Advanced Science Letters
Vol. 24(2), 1484-1487, 2016

Optimization of Travelling Salesman Problem with Precedence Constraint using Modified GA Encoding

M.F.F. Ab Rashid[1*,] M. Jusop[1], N.M.Z. Nik Mohamed[1] and F.R.M. Romlay[1]
[1]Faculty of Mechanical Engineering, Universiti Malaysia Pahang, Pekan, Malaysia
*Corresponding author. Tel.: +609-4246321; email: ffaisae@ump.edu.my

One of the challenges in combinatorial optimization is to optimize travelling salesman problem with precedence constraint (TSPPC). The optimization algorithm to deal with this problem is continuously developed and improved to enhance its performance. Genetic algorithm (GA) is one of popular algorithm used to optimize TSPPC. In this work, the Genetic algorithm is improved by using a discrete encoding instead of continuous encoding. The numerical experimental results indicated that the proposed algorithm able to search for optimal solution faster compared with original encoding.

Keywords: Travelling salesman problem, precedence constraint algorithm.

1. INTRODUCTION

Travelling salesman problem with precedence constraint (TSPPC) involves finding an optimal route for visiting a number of cities exactly once by following a set of precedence constraint[1] . Previously, different methods have been proposed for obtaining optimal solution for the TSPPC. The methods used to solve TSPPC are classified into exact and heuristic methods. Exact methods like Branch-and-Bound, dynamic programming and local search techniques always lead to the optimal solution[2] . However, they usually take sizeable time to solve the problem. Thus, it can only handle smaller size problems[3] .

Heuristic methods such as neural network, Tabu search and genetic algorithm (GA) were developed to find the near-optimal solution for larger dimension problems within a reasonable CPU time. However, these methods do not guarantee an optimal solution[4] . [5] applied the traditional GA to solve TSPPC. Later, they improved the existing algorithm using the hybrid approach[3] . They found that, the proposed algorithm generated better solution for larger size problem compared to the traditional GA.

However, it is computationally expensive to use priority factor as chromosome because when a specific string in chromosome is changed, the element inside the sequence is randomly changing. This will increase the number of generations to come out with optimal solution because of the unpredictable changes of sequence when a particular string in chromosome is changed [6] .

This paper presents an improved genetic algorithm to solve the TSPPC with optimal sequence and less number of generations. The proposed algorithm also will have faster iteration time compare to the algorithm that was proposed by [7] . The proposed algorithm directly used sequence of solution as chromosome instead of priority factor. However, by using sequence of solution as chromosome, the chances of generating infeasible chromosome is still exist. Therefore, the repair operator is required in the proposed algorithm. In this case, the repair operator is adopted from topological sort that was used by[7] .

[*]Email Address: ffaisae@ump.edu.my

2. DEVELOPMENT OF THE PROPOSED ALGORITHM

An efficient genetic algorithm (GA) was introduced by[7] as an improvement of traditional GA to solve TSPPC with better efficiency. By introducing new chromosome representation and crossover operator, Moon's algorithm had successfully generated better solution for larger size problem. The proposed algorithm is an extended study of Moon's algorithm with the purpose of generating optimal solution with less number of generations and faster iteration time.

The main idea of the proposed algorithm is to directly use sequence of solution instead of priority factor as chromosome. In GA, the chromosome selection to represent a particular problem determines performance of the algorithm. A good GA chromosome design should reduce or eliminate redundant variables from the algorithm[8] . Redundancy refers to a solution that being able to be represented by a variable, but a few variables appear in the algorithm multiple times. Multiple representations of the same solution increase the search space and slow the search.

Therefore in Moon's algorithm, the multiple representations are associated with two different set of variables that exist in algorithm. The variables refer to the priority factor as chromosome and sequence of solution as the output in this problem. By using different variables for chromosome and sequence of task, the changes in sequence of task cannot be predicted when a specific string in chromosome changed.

2.1 INITIALIZATION

For initial population, random permutation method is used to generate chromosomes. The integer from 1 to N, which is the number of node, is generated in random sequence. For example, Figure 1 shows a problem that consists of six nodes.

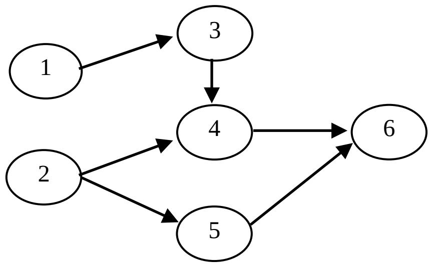


Fig. 1. An example of precedence diagram

The initial chromosome is generated randomly using integer 1 to 6 (e.g. [4, 2, 6, 1, 5, 3]). The number of chromosome depends on the size of population, P. These sequences (chromosomes) normally did not satisfy the precedence constraint. Therefore, the topological sort method with some modification on the selection method is used. Besides that, total number of generation also is included.

2.2 REPRESENTATION

The chromosome for the proposed algorithm consists of integer from 1 to N, which N is number of nodes to be visited. The number of string represents the task number. The representation stage consists of three main steps. The first step is identifying the available nodes. The available nodes means that the node without predecessor. For example, in Figure 1, the available nodes are node number 1 and 2.

Then, the second step in representation is selecting task in earlier position of chromosome. By referring to the available node (1 and 2), the node 2 is firstly found in initial chromosome [4, 2, 6, 1, 5, 3] compare to node 1. Therefore, node 2 is selected as the first string in sequence of task, seq. The third step in representation is removing edge from selected node. Since node 2 was selected, it is removed from the precedence diagram. Therefore the new available nodes are node number 1 and 5. By repeating similar steps in representation, the feasible sequence that is generated from chromosome [4, 2, 6, 1, 5, 3] is [2, 1, 5, 3, 4, 6].

2.3 EVALUATION AND SELECTION

Evaluation and selection process of chromosome is closely related to the genetic algorithms. The evaluation and selection of good chromosomes will ensure better generated chromosome for the next generation. The chromosome is evaluated using a fitness value through a fitness function. In this case, the fitness function is given by equation (1).

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1}$$

$i, j = 1,2,\ldots,n$ and $i \neq j$

In Equation 1, cij represents the traveling distance or traveling time from node i to node j according to a specific problem. The objective in Equation 1 is to minimize the total traveling distance or the total traveling time. The binary variable xij is constraint to ensure each node is entered and exited exactly once.

The roulette wheel selection is used to select parent chromosomes to be re-generated for the next chromosome. A selection chance for chromosome is depending on fitness value. For minimization problem, the chromosome with smaller fitness value has better chance to be selected and re-generated.

2.4 GENERATION OF OFFSPRING

The purpose of re-generating chromosomes is to generate new chromosome which is known as offspring. There are two operators for generating offspring which are crossover and mutation. In the proposed algorithm, two successive chromosomes are selected as parents by using roulette wheel selection method. Then, Moon crossover is applied to generate two new children. Moon crossover is a crossover operator that was introduced by[7] .

Mutation procedure which called swap mutation is then applied to the chromosome. For mutation, two strings in chromosome are selected at random. Then, the position of the selected string is swap to create new chromosome. The purpose of swapping string at random is to avoid local optimum. The procedure of the proposed algorithm is presented as follows:

Procedure: Proposed Algorithm

```
Begin
Initialization
set random permutation of sequence (x1, x2,…, xN) with N
strings, population size P and total number of generation;
      generation ← 0, population ← 0, chromosome ← 0
while generation < total number of generation do
  while population < population size do
    while chromosome < length of chromosome, N do
Representation
check and store available node without incoming edge in
available set;
select and store task in earlier position of sequence in seq;
  remove edge from selected task;
    end while
  end while
Evaluation and Selection
    Evaluate fitness value;
Roulette wheel selection, select 2 chromosomes, Pa and Pb;
Generation of Offspring
   Moon crossover;
   Mutation;
End while
End procedure
```

3. NUMERICAL EXPERIMENTS

Three numerical experiments using different size of problems were conducted to show the effectiveness of the proposed algorithm. These problems were acquired from[7] . In order to compare performance of the proposed algorithm, Moon's algorithm together with traditional GA (also known as OX algorithm) were also experimented.

The objective of these experiments was to find the optimum sequence, which provide the fastest transition time for all nodes. Parameters that were measured in these experiments are:

Optimal solution

Number of generations to come out with optimal solution

Iteration time to complete generations

Iteration time to generate optimal solution

Numerical experiments were performed on HP Compaq with Pentium CORE i5 CPU, 2.6 Gigahertz clock speed and 8 Gigabyte of RAM. The programming language for all algorithms is MATLAB Version 7.8.0.

3.1 TEST PROBLEMS

The first experiment contains six nodes and six precedence constraint as shown in Figure 2. All parameters and data are available in Moon et.al (2002). Figure 3 shows the transition time versus number of generation for the first problem. According to the graph, all algorithms achieved optimal solution, 39 seconds as achieved by Moon et.al (2002).

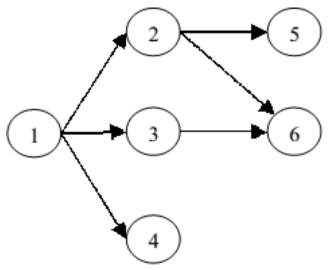

$$\begin{tikzcd}
 & 2 \arrow[r] \arrow[dr] & 5 \\
1 \arrow[ur] \arrow[r] \arrow[dr] & 3 \arrow[r] & 6 \\
 & 4 &
\end{tikzcd}$$

Fig. 2. TSP with precedence constraint [7]

The second problem deals with 20 vertices and 31 precedence constraints as shown in Figure 3. The details data of this problem is acquired in[7] .

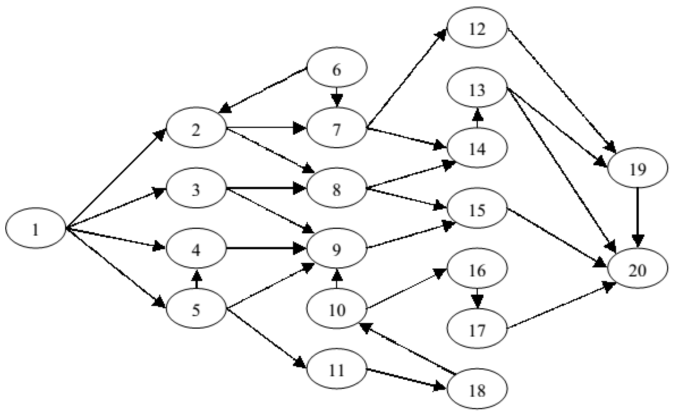


Fig. 3. TSPPC with 20 nodes and 31 precedence constraint [7]

The third problem which involves 40 tasks and 56 precedence constraints is also taken from[7] . The transition time between operations are randomly generated within 1 and 15 as proposed by[7] . The precedence diagram for this problem is presented in Figure 4.

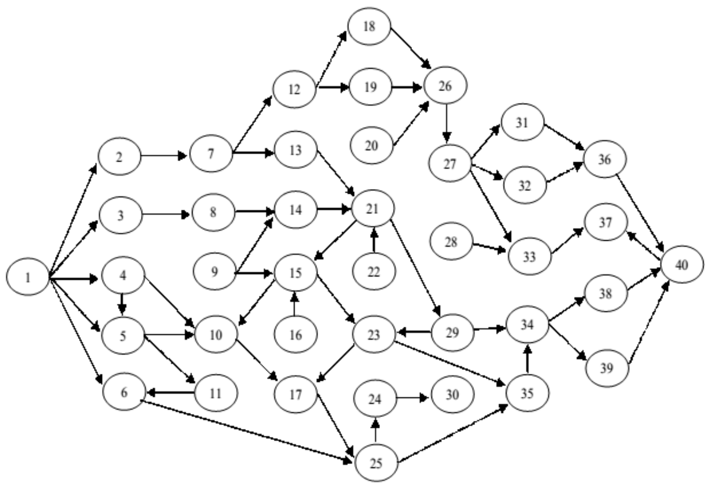


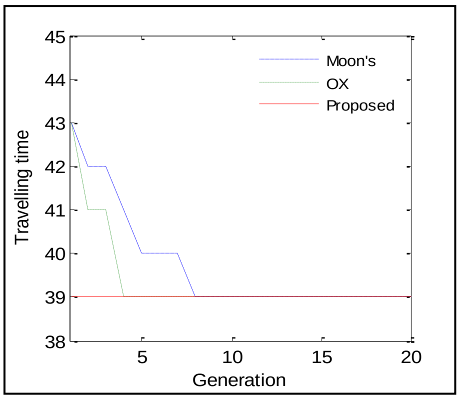
Fig. 4. TSPPC with 40 nodes and 56 precedence constraint[7]

3.2 NUMERICAL RESULTS

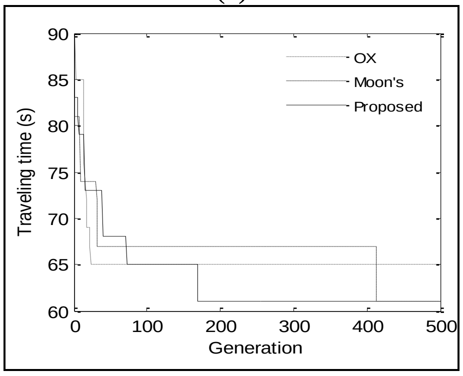
The numerical experiment results are presented in Table 1. Based on this table, the proposed algorithm consistently obtained the best known solutions as presented in[7] . However, the iteration time and time to reach optimum solution is slightly improved compared with Moon algorithm.

Table. 1. Numerical experiment results

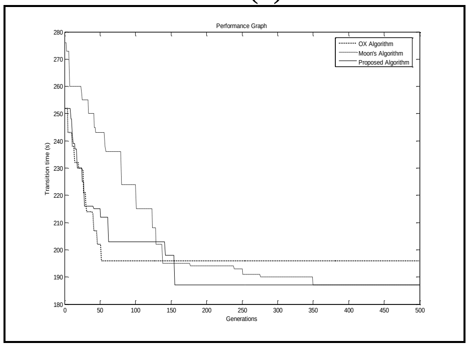
Problem No	No of task	Algorithm	Optimum travelling time (s)	Iteration time (s)	Iteration time to optimum (s)	No of generation to optimum
1	6	Moon	39	1.625	0.859	8
		OX	39	2.718	1.437	4
		Proposed	39	1.484	0.234	1
2	20	Moon	61	1090	896.1	413
		OX	65	188.3	28.2	23
		Proposed	61	222.4	82.47	170
3	40	Moon	187	30,962	21683	350
		OX	196	1619	177	52
		Proposed	187	490	1568	155



(a)



(b)



(c)

Fig. 5. Algorithms performance for the Problem 1(a), Problem 2(b) and Problem 3(c)

Figure 5 show the convergence plot for the test problems. The results show that for this particular problem, the proposed algorithm generates optimal solution with less number of generations compare to Moon's algorithm. Moon's algorithm required longer time to perform iterations. The results also indicate that the proposed algorithm is able to produce optimal solution with less generation and less time consuming for this particular problem.

4. CONCLUSIONS

Numerical experiment results show that the performance of the proposed algorithm is better than Moon's algorithm in terms of generating optimal solution with less generation of populations. For all cases, the numbers of generations to generate optimal solution are reduced within 55.7% to 87.5%. The results also indicated that the iteration time to generate optimal solution for the proposed algorithm is smaller than iteration time for Moon's algorithm. The improvement percentage of CPU iteration time is within 72.7% to 97.7%.

The numerical experiment results of traveling salesman problem with precedence constraint confirmed that the proposed algorithm is more efficient than Moon's algorithm for generating the optimal solution with less generation of populations.

ACKNOWLEDGMENTS

The authors would like to acknowledge the Malaysian Ministry of Higher Education and Universiti Malaysia Pahang for supporting this research under RDU140103 grant.

REFERENCES

[1] H.-J. Böckenhauer, T. Mömke, and M. Steinová, "Improved approximations for TSP with simple precedence constraints," *Journal of Discrete Algorithms,* vol. 21, pp. 32-40, 2013.

[2] J.-F. Cordeau, M. Dell'Amico, and M. Iori, "Branch-and-cut for the pickup and delivery traveling salesman problem with FIFO loading," *Computers & Operations Research,* vol. 37, pp. 970-980, 2010.

[3] Y. Yun, H. Chung, and C. Moon, "Hybrid genetic algorithm approach for precedence-constrained sequencing problem," *Computers & Industrial Engineering,* vol. 65, pp. 137-147, 2013.

[4] K. M. Curtin, G. Voicu, M. T. Rice, and A. Stefanidis, "A comparative analysis of traveling salesman solutions from geographic information systems," *Transactions in GIS,* vol. 18, pp. 286-301, 2014.

[5] Y. Yun and C. Moon, "Genetic algorithm approach for precedence-constrained sequencing problems," *Journal of Intelligent Manufacturing,* vol. 22, pp. 379-388, 2011.

[6] S. Mizuno, S. Iwamoto, and N. Yamaki, "Proposal of an Effective Computation Environment for the Traveling Salesman Problem Using Cloud Computing," *Journal of Advanced Mechanical Design, Systems, and Manufacturing,* vol. 6, pp. 703-714, 2012.

[7] C. Moon, J. Kim, G. Choi, and Y. Seo, "An efficient genetic algorithm for the traveling salesman problem with precedence constraints," *European Journal of Operational Research,* vol. 140, pp. 606-617, 2002.

[8] M. Albayrak and N. Allahverdi, "Development a new mutation operator to solve the Traveling Salesman Problem by aid of Genetic Algorithms," *Expert Systems with Applications,* vol. 38, pp. 1313-1320, 2011.

Received: 22 September 2016. Accepted: 18 October 2016