ISSN 1996-3343

Asian Journal of Applied Sciences

http://knowledgiascientific.com

Knowledgia SCIENTIFIC
A Place to Publish Outstanding Research

Asian Journal of Applied Sciences 7 (7): 662-670, 2014
ISSN 1996-3343 / DOI: 10.3923/ajaps.2014.662.670

Comparative Study of Meta-Heuristic Approaches for Solving Traveling Salesman Problems

Anas Arram, Masri Ayob and Mohammad Zakree
Data Mining and Optimization Research Group, Center for Artificial Intelligence, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, 43600, Selangor, Malaysia

Corresponding Author: Anas Arram, Data Mining and Optimization Research Group, Center for Artificial Intelligence, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, 43600, Selangor, Malaysia

ABSTRACT

In this study, we compare three meta-heuristics approaches: hill-climbing, simulated annealing and late acceptance hill-climbing algorithm, for solving traveling salesman problem. All algorithms were tested on seven traveling salesman problem instances listed in LIBTSP datasets. Each algorithm was implemented and evaluated independently using its parameter. Performance were compared between all algorithms and evaluated in terms of objective function that calculated to find the optimal rout within several selected routes. The main conclusion is that late acceptance hill-climbing algorithm performs better than simulated annealing and simple hill-climbing as far as solution quality is concerned whilst simple hill-climbing gives the worst performance according to its poorness in global search.

Key words: Hill-climbing, late acceptance hill-climbing, simulated annealing, traveling salesman problem

INTRODUCTION

The Traveling Salesman Problem (TSP) is one of the most widely studied combinatorial optimization problem. Given a completely connected graph, G = (N, E) where, N is the nodes, E is the set of directed edges and distances between nodes are euclidean, we are required to find the shortest tour that passes through each node in N exactly once.

Since 1950s, many approaches have been applied to solve TSP. In general, these approaches were classified into two categories; approximation algorithm and exact algorithm (Laporte, 1992). Exact algorithms use mathematical models, whilst approximation algorithms find solution using heuristic/meta-heuristic and iterative improvements. Example of exact algorithms are; Branch and Bound, Lagrangian Relaxation and Integer Programing (Rosenkrantz *et al*., 1977). In approximation algorithms, there are classified as constructive and improvement heuristics. The constructive heuristics are for example Nearest Neighbourhood, Greedy Heuristic and Insertion Heuristic (Rosenkrantz *et al*., 1977). Whilst, example of improvement heuristics are: K-opt, Line-Kernighan Heuristic, Simulated Annealing, Tabu Search (Tsubakitani and Evans, 1998), Evolutionary Algorithms (Freisleben and Merz, 1996), Ant colony Optimization (Dorigo and Gambardella, 1997) and Bee System (Wong *et al*., 2010). Most meta-heuristics algorithms are designed to escape from local minima by accepting worse solutions and give the possibility to find the best solution in global search. Therefore, in this study, we conduct a comparative study of three meta-heuristic approaches for solving TSP. We investigate simulated annealing, hill climbing and

late acceptance hill climbing for solving TSP. Each algorithm is applied to seven different instances taken from LIBTSP dataset (Reinelt, 1991). The performance of each algorithm was evaluated in terms of the quality of the obtained solution. The purpose of this study is to investigate which algorithm is better for solving TSP.

PROBLEM DESCRIPTIONS

Traveling salesman problem is considered as a discrete optimization problem and classified as a NP-hard (Laporte, 1992). Traveling salesman problem via graph notation can be represented as follows:

- Giving G = (N, E) as a graph
- N is a set of n cites, N = {N_1,..., N_n}
- E is a set of directed edges, E = {(i, j):i, jϵN}

E is usually depicts the distance (or cost) between cities, which is defined as D = ($d_{i,j}$). In case of $d_{i,j} = d_{j,i}$, the problem is considered as a symmetric TSP. Otherwise, it becomes an asymmetric TSP. The main objective in TSP is to find the optimum swapping of set N that minimize Eq. 1:

$$Z = \sum_{i=1}^{n}\sum_{j=1}^{n} x(i, j)\, d(i, j) \tag{1}$$

While:

- x(i,j)ϵ{0,1}, x(i,j) = 1, if we go from city i to city j
- d(i,j) is the euclidean distance

In particular, this is the case for which the vertices are points $P_i = (X_i, Y_i)$ in the graph and Eq. 2 defines the euclidean distance:

$$d(i, j) = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \tag{2}$$

This section reviews some meta-heuristic algorithms that have been applied to solve traveling salesman problems. These include simulated annealing (Aarts *et al.*, 1988), Ant colony optimization (Dorigo and Gambardella, 1997).

Aarts *et al.* (1988) presented a quantitative study of typical behaviour of SA based on a cooling schedule by applying the algorithm to a 100 city TSP. The study analyzed the expectation and the variance of the cost as a function of the control parameter of the cooling schedule. The performance analyzed for which estimates have been obtained on the expectation of the average final results obtained by SA as a function of the distance parameter which defines the decrease of the control parameter.

Dorigo and Gambardella (1997) presented the capability of Ant Colony Optimization algorithm for solving TSP. They used an information stored in the form of a pheromone path dropped on the edges of the TSP graph to continuously generate shortest possible tours. The experiments showed that this algorithm can perform well to achieve good solution to both symmetric and asymmetric instances of the TSP. From the experiments presented, local search (such as 2-opt and 3-opt) was

embedded in the ACO to enhance the results generated from the algorithm. On the other hand, the global trial will be updated once each ant has been taken to its local optimum before global trail updating is performed. Then, the algorithm could efficiently improve the performance to achieve good quality solution by performing an efficient parallelization. The best way to achieve parallelization is to distribute the ants into different processors which can solve the TSP on each processor by specific number of ants and exchange the best founded tour among the processors.

Tsubakitani and Evans (1998) studied the size of tabu list when it applied to symmetric TSP with a short term memory. They used two types of neighbourhood operators: 2-opt and 3-opt. Firstly, they tested problems of 20, 50 and 150 nodes, which randomly generated. The best size of tabu list was identified for each combination of problem size and type of neighbourhood used within a given time limit. The study depicted that the smaller neighbourhoods require a huge size of tabu list to be effective and that good tabu list sizes are smaller than generally believed.

Pepper *et al.* (2002) conducted a comprehensive computational study of 11 annealing based heuristic for solving the TSP. These algorithms consist demon and non-demon algorithms. Demon algorithm is a variant of SA aim to speed up SA by reducing computation time per trial without losing the quality of solution. The non-demon algorithms are; simulated annealing, threshold accepting and record to record travel and demon algorithms are; bounded demon algorithm, randomized bounded demon algorithm, annealing demon algorithm, randomized annealing demon algorithm, annealed bounded demon algorithm, randomized annealing bounded demon algorithm, hybrid annealed demon algorithm and hybrid annealed bounded demon algorithm. Each algorithm applied to 29 TSP instances which taken from TSPLIB online website (Reinelt, 1991). The results showed that SA outperformed the best result as a non-demon algorithms in teams overall accuracy and running time. The other demon algorithms were nearly close to SA and the randomized annealing demon algorithm was 10% faster than SA.

Liu (2007) developed hybrid scatter search by integrating nearest neighbour rule, threshold accepting and the edge recombination crossover into scatter search structure to solve TSP. The test problems which presented from (Tang and Miller-Hooks, 2006) study were used in the experiments. Results showed that by incorporating threshold accepting with scatter search can effectively improve computation efficiency while solving TSP.

Wong *et al.* (2010) presented a Bee Colony Optimization (BCO) algorithm for solving symmetric TSP. The BCO is naturally inspired from collective intelligent of bee foraging behavior. The study integrated BCO model with 2-opt heuristic to significantly improve the performance of BCO. The model was applied to solve 84 TSP benchmark instances. It was able to achieve an average solution quality of 0.31% from known optimum solution.

Ouaarab *et al.* (2014) proposed a discrete Cuckoo Search (CS) algorithm to solve TSP. Cuckoo search algorithm was inspired by the breeding behavior of cuckoos. The idea of this algorithm is to solve continues optimization problem. They intended to improve cuckoo search by restructuring its population to be able to solve combinatorial problems as well as continues problems. The proposed method was tested on symmetric TSP benchmarks which collected from TSPLIB library (Reinelt, 1991). Results showed that the proposed discrete cuckoo search method performs better than Genetic Algorithm and Particle Swarm Optimization.

LOCAL SEARCH ALGORITHMS

Local search is an improvement heuristic method for solving computationally hard optimization problem. Local search algorithm starts from an initial S and iterates exploring the research space,

using the moves associated with the neighbour that improves the objective function (Di Gaspero, 2003). Some common local search are: WalkSAT, 2-opt, SA, HC, LAHC and Genetic Algorithm. This study focuses on hill-climbing, Late-Acceptance Hill Climbing and simulated annealing. Before applying the local, we need to generate an initial solution using a constructive heuristic. This study use nearest neighbour heuristic which starts by randomly select a starting city then visit the nearest city that has not visited yet in the tour until all cities are visited. Finally, return to the city that the algorithm starts from. In the next step, improvement process will be done using a local search algorithm. A 2-opt operator is used in the local search by replacing two links of the tour with other two links.

HILL-CLIMBING ALGORITHM

Hill-climbing is one of the earliest studies in local search optimization algorithms. It was applied by Burke and Bykov (2012). The algorithm starts with an initial solution which is considered as a current solution at the first iteration. Then the solution will be modified at each iteration using neighbourhood operators (e.g., 2-opt exchange) to generate a new candidate solution. The candidate solution is compared with the current one. If it is better then accept it and use it as a current solution for the next iteration. If it is worse then reject it and carry out the next iteration with the same current solution. The search continues until the maximum number of iteration is reached (Talbi, 2009).

This algorithm is efficient in terms of time as it performs very fast. In terms of quality of solution usually it can easily stuck in a local optimum and return with a solution that is quite far from the local optima. This is due to not accepting worse solution. Figure 1 describes the pseudo-code of hill-climbing algorithm.

LATE ACCEPTANCE HILL-CLIMBING ALGORITHM

Late Acceptance Hill-Climbing (LAHC) algorithm is an improved version of hill-climbing algorithm, was proposed by Burke and Bykov (2008). The main idea of this algorithm is to delay the comparison between current initial solution, S and candidate solution, S* (as known in greedy hill climbing), i.e., to compare the new candidate solution with the one which was the current solution several iterations ago.

LAHC is like other meta-heuristics starts from a given initial solution and iteratively improve it by comparing the new candidate solution with the current one in order to accept or reject it.

```
Generate the initial solution S
Calculate initial cost function f(s)
First iteration I = 0
Repeat
 Construct a candidate solution S*
 Calculate its cost function f(s*)
 If f(s*)≤f(s)
 Then accept the candidate (s = s*)
 Else reject the candidate (s = s)
 Increment the number of iteration (I = I+1)
Until stopping criteria satisfied
```

Fig. 1: Pseudo-code of hill-climbing algorithm (Burke and Bykov, 2012)

```
Generate the initial solution S
Calculate initial cost function f(s)
Set the length of array Lfa
For all K∈{0... Lfa-1}, f_K = f(s)
First iteration I = 0
Repeat
 Construct a candidate solution S*
 Calculate its cost function f(s*)
 v = I mod L_fa
 If f(s*)≤f_v or f(s*)≤f(s)
 Then accept the candidate (S = S*)
 Else reject the candidate (S=S)
 Insert the current cost into the fitness
 array f_v = f(s)
Increment the number of iteration (I = I+1)
Until stopping criteria satisfied
```

Fig. 2: Pseudo-code of late acceptance hill-climbing algorithm (Burke and Bykov, 2012)

Hence, to apply LAHC rule, the algorithm will create a list with fixed length to save the quality of recently visited solutions. If the quality of the new candidate solution is better than the quality value of the last element in the list then it will be accepted as the current initial solution. Then the element at the end of the list will be removed and the quality value of the newly accepted solution will be added into the beginning of the list (Burke and Bykov, 2012). Figure 2 presents the pseudo-code of LAHC algorithm (Burke and Bykov, 2012).

SIMULATED ANNEALING ALGORITHM

Simulated annealing algorithm was proposed by Kirkpatrick *et al.* (1983). The algorithm SA is based on the principles of statistical mechanics whereby the annealing process requires heating and then slowly cooling a substance to obtain a strong crystalline structure. The notion of SA is to give a probability to accept worse solution by probabilistically accept a worse neighbour. This process attempted to find possibly the best global solution. The probability P(S*) to escape from local optimum and accept state X in terms of its energy E(x) and temperature T is giving by the following Eq. 3:

$$P(S^*) = \exp\left(-\frac{f(S) - f(S^*)}{T}\right) \quad (3)$$

Where:

P(S*) = Probability to accept worse solution
f(S) = Cost function of the current solution
f(S*) = Cost function of the candidate solution
T = Temperature

Table 1 illustrates the analogy between physical system and the optimization problem (Talbi, 2009). Figure 3 presents the pseudo-code of simulated annealing algorithm (Talbi, 2009).

Set the maximum number of iteration $I = I_{max}$
Set the starting temperature $T = T_0$
Set Cooling rate.
Generate the initial solution $S = S_0$
Repeat
 Repeat
 Generate a random neighbour s*
 $\Delta E = f(s^*)-f(s)$
 If $\Delta E \leq 0$ Then s = s*/accept the neighbour solution
 Else Accept s* with probability $e^{\frac{-\Delta E}{T}}$
 Until Equilibrium condition
 Reduce temperature rate T = T* Cooling rate
 Update the best solution
Until Stopping criteria satisfied $l \leq l_{min}$

Fig. 3: Pseudo-code of simulated annealing algorithm (Talbi, 2009)

Table 1: Analogy between physical system and the optimization problem

Physical system	Optimization problem
System state	Solution
Molecular positions	Decision variables
Energy	Objective function
Ground state	Global optimum solution
Metastable state	Local optimum
Rapid quenching	Local search
Temperature	Control parameter T
Careful annealing	Simulated annealing

The main steps of SA implementation are as follows: first, generate a solution S to start the algorithm. At each iteration, a neighbour, S* of the current solution will be generated, using neighbourhood operator. If the quality of the new solution is better than the solution S then we select the new solution as a new solution S. Otherwise, the new solution could be selected if P>r, where $P = e^{\frac{-\Delta E}{T}}$ (Fig. 3) and r is a randomly generated number between [0, 1]. Then we apply the cooling schedule to decrease the temperature. All the procedures are repeated until the termination criterion is reached (maximum iteration). By reducing the temperature, we are potentially reach a promising solutions and unlikely to be trapped in local optima.

EXPERIMENTAL RESULTS

The algorithms have been tested against seven instances of euclidian symmetric TSP as tested in Burke and Bykov (2012), the instances were collected from TSPLIB online library. The problem range in size from 783 to 3795 nodes as used in previous studies (Burke and Bykov, 2012). Each instance presents a specific number of nods in the problem (e.g., the data in instance rat783 includes 783 nodes and instance U1060 includes 1060 nodes and so on). The proposed algorithms were implemented in Java on a PC Dell with a 2400 Ghz processor and 4 GB RAM running on windows 7.

The parameter settings for HC, LAHC and SA are listed in Table 2. The parameters of SA as suggested in Soubeiga (2003) were: the initial temperature t was 50% of the value of the initial

Table 2: Parameter settings

Parameters	Values
t	50% of the value of the initial solution
β	0.85
Lfa	5000
Iterations	10^6

Table 3: Results obtained from LAHC, HC and SA

	HC		SA		LAHC	
DataSet	Ave	Best	Ave	Best	Ave	Best
Rat783	10820	10140	10600	10019	9243	9158*
U1060	263037	256233	259683	248891	233625	231324*
Fl1400	22643	21378	22490	21117	20586	20306*
U1817	69267	67705	68809	66492	61471	60797*
D2103	98618	94862	98800	96291	88696	86951*
Pcb3038	159678	158029	153914	140130	149395	148236*
Fl3795	33297	31133	32667	31124	30841	30139*

*Best solutions

Table 4: p-values of Wilcoxon test between LAHC, HC and SA

DataSet	HC-LAHC	HC-SA	LAHC-SA
Rat783	0.000	0.196	0.000
U1060	0.000	0.005	0.000
Fl1400	0.000	0.410	0.000
U1817	0.000	0.136	0.000
D2103	0.000	0.659	0.000
Pcb3038	0.000	0.001	0.001
Fl3795	0.000	0.002	0.000

solution and the cooling rate **β** was 0.85. LAHC has only one parameter that need to be tuned that was Lfa (length of fitness array) was 5000 as suggested in Burke and Bykov (2008). The number of iterations for all algorithms was 10^6 as proposed by Burke and Bykov (2008). Each algorithm was run 31 independent runs.

Table 3 presents average solutions and the best solution (out of 31 runs) for all 7 instances. Results shows that LAHC outperformed other algorithms. Table 4 shows the p-values of Wilcoxon test. In this test, if p-value less than 0.05 ($p<0.05$) then, there is a significant difference between both methods. But if p-value less than 0.01 ($p<0.01$) then, there is a highly significant difference between both methods. Otherwise, there is no significant difference between methods.

As we can see from Table 4, all p-values of LAHC-HC and LAHC-SA are less than 0.01 which means there is a high significant difference between both methods and it's due to the effectiveness of LAHC in finding good quality solution compared to HC and SA. In case of SA-HC, p-value is less than 0.05 which mean that there is a significant difference on U1060 and Fl3795. There is a highly significant difference between SA-HC on Pcb3038 instance. These results show that LAHC could produce competitive results which able to find good quality solution on TSP.

CONCLUSION

We have presented three meta-heuristic approaches for solving symmetric TSP which are; hill climbing, late acceptance hill climbing and simulated annealing.

Based on our experimental results and analysis, LAHC emerged the best results compared to SA and HC in terms of the quality of solution. This is due to its ability to exploit and explore the search space and the less number of parameters required. SA is close to LAHC in some instances and close to HC in some other instances. Hill climbing performed badly when compared to LAHC algorithm for all instances.

ACKNOWLEDGMENT

The authors wish to thank the Universiti Kebangsaan Malaysia for supporting this study under the Industrial and Community Research Grant Scheme (INDUSTRI-2013-034).

REFERENCES

Aarts, E.H.L., J.H.M. Korst and P.J.M. van Laarhoven, 1988. A quantitative analysis of the simulated annealing algorithm: A case study for the traveling salesman problem. J. Stat. Phys., 50: 187-206.

Burke, E.K. and Y. Bykov, 2008. A late acceptance strategy in hill-climbing for exam timetabling problems. Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, August 18-22, 2008, Montreal, Canada.

Burke, E.K. and Y. Bykov, 2012. The late acceptance hill-climbing heuristic. Technical Report CSM-192, Department of Computing Science and Mathematics University of Stirling, UK., June 2012.

Di Gaspero, L., 2003. Local search techniques for scheduling problems: Algorithms and software tools. Ph.D. Thesis, Dipartimento di Matematica e Informatica-Universita degli Studi di Udine, Italy.

Dorigo, M. and L.M. Gambardella, 1997. Ant colonies for the travelling salesman problem. Biosystems, 43: 73-81.

Freisleben, B. and P. Merz, 1996. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. Proceedings of the International Conference on Evolutionary Computation, May 20-22, 1996, Nayoya, Japan, pp: 616-621.

Kirkpatrick, S., D.G. Gelatt Jr. and M.P. Vecchi, 1983. Optimization by simulated annealing. Science, 220: 671-680.

Laporte, G., 1992. The traveling salesman problem: An overview of exact and approximate algorithms. Eur. J. Oper. Res., 59: 231-247.

Liu, Y.H., 2007. A hybrid scatter search for the probabilistic traveling salesman problem. Comput. Oper. Res., 34: 2949-2963.

Ouaarab, A., B. Ahiod and X.S. Yang, 2014. Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput. Applic., 24: 1659-1669.

Pepper, J.W., B.L. Golden and E.A. Wasil, 2002. Solving the traveling salesman problem with annealing-based heuristics: A computational study. IEEE Trans. Syst. Man Cybern. Part A: Syst. Humans, 32: 72-77.

Reinelt, G., 1991. TSPLIB-A traveling salesman problem library. ORSA J. Comput., 3: 376-384.

Rosenkrantz, D.J., R.E. Stearns, P.M. Lewis II, 1977. An analysis of several heuristics for the traveling salesman problem. SIAM J. Comput., 6: 563-581.

Soubeiga, E., 2003. Development and application of hyperheuristics to personnel scheduling. Ph.D. Thesis, University of Nottingham.

Talbi, E.G., 2009. Metaheuristics: From Design to Implementation. John Wiley and Sons, New York, USA., ISBN-13: 978-0470278581, Pages: 624.

Tang, H. and E. Miller-Hooks, 2006. Interactive heuristic for practical vehicle routing problem with solution shape constraints. Transp. Res. Record: J. Transp. Res. Board, 1964: 9-18.

Tsubakitani, S. and J.R. Evans, 1998. Optimizing tabu list size for the traveling salesman problem. Comput. Oper. Res., 25: 91-97.

Wong, L.P., M.Y.H. Low and C.S. Chong, 2010. Bee colony optimization with local search for traveling salesman problem. Int. J. Artif. Intell. Tools, 19: 305-334.