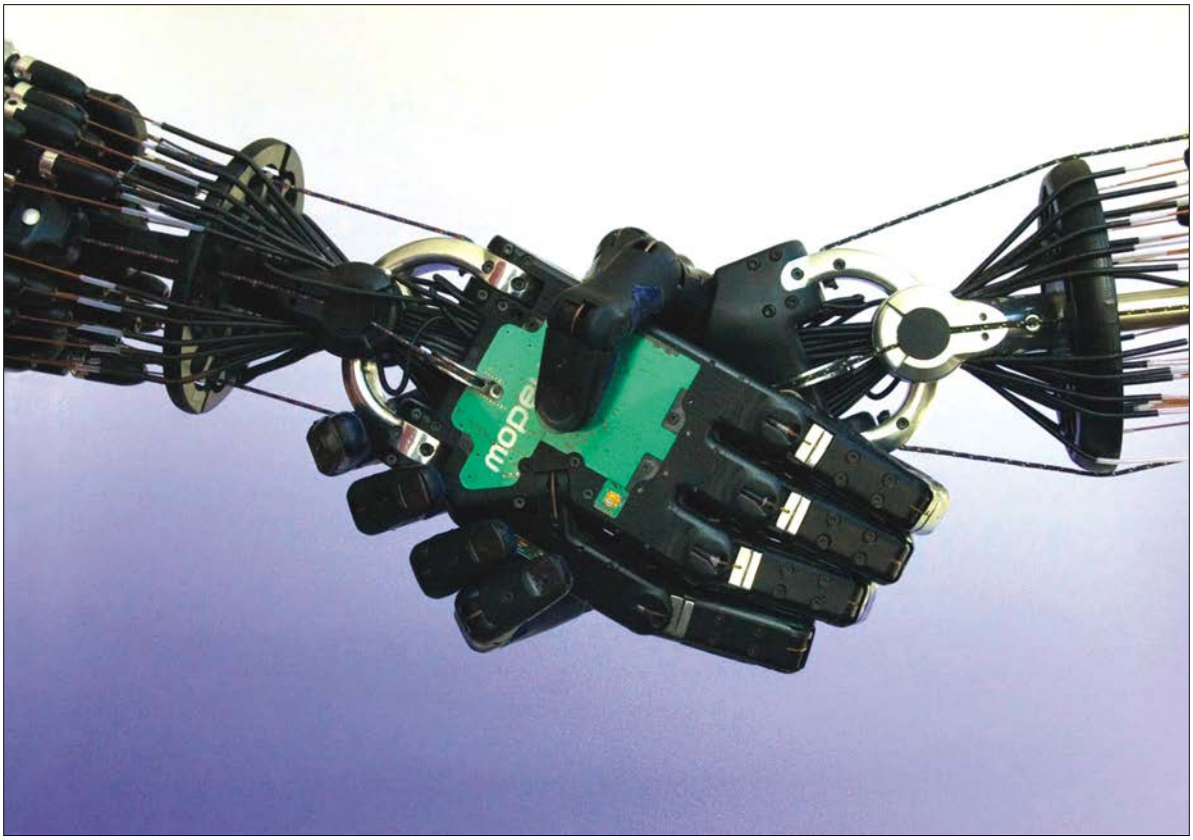

Journal of Artificial Intelligence

ISSN 1994-5450

science alert

ANSInet
an open access publisher
http://ansinet.com

OPEN ACCESS Journal of Artificial Intelligence

ISSN 1994-5450
DOI: 10.3923/jai.2016.12.22

Research Article
Performance Evaluation of Heuristic Methods in Solving Symmetric Travelling Salesman Problems

Yai-Fung Lim, Pei-Yee Hong, Razamin Ramli, Ruzelan Khalid and Md. Azizul Baten

School of Quantitative Sciences, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah, Malaysia

Abstract

Background and Objective: The Travelling Salesman Problem (TSP) is a challenging problem in combinatorial optimization whose main purpose is to find the shortest path reaching all interconnected cities by straight lines. In spite of many available heuristic methods for solving TSPs, no attempts have been made to evaluate and compare their performances. The purpose of this study is to carry out a comparative evaluation study on Simulated Annealing (SA) and several variation of Tabu Search (TS). **Materials and Method:** This study considers four heuristic methods, i.e., Simulated Annealing (SA), conventional Tabu Search (TS), Improved Tabu Search (ITS) and modified Reactive Tabu Search (RTS) to solve symmetric TSPs. The algorithms were tested on five chosen benchmark problems. Their performances were compared and the appropriate algorithm for solving TSPs was then identified. The solution quality was evaluated using empirical testing, benchmark solutions and probabilistic analyses. **Results:** The analysis of computational results showed that the modified RTS algorithm provided a better solution quality in terms of minimizing the objective function of TSPs, while the SA algorithm was useful for obtaining instant solutions for TSPs with a large number of cities. The modified RTS algorithm also performed better compared to the existing heuristic methods. **Conclusion:** This study has explored the most effective heuristic method for solving TSPs based on the intended solution quality. The algorithms proposed in this study should be considered in solving symmetric travelling salesman problems.

Key words: Traveling salesman problem, heuristic method, simulated annealing, tabu search, combinatorial optimization

Received: December 03, 2015 Accepted: May 02, 2016 Published: June 15, 2016

Citation: Yai-Fung Lim, Pei-Yee Hong, Razamin Ramli, Ruzelan Khalid and Md. Azizul Baten, 2016. Performance evaluation of heuristic methods in solving symmetric travelling salesman problems. J. Artif. Intel., 9: 12-22.

Corresponding Author: Yai-Fung Lim, School of Quantitative Sciences, University Utara Malaysia, 06010 UUM Sintok, Kedah, Malaysia

Competing Interest: The authors have declared that no competing interest exists.

Data Availability: All relevant data are within the paper and its supporting information files.

INTRODUCTION

The Traveling Salesman Problem (TSP) is one of the challenging classical combinatorial optimization problems[1-3]. It can be described as a salesman touring and visiting all of his/her customers at different cities exactly once before returning to his/her home so as to minimize the total distance of the tour. This concept has been applied to many real-life applications, e.g., vehicle routing[4], data clustering[5] and material handling in a warehouse[6,7].

Over the recent decades, the TSP has attracted attention of numerous researchers where various exact e.g., branch and bound and linear programming[2] and heuristic algorithms e.g., neural network[8] and ant colony[9] have been proposed to solve it. Since the exact algorithms only work well for TSPs containing a small number of cities, the heuristic methods are used to find solutions and demonstrate their efficiency for extremely large TSPs. Most widely known heuristic methods for solving TSPs are Genetic Algorithms (GA)[10-12], SA[1,7] and TS[13,14].

These algorithms have been found to be very effective and robust in solving numerous TSPs and can handle TSPs with missing parameters. The survey and comparison of the algorithms has been documented in the literature[2,15]. Their performances in solving TSPs have also been evaluated and compared in previous studies. For example, Youssef *et al.*[16] compared the performances of TS, GA and SA in terms of the quality of solution, searching progress, the number of iterations and the number of solutions at cost function intervals. Adewole *et al.*[17] meanwhile examined the performances of GA and SA on several benchmarks with a number of cities. Other comparative studies of GA and SA include[18-20]. All these studies reported that TS performed better than SA in terms of CPU time needed to reach a quality solution.

The performances of TS and SA in solving other types of problems have also been conducted. Hussin and Stutzl[21] as an example examined the dependencies of relative performances of TS and SA variants on instance sizes in the Quadratic Assignment Problem (QAP). Battiti and Tecchiolli[20] and Chiang and Chiang[22] also examined the performances of TS and SA in the QAP and reported that TS performed better than SA. The TS has been claimed to have an advantage over other heuristics techniques because of its intelligent use of memory along with responsive exploration in the solution space[23]. In spite of this fact, however not any studies comparing the performances of SA and different variations of TS in solving TSPs have been found. This therefore a comparative evaluation study on SA and several variations of TS which known has not been done before is inspired.

The objective of this study is to evaluate the performance of four heuristic methods, i.e., SA, TS, ITS and modified RTS in solving symmetric TSPs. This study extends the work of Lim *et al.*[24,25] who proposed some successful heuristic methods to solve symmetric TSPs whose bi-directional distances between a pair of cities are identical. In brief, they first introduced the ITS algorithm integrating two heuristic algorithms i.e., SA and TS. To improve the solution quality, they then developed the RTS algorithm which dynamically adjusts its tabu list size. In the mean time, Hong *et al.*[26] presented an empirical work to provide a comparable annealing schedule of the SA algorithm. They showed that with the right setting of an annealing schedule, a good solution quality could be achieved. In this study, the performances of the SA, TS, ITS and modified RTS algorithms to solve symmetric TSPs will be evaluated and compared. These four algorithms were analyzed using theoretical analyses and empirical testing. From the analysis of the results, the best algorithm for solving symmetric TSPs can then be identified and reported.

MATERIALS AND METHODS

Heuristics methods

Simulated Annealing (SA): The SA is a meta-heuristic optimization method founded on the annealing process of metal re-crystallization[27]. If this process is allocated with enough time, SA could then find the optimal solution of a considered problem. Based on this analogy of how metal is cool and annealed, each step of the SA algorithm replaces the current solution by a random nearby solution which is gradually decreased during the searching process.

The basic procedure of SA starts with an initial solution which then gradually moves to a nearby solution obtained by a local search. In order to obtain the nearby solution, 2-opt switch procedures which swap 2 cities have been used. The SA algorithm could escape from a local minimum by accepting non-improving moves based on the transition probability controlled by two factors i.e., the difference between the objective functions and the temperature as follows:

$$P(\delta) = \exp\left(\frac{-|\delta|}{t_k}\right)$$

where, δ is the cost difference between the candidate neighboring solution and the current solution and t_k is the current temperature.

The basic algorithm of SA is as in Fig. 1. The algorithm continues searching as temperature declines and stops once the temperature becomes zero. Thus, the efficiency of the annealing process is significantly affected by its annealing schedule which is controlled by the initial temperature, the cooling rate and the number of iterations which should be performed at each temperature. A favorable annealing schedule through the use of a statistical framework for solving symmetric TSPs has been discussed in detail by Hong *et al.*[26].

Tabu Search (TS): The TS is a meta-heuristic search method created by Glover[28]. It utilizes a hill climbing search strategy based on a set of elementary moves to diversify the search and a systematic use of memory to avoid any traps at local optimal points. Its key strategy is to move from solution to solution by accepting non-improving moves to the best solution in the neighborhood of the local optimal[29]. The main advantage of TS lies in the intelligent use of memory along with responsive exploration in the solution space[23]. The TS does not remember the current and the best solution. However, it keeps memory on the tour through the last solution visited to guide the move from the current to the next solution.

In TS, the memory ability is represented by its tabu list size. The use of memory helps intensify in elite regions or diversify the search towards unexplored regions. The balance between the intensification and diversification strategies is used to control and run the search process. To ensure an efficient search process, TS requires search parameters whose values significantly depend on the types of problems. Parameter tuning especially for the tabu list size is often needed to obtain good results[29].

The TS has been proved to successfully solve TSPs. However, since its performance depends significantly on its initial solution[30], Lim *et al.*[24,25] developed a modified version of TS called the Improve Tabu Search (ITS) algorithm to enhance the initial value of TS Unfortunately, besides the initial solution, the conventional TS and ITS algorithms are still suffering from parameter tuning in their tabu list sizes. In order to dynamically tune the tabu list size, they then presented the

```
begin
    read input data;
    generate an initial solution S;
    t=initial_t;
    previous_d=TotalDistance(s);
    best_d=previous_d;
 for(iterations=1;iterations<=max_iteration;iterations++)
 {
                for(repeat=1;repeat<=repeatation;repeat++)
                {
                                Swap(temp_s,s);
                                current_d=TotalDistance(temp_s);
                                diff=fabs(current_d-previous_d);
                                if(current_d<previous_d)
                                {
                                                previous_d=current_d;
                                                if(current_d<best_d)
                                                {
                                                                best_d=current_d;
                                                                best_iteration=iterations;
                                                }
                                }
                                else if((double)rand()/RAND_MAX<exp(-diff/t))
                                {
                                                previous_d=current_d;
                                }
                }
                repeat=1;
                t *= cooling_rate;
                if (t==0)
                {
                                break;
                }
                }
end.
```

Fig. 1: Pseudo code of SA

```
begin
    read input data;
    used SA to generate an initial solution S;
    current_d=TotalDistance(s);
    best_d=current_d;
    tabu[i][j]=0;
    for(iterations=1;iterations<=max_iteration;iterations++)
{
find all possible moves;
              if(m[1].Distance<best_d)
              {
                            current_d=m[1].Distance;
                            best_d=m[1].Distance;
                            best_iteration=iterations;
                            Update();
                            tabu[m[1].City1][m[1].City2]=tabu_list_size;
              }
              else
              {
                            i=1;
                            while (i<=k-1)
                            {
                                          if(tabu[m[i].City1][m[i].City2]==0)
                                          {
                                                        current_d=m[i].Distance;
                                                        Update();
                                                        tabu[m[i].City1][m[i].City2]=tabu_list_size;
                                                        break;
                                          }
                                          i++;
                                          if (i==k-1)
                                          {
                                                        if(tabu[m[i].City1][m[i].City2]!=0)
                                                        {
                                                                      current_d=m[1].Distance;
                                                                      Update();
                                                                      tabu[m[1].City1][m[1].City2]=tabu_list_size;
                                                        }
                                          }
                            }
              }
}
end.
```

Fig. 2: Pseudo code of improved TS

modified RTS algorithm to achieve a good balance between intensification and diversification.

Conventional TS: In conventional TS, randomly generate its initial solution and all possible neighbourhood solutions are then started with the 2-opt switch procedures. A new solution is generated by a move which swaps two cities at the current solution. These search processes continue until stopping criteria are met e.g., when the search process has reached the specified number of iterations.

Improved Tabu Search (ITS): The distinct feature of the ITS algorithm is the initial solution generation to promise its good performance[31]. An initial solution chosen from a local optimal configuration will generate a high quality solution. Lim *et al.*[24] showed that a better performance could be achieved when combining the SA and TS algorithms.

Figure 2 shows the algorithm of ITS. At the beginning, the SA algorithm have been used to generate an initial solution for the ITS algorithm. After that, its neighborhood was explored to get the best neighborhood solution as the current solution using the 2-opt switch procedures. Next, TS algorithm was performing to search the best neighborhood solution as the new current solution for the next iteration. The search process continues until stopping criteria are met. In ITS, the memory on the search process is do not preserve and the tabu list size is always static which is set based on the number of cities of a considered problem.

Modified RTS: The problem of parameter tuning in TS is recovered in RTS which was initially introduced by Battiti and Tecchiolli[32]. Basically, RTS attempts to remedy the difficulty in choosing an appropriate tabu list size without prior knowledge about the search space structure. The RTS algorithm maintains the basic steps of TS except that its tabu list size is adaptive to a considered problem and the current solution of the search process. Through this appropriate parameter values, a good balance between intensification and diversification can be achieved without requiring a lot of prior experience or appropriate parameter values of the problem. Thus, RTS is one the reactive search methods employing two mechanisms i.e., feedback schemes and escape strategies.

Based on the RTS algorithm, Lim *et al.*[25] developed the modified RTS algorithm which is slightly difference with the RTS algorithm in terms of the reactive mechanism. The algoritm is shown in Fig. 3. In Battiti and Tecchiolli's algorithm[32], its tabu list size depends on the repetition of configuration. However, in modified RTS algorithm, the tabu list size depends on the number of iterations when the solutions do not override the aspiration level. The judgment was made since the repetition of configurations in TSPs is less significant as the number of cities increases. Thus, the RTS algorithm is not so effective in solving TSPs with a large number of cities.

Generally, the procedures of the modified RTS and ITS are the same except their tabu list sizes. In modified RTS, an initial tabu list size was generating based on the number of available cities and keep the memory on the search process. The algorithm then searches the best neighborhood solution as the new current solution for the next iteration and keeps updating the search process. The tabu list size is increased by 1 if the solutions are not improved for a specified number of iterations and reset back to the initial tabu list size if it achieves a specified value of the tabu list size. The search process continues until stopping criteria are met.

RESULTS AND DISCUSSION

Two experiments were conducted to evaluate and compare the four heuristic methods. The first experiment studied the behavior of SA's schedules to select its suitable schedule for the second experiment. Five dataset will be used to evaluate the performance of the proposed algorithms. These five chosen benchmark problems of symmetric TSPs are *hagaregn*20, *wi*29, *dj*38, *eil*51 and *eil*76 ranging in the size of cities from 20-76. These data sets were obtained through three different open access websites. The *hagaregn*20 was generated using hagaregn applet (http://hagaregn.org.uk/npsudoku/data/demo20.txt). Meanwhile, *wi*29 and *dj*38 are the TSP instances obtained from the World TSP (http://www.tsp.gatech.edu/world/) representing 29 cities in Western Sahara, North Africa and 38 cities in Republic of Djibouti, Horn of Africa, respectively. The other data sets were taken from the TSPLIB library (http://www2.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/). The locations of the cities in the data sets are displayed in node coordinates. The distance between two cities is computed using the Euclidean distance Eq. 1 which is then round off to integer numbers:

$$d = \sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2} \quad (1)$$

First experiment: The SA algorithm on a 20-city problem, *hagaregn20* was implemented to assist us choose relevant parameter values for SA. Different combinations of annealing schedules were used to select the suitable cooling rate and initial temperature.

Six different cooling rates i.e., 0.5, 0.8, 0.9, 0.99, 0.999 and 0.9999 and three different initial temperatures i.e., 10000, 100000 and 1000000, which formed eighteen combinations of annealing schedules were used in this experiment. Each combination was solved by a number of independent runs each of which consisted of fifty trials. For each trial, the minimum distance was recorded.

The average distance results for the tour from the fifty trials are summarized in Table 1. The data in Table 1 were run as a randomized complete block design to determine the significant difference between the cooling rate and initial temperature. The two-way analysis of variance (ANOVA) was performed using IBM SPSS Statistics 19. The outputs are shown in Table 2 and 3.

Table 3 shows that different cooling rates can affect the performance of SA since there is a significant difference among the five different cooling rates. However, there is no significant difference in performance of SA with different initial temperatures. Therefore, a reasonable value, 10000, as

Table 1: Average distance obtained from 50 trials

	Initial temperature		
Cooling rate	10,000	100,000	1,000,000
0.5000	1765.06	1736.18	1759.88
0.8000	1743.32	1732.70	1736.48
0.9000	1748.78	1761.14	1746.38
0.9900	1692.32	1731.36	1706.76
0.9990	1623.00	1634.36	1627.86
0.9999	1560.60	1560.54	1566.34

```
begin
    read input data;
    used SA to generate an initial solution S;
    current_d=TotalDistance(s);
    best_d=current_d;
    tabu[i][j]=0;
    noimprove=0;
    for(iterations=1;iterations<=max_iteration;iterations++)
 {
            find all possible moves;
            k=1;
            if(m[1].Distance<best_d)
            {
                    current_d=m[1].Distance;
                    best_d=m[1].Distance;
                    best_iteration=iterations;
                    noimprove=0;
                    Update();
                    tabu[m[1].City1][m[1].City2]=tabu_list_size;
 }
            else
            {
                    i=1;
                    while (i<=k-1)
                    {
                            if(tabu[m[i].City1][m[i].City2]==0)
                            {
                                    current_d=m[i].Distance;
                                    Update();
                                    tabu[m[i].City1][m[i].City2]=tabu_list_size;
                                    break;
                            }
                            i++;
                            if (i==k-1)
                            {
                                    if(tabu[m[i].City1][m[i].City2]!=0)
                                    {
                                            current_d=m[1].Distance;
                                            Update();
                                            tabu[m[1].City1][m[1].City2]=tabu_list_size;
                                    }
                            }
                    }
                    noimprove+=1;
            }
            if(noimprove>=25)
            {
                    tabu_list_size+=1;
            }

            if (noimprove>=50||tabu_list_size==ceil ((double)0.75*nc))
            {
                    tabu_list_size=ceil ((double)0.25*nc);
                    noimprove=0;
            }
 }
end.
```

Fig. 3: Pseudo code of modified RTS

the initial temperature in the SA algorithm is selected. The results from Table 3 suggested that a slow cooling rate of β = 0.9999 is the best performance among the five different cooling rates. Thus, these two parameters were

Table 2: Two-way ANOVA output for parameter selection in SA

Source	Type III sum of squares	df	Mean square	F	Sig.
Corrected model	91921.813[a]	7	13131.688	88.783	0.000
Intercept	51453952.276	1	51453952.276	347877.471	0.000
Initial_temperature	44.960	2	22.480	0.152	0.861
Cooling_rate	91876.853	5	18375.371	124.235	0.000
Error	1479.083	10	147.908		
Total	51547353.172	18			
Corrected total	93400.896	17			

a: R squared = 0.984 (Adjusted R squared = 0.973) and univariate analysis of variance, tests of between-subjects effects dependent variable: Average_dist, df: Degree of freedom

Table 3: *Post hoc* test output for six different cooling rates

					Confidence interval (95%)	
(I) Cooling_rate	(J) Cooling_rate	Mean difference (I-J)	Standard error	Sig.	Lower bound	Upper bound
0.5000	0.8000	16.2067	9.93003	0.134	-5.9188	38.3322
	0.9000	1.6067	9.93003	0.875	-20.5188	23.7322
	0.9900	43.5600*	9.93003	0.001	21.4345	65.6855
	0.9990	125.3000*	9.93003	0.000	103.1745	147.4255
	0.9999	191.2133*	9.93003	0.000	169.0878	213.3388
0.8000	0.9000	-14.6000	9.93003	0.172	-36.7255	7.5255
	0.9900	27.3533*	9.93003	0.020	5.2278	49.4788
	0.9990	109.0933*	9.93003	0.000	86.9678	131.2188
	0.9999	175.0067*	9.93003	0.000	152.8812	197.1322
0.9000	0.9900	41.9533*	9.93003	0.002	19.8278	64.0788
	0.9990	123.6933*	9.93003	0.000	101.5678	145.8188
	0.9999	189.6067*	9.93003	0.000	167.4812	211.7322
0.9900	0.9990	81.7400*	9.93003	0.000	59.6145	103.8655
	0.9999	147.6533*	9.93003	0.000	125.5278	169.7788
0.9990	0.9999	65.9133*	9.93003	0.000	43.7878	88.0388

Based on observed means, the error term is mean square (Error) = 147.908 and multiple comparisons average_dist (LSD)

Table 4: Parameters for the SA algorithm

Parameters	SA
Initial solution	Randomly generate
Neighborhood structure	Swaps two cities that is 2-opt switch procedure
Cooling rate	0.9999
Initial temperature	10000
Stopping criteria	Maximum iteration: 1000000

used in evaluating the performance of SA. The parameter values for SA are summarized in Table 4.

Second experiment: The SA, TS, ITS and modified RTS algorithms using Microsoft Visual C^{++} were executed[33]. These algorithms were then tested on five chosen benchmark problems of symmetric TSPs which ran on an Intel® Core^{TM} i3 M390@2.67 GHz CPU. Each benchmark was solved using a number of independent runs; each of which consists of thirty trials. For each trial, a tour was determined by the four variant heuristics algorithms. The parameter settings for the SA algorithm were determined through the empirical testing and statistical analyses from the first experiment as shown in Table 4. Meanwhile, the parameter settings for the TS algorithm are shown in Table 5. For each trial, the minimum distance of the tour and its computational time were recorded.

For each algorithm, the relative differences, RDbs and RDav, to access its performance will be computed. The RDbs and RDav are computed as follows:

$$\text{RDbs} = \frac{\text{Best found solution} - \text{optimum solution}}{\text{Optimum solution}}$$

$$\text{RDav} = \frac{\text{Average solution} - \text{optimum solution}}{\text{Optimum solution}}$$

The relative differences were chosen as performance indexes (in percentage) to compare the performances of the four algorithms. A smaller value of the index indicates better algorithm performance, where the performance index of 0 is the optimal performance. The results of the relative differences for the problems are shown in Table 6 and 7. The Opt column shows the benchmark solutions as reported in the literature.

Based on the descriptive statistics in Table 6 and 7, all the performance indexes of the modified RTS algorithm

Table 5: Parameter settings for the TS algorithm

	TS algorithm		
Parameters	Conventional TS	ITS	Modified TS
Initial solution	Randomly generated	Generated using the SA algorithm Parameters for SA: • Initial temperature: 10000 • Cooling rate: 0.9999 • Maximum iteration: 1000000	Same as ITS
Neighborhood structure	2-opt switch procedure by swapping two cities	Same as conventional TS	Same as conventional TS
Attribute	Both swapped cities are tabued	Same as conventional TS	Same as conventional TS
Tabu list	Decreasing tabu tenure by 1 after performing one iteration	Same as conventional TS	Same as conventional TS
Tabu restriction	TAB (i) = |Ts|	Same as conventional TS	Same as conventional TS
Selection strategy	Forbidding strategy: TAB (i)>0 Freeing strategy: TAB (i)≤0	Same as conventional TS	Same as conventional TS
Tabu list size	• Static • 25% of the number of cities[35]	Same as conventional TS	• Dynamic • Initially set as 25% of the number of cities • Increased by 1 if 25 iterations pass without finding a new best distance and continues to increase by 1 until it satisfies following conditions • Reset to initial if a new solution is found or reaches 75% of the number of cities
Aspiration criteria	Two aspiration criteria: • Aspiration by objective current distance <aspiration level • Aspiration by default that is free the least distance move	Same as conventional TS	Same as conventional TS
Stopping criteria	Maximum iteration: 1000	Same as conventional TS	Same as conventional TS

Table 6: Results of best relative difference (%) for TSP

Data set	Opt.	SA	Conventional TS	ITS	Modified RTS
*hagaregn*20	1508	0.00	0.07	0.00	0.00
*wi*29	27603	0.07	0.00	0.00	0.00
*dj*38	6656	13.76	17.08	3.34	0.00
*eil*51	426	8.92	5.16	4.69	2.58
*eil*76	538	11.15	10.22	8.36	6.51

Table 7: Results of average relative difference (%) for TSP

Data set	Opt.	SA	Conventional TS	ITS	Modified RTS
*hagaregn*20	1508	4.01	3.36	1.46	0.30
*wi*29	27603	6.58	16.42	3.77	1.14
*dj*38	6656	24.48	31.32	21.41	11.86
*eil*51	426	16.89	12.06	8.90	6.88
*eil*76	538	15.86	16.45	12.53	10.11

Table 8: Results of assumption tests

Data set	Algorithm	Shapiro-Wilk (p-value)	Levene test (p-value)
*hagaregn*20	Conventional TS	0.089	-
	ITS	0.000	
	Modified RTS	0.000	
*wi*29	Conventional TS	0.044	
	ITS	0.035	
	Modified RTS	0.004	
*dj*38	Conventional TS	0.113	
	ITS	0.932	
	Modified RTS	0.004	
*eil*51	Conventional TS	0.110	
	ITS	0.017	
	Modified RTS	0.457	
*eil*76	Conventional TS	0.062	0.000
	ITS	0.089	
	Modified RTS	0.122	

for both RDav and RDbs were relatively smaller than the performance indexes of the other algorithms. This shows that the modified RTS is the best algorithm for solving TSPs in terms of the solution quality. Apart from the descriptive statistics, an inferential statistical test using IBM SPSS Statistics19 also have been employed to discover any significant differences among the performances of the three TS algorithms i.e., conventional TS, ITS and modified RTS. For each data set, there are two kinds of independent sample tests based on the assumptions of the data which are normality and homogeneity of variance. If both of the assumptions are met, the parametric test i.e., one-way analysis of variance (ANOVA) will be used. Otherwise, if either one assumption is violated, the nonparametric test i.e., the Kruskal Wallis test should be used.

For the conventional TS, ITS and modified RTS algorithms, the Shapiro-Wilk normality test using thirty trials for each data set was conducted to show the results of the assumption tests in Table 8. Based on Table 8, only the 5th TSP data set i.e., *eil*76 fulfilled the assumption of normality since the probability value (p-value) for the other three algorithms were greater than 0.05. However, the variances among the TS algorithms were not constant for the *eil*76 problem through the levene test. Therefore, the assumptions of the data have been violated for all of the data sets. Thus, Kruskal Wallis test was used in the further data analysis. The results of the Kruskal Wallis test are shown in Table 9.

Table 9 presents that the performance indexes among the three variants of the TS algorithms i.e., conventional TS, ITS and modified RTS differ significantly for each data set since their the p-values were small and less than 0.05. The Kruskal Wallis test could not carry out to judge which TS algorithm is much efficiency. Thus, the pairwise comparisons were used to overcome this problem. The results of pairwise comparison are shown in Fig. 4.

Table 9: Results of Kruskal Wallis test

Data set	p-value
*hagaregn*20	0.000
*wi*29	0.000
*dj*38	0.000
*eil*51	0.000
*eil*76	0.000

Based on the inferential statistics in Fig. 4, the modified RTS is much better than other variant of TS algorithms for each tested data set since all the mean rank of the modified RTS is significantly lower than the other TS algorithms. Apart from the solution quality, the performance of the algorithms was also evaluated based on the computational time needed to obtain the solution. The average computational time (sec) for each algorithm from the thirty trials is summarized in Table 10.

The computational time for each algorithm is not much different when solving a problem with a small sample size (<30 cities). The SA needs less than 20 sec to solve the problem with 76 cities, while the variant of TS algorithms

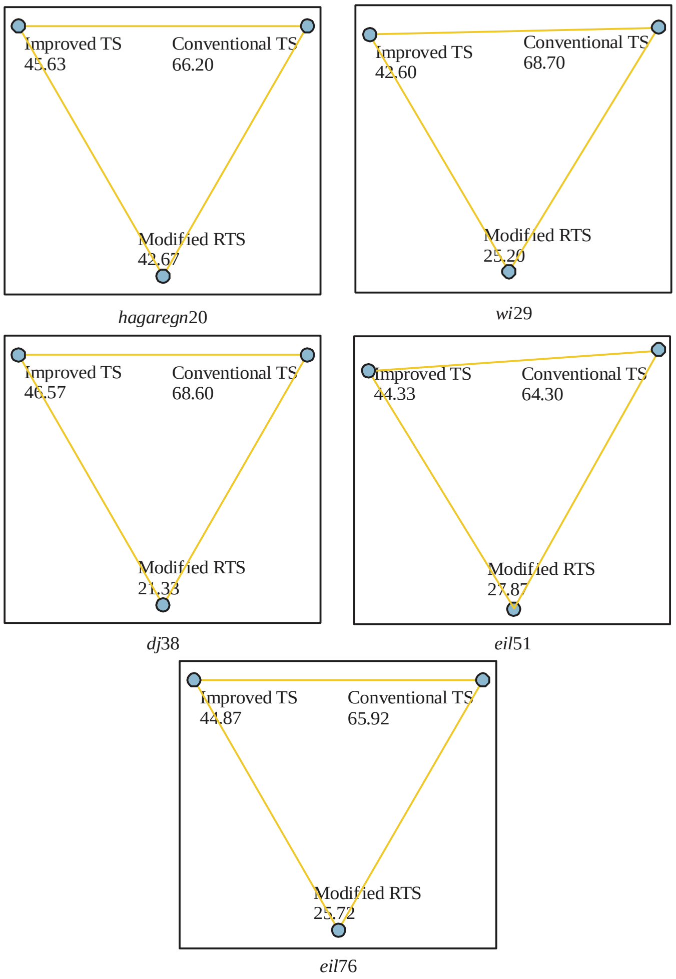

Fig. 4: Results of pairwise comparison for the TS algorithm

Table 10: Average computational time in seconds for SA and TS algorithms

Problem	SA	Conventional TS	ITS	Modified RTS
*hagaregn*20	4.73	3.17	8.80	12.33
*wi*29	6.40	17.87	27.73	33.50
*dj*38	8.27	62.03	80.93	85.73
*eil*51	11.40	254.03	250.37	271.40
*eil*76	16.67	1915.87	2195.43	1796.27

Table 11: Comparison of the modified RTS with other various techniques to solve TSP

Problem	Modified RTS	PSO[35]	BEA[36]
*hagaregn*20	1513 (1508)	NA	NA
*wi*29	27917 (27603)	NA	(28670)
*dj*38	7445 (6556)	NA	(7555)
*eil*51	455 (437)	473.34 (443)	(472)
*eil*76	592 (573)	626.94 (566)	(610)

need about 32 min to solve the same problem. However, when the size increases, the computational time for the variants of TS algorithms dramatically increases. The SA algorithm needs less time to obtain the tour compared to the variants of TS algorithms.

Table 11 shows the comparison of the proposed method, modified RTS, with the methods that are applied by various researchers[35,36] to solve TSP. Since, the modified RTS algorithm provides a good solution quality in this study. Therefore, the modified RTS is used to do the comparison with other methods that done by various researches. The comparison methods are Particle Swarm Optimization (PSO) and Basic Evolutionary Approach (BEA). The best and average computational results are recorded. Different number of independent runs is done by various researchers to obtain the average result while the best result is given in the brackets. From Table 11, the modified RTS can find better average results than the average results obtained by the four recent algorithms. Furthermore, the computational time for BEA is much slower than the proposed method. The BEA needs as much as 1827 sec and 9547 to solve the problem with 29 cities and 76 cities, respectively.

CONCLUSION

This study presents four heuristic algorithms, i.e., SA, conventional TS, ITS and modified RTS to solve symmetric TSPs. The performances of the four algorithms have been compared to identify the most effective algorithm for solving TSPs. The experiments showed that the modified RTS algorithm provides a good solution quality in terms of minimizing the objective function. Even when compared to the other methods that done by various researchers also prove that the modified RTS is comparable or sometimes better. However, the modified RTS algorithm and the other TS algorithms are more time consuming in obtaining the results for the problem with a large number of cities compared to the SA algorithm. Generally, modified RTS is more effective in producing a good solution quality while SA may be used to obtain instant solutions. As a conclusion, the algorithms proposed in this study should be considered in solving symmetric travelling salesman problems.

To enhance the algorithms, some considerations are suggested. Firstly, dynamic stopping criteria could be applied to increase the efficiency of the algorithms. Next, different move procedures such as 3-opt or multi-opt switch procedures can be used to construct the neighborhood structure. This study can expedite the search process and decrease the computational time. Last but not least, diversification such as multi-start may be applied to improve the quality of solution.

REFERENCES

1. Geng, X., Z. Chen, W. Yang, D. Shi and K. Zhao, 2011. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. Applied Soft Comput. J., 11: 3680-3689.
2. Applegate, D.L., R.E. Bixby, V. Chvatal and W.J. Cook, 2007. The Traveling Salesman Problem: A Computational Study. Princeton University Press, New Jersey.
3. Reinelt, G., 1994. The Traveling Salesman: Computational Solutions for TSP Applications. Springer-Verlag, Berling, Germany, ISBN-13: 9783540583349, Pages: 223.
4. Baldacci, R., A. Mingozzi and R. Roberti, 2012. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. Eur. J. Operat. Res., 218: 1-6.
5. Climer, S. and W. Zhang, 2006. Rearrangement clustering: Pitfalls, remedies and applications. J. Mach. Learn. Res., 7: 919-943.
6. Theys, C., O. Braysy, W. Dullaert and B. Raa, 2010. Using a TSP heuristic for routing order pickers in warehouses. Eur. J. Operat. Res., 200: 755-763.
7. Matusiak, M., R. de Koster, L. Kroon and J. Saarinen, 2014. A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. Eur. J. Operat. Res., 236: 968-977.
8. Jolai, F. and A. Ghanbari, 2010. Integrating data transformation techniques with Hopfield neural networks for solving travelling salesman problem. Expert Syst. Applic., 37: 5331-5335.
9. Chen, S.M. and C.Y. Chien, 2011. Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. Expert Syst. Applic., 38: 14439-14450.

10. Liu, F. and G. Zeng, 2009. Study of genetic algorithm with reinforcement learning to solve the TSP. Expert Syst. Applic., 36: 6995-7001.
11. Tan, L. and X. Liu, 2014. Advanced Backtracking Genetic Algorithm for TSP. In: Foundations of Intelligent Systems, Wen, Z. and T. Li (Eds.). Springer, Berlin, Heidelberg, ISBN: 978-3-642-54923-6, pp: 1025-1031.
12. Chang, P.C., W.H. Huang and C.J. Ting, 2010. Dynamic diversity control in genetic algorithm for mining unsearched solution space in TSP problems. Expert Syst. Applic., 37: 1863-1878.
13. Hung, Y.F. and W.C. Chen, 2011. A heterogeneous cooperative parallel search of branch-and-bound method and tabu search algorithm. J. Global Optimiz., 51: 133-148.
14. Basu, S., 2012. Neighborhood reduction strategy for tabu search implementation in asymmetric traveling salesman problem. OPSEARCH, 49: 400-412.
15. Anbuudayasankar, S.P., K. Ganesh and S. Mohapatra, 2014. Survey of Methodologies for TSP and VRP. In: Models for Practical Routing Problems in Logistics, Anbuudayasankar, S.P., K. Ganesh and S. Mohapatra (Eds.). Springer International Publishing, Switzerland, ISBN: 978-3-319-05034-8, pp: 11-42.
16. Youssef, H., S.M. Sait and H. Adiche, 2001. Evolutionary algorithms, simulated annealing and tabu search: A comparative study. Eng. Applic. Artif. Intell., 14: 167-181.
17. Adewole, A.P., K. Otubamowo and T.O. Egunjobi, 2012. A comparative study of simulated annealing and genetic algorithm for solving the travelling salesman problem. Int. J. Applied Inform. Syst., 4: 6-12.
18. Arostegui, Jr. M.A., S.N. Kadipasaoglu and B.M. Khumawala, 2006. An empirical comparison of tabu search, simulated annealing and genetic algorithms for facilities location problems. Int. J. Prod. Econ., 103: 742-754.
19. Sinclair, M., 1993. Comparison of the performance of modern heuristics for combinatorial optimization on real data. Comput. Operat. Res., 20: 687-695.
20. Battiti, R. and G. Tecchiolli, 1994. Simulated annealing and Tabu search in the long run: A comparison on QAP tasks. Comput. Math. Applic., 28: 1-8.
21. Hussin, M.S. and T. Stutzle, 2014. Tabu search vs. simulated annealing as a function of the size of quadratic assignment problem instances. Comput. Operat. Res., 43: 286-291.
22. Chiang, W.C. and C. Chiang, 1998. Intelligent local search strategies for solving facility layout problems with the quadratic assignment problem formulation. Eur. J. Operat. Res., 106: 457-488.
23. Fukuyama, Y., 2000. Reactive tabu search for distribution load transfer operation. Proceedings of the IEEE Power Engineering Society Winter Meeting, Volume 2, January 23-27, 2000, Singapore, pp: 1301-1306.
24. Lim, Y.F., P.Y. Hong, R. Ramli and R. Khalid, 2011. An improved tabu search for solving symmetric traveling salesman problems. Proceedings of the IEEE Colloquium on Humanities, Science and Engineering, December 5-6, 2011, Penang, Malaysia, pp: 861-864.
25. Lim, Y.F., P.Y. Hong, R. Ramli and R. Khalid, 2013. Modified reactive tabu search for the symmetric traveling salesman problems. AIP Conf. Proc., 1557: 505-509.
26. Hong, P.Y., Y.F. Lim, R. Ramli and R. Khalid, 2013. Simulated annealing with probabilistic analysis for solving traveling salesman problems. AIP Conf. Proc., 1557: 515-519.
27. Kirkpatrick, S., C.D. Gelatt Jr. and M.P. Vecchi, 1983. Optimization by simulated annealing. Science, 220: 671-680.
28. Glover, F., 1989. Tabu search-Part I. ORSA J. Comput., 1: 190-206.
29. Castellani, U., A. Fusiello, R. Gherardi and V. Murino, 2007. Automatic selection of MRF control parameters by reactive tabu search. Image Vision Comput., 25: 1824-1832.
30. Liu, Y., S. Xiong and H. Liu, 2009. Hybrid simulated annealing algorithm based on adaptive cooling schedule for TSP. Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation, June 12-14, 2009, Shanghai, China, pp: 895-898.
31. Da Silva, L.G.W., R.A.F. Pereira, J.R. Abbad and J.R.S. Mantovani, 2008. Optimised placement of control and protective devices in electric distribution systems through reactive tabu search algorithm. Electric Power Syst. Res., 78: 372-381.
32. Battiti, R. and G. Tecchiolli, 1994. The reactive tabu search. ORSA J. Comput., 6: 126-140.
33. Templeman, J., 2013. Microsoft Visual C++/CLI Step by Step (Step by Step Developer). 1st Edn., Microsoft Press, USA., ISBN-13: 978-0735675179, Pages: 540.
34. Tsubakitani, S. and J.R. Evans, 1998. Optimizing tabu list size for the traveling salesman problem. Comput. Operat. Res., 25: 91-97.
35. Zhong, W.L., J. Zhang and W.N. Chen, 2007. A novel discrete particle swarm optimization to solve traveling salesman problem. Proceedings of the IEEE Congress on Evolutionary Computation, September 25-28, 2007, Singapore, pp: 3283-3287.
36. De Sao Jose, D.R. and M.G. Hernandez, 2015. Basic evolutionary approach to the traveling salesman problem. U. Porto J. Eng., 1: 30-38.