Journal of Scientific Research and Development 2 (13): 212-216, 2015
Available online at www.jsrad.org
ISSN 1115-7569

Optimization of traveling salesman problem with precedence constraint using particle swarm optimization

M.F.F. Ab Rashid *, N.M.Z. Nik Mohamed, A.N. Mohd Rose

Faculty of Mechanical Engineering, University Malaysia Pahang, Pekan, Pahang, Malaysia

Abstract: Traveling salesman problem with precedence constraint (TSPPC) is one of the complex problems in combinatorial optimization. Many researchers have suggested various mechanisms to solve this problem effectively. This work presents a combination of Particle Swarm Optimization (PSO) and Topological Sort approach to optimize TSPPC. The original PSO is designed for continuous problem. Therefore, it is incapable to optimize TSPPC, besides the reason of precedence constraint. The proposed PSO with Topological Sort approach was tested with different size of TSPPC problem. The optimization result showed that it yielded better performance in finding the shortest travelling time in all problem size. This finding is associated with better exploration and solution diversity features in PSO.

Key words: Travelling salesman problem; Precedence constraint; Particle Swarm optimization

1. Introduction

The travelling salesman problem (TSP) is modelled to locate the shortest possible length among the points once and only once which concerns on optimization (Sarin et al., 2014). It is formulated mostly using graph for easy interpretation. This problem comprises application of science and engineering in order to find the cheapest way of visiting all points and return to the starting point (Sugumar, 2012). The TSP has been implemented to solve different type of problem such as scheduling, manufacturing sequence and even sub-problem like DNA sequencing. This problem is not only crucially linked with computer science and mathematics, but the significant applied problem is also tied with TSP variants.

Travelling salesman problem with precedence constraint (TSPPC) is a variant of TSP. In TSPPC, all nodes should be visited once, but in a predetermined order. In optimization method, TSP is known to be a class of NP-hard problem and TSPPC is even more complex with the additional constraints (Sung & Jeong, 2014). Since no exact solution can be obtained in reasonable computational time and a good solution needs to be practical and implementable in real environment, careful development of new solution method is crucial.

With precedence constraints, TSPPC has a wider range of industrial applications which can be found in assembly line, routing, project management, scheduling and logistic. The most common application of TSPPC is in the field of pickup and delivery service such as taxi, chartered car, delivery of fast food and others (Ropke & Cordeau, 2009). Some other instance is where the Aircraft Landing Problem (ALP) and the Airport Take-Off Problems (ATP) seek to learn the chronological succession of aircraft landing on, or taking off from the available runways at airports (Bencheikh et al., 2011). Furthermore, TSPPC was applied to achieve maximum machine utility in real printed circuit board (PCB) assembly application. Computer or numerically controlled electronic component placement machines have brought speed, precision and reliability to complete the placement of all components on PCB (Alkaya & Duman, 2013).

TSPPC has been widely studied throughout the decades using various types of algorithms. It is popular among the researchers because its formulation can be modelled into numerous solutions as seen in the papers presented over the past few years. Various methods have been established to solve and optimize TSPPC, one of them is a genetic algorithm (GA) approach based on a topological sort (TS)-based representation procedure which can generate feasible sequences in TSPPCs (Yun & Moon, 2011). Recently, (Dao & Marian, 2013) suggested the development of genetic algorithms (GA) for integrated optimizations, (Sung et al., 2008) proposed an integer programming model for the operation sequencing with precedence constraints using adaptive evolutionary approach. For machining scheme selection and operation sequencing related problem, (Hua et al., 2007) developed a synthesizing optimization approach based on genetic algorithm (GA) to obtain the global optimal process plan. To ensure the solution's viability, the operation precedence constraints of

* Corresponding Author.

features can be changed automatically with the selection of different machining schemes.

From previous works, researchers tended to implement evolutionary computation techniques such as Genetic Algorithms (GA) to optimize TSPPC problem. In this study, we implement a combination of Particle Swarm Optimization (PSO) and Topological Sort approach to improve the computational efficiency and the quality of solutions. Topological Sort approach is a repair mechanism to ensure all the generated solution will be in a feasible region.

2. TSPPC Modelling

TSPPC can be represented by using directed acyclic graph (DAG). In the example as shown in Fig. 1, node represents the city that needs to be visited, while arc represents the precedence constraints. In this case, the incoming arc depicts the order of the node to be visited by its successive node. While, the outgoing arc means that the next node is to be visited after the successive one.

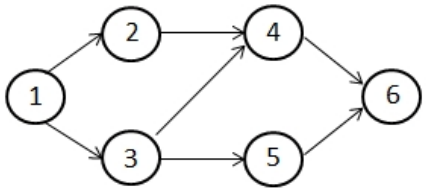

Fig. 1: Example of directed acyclic graph

The travel time, t_{ij} between nodes i and j is normally presented in n x n matrix as presented in Table 1. Based on Table 1, the travelling time between city 1 and 2 is t_{12} = 10 minutes while t_{51} = 5 minutes.

Table 1: Traveling time between node *i* and *j*

j *i*	1	2	3	4	5	6
1	-	10	6	8	5	9
2	10	-	2	4	6	8
3	6	2	-	3	5	7
4	8	4	3	-	9	10
5	5	6	5	9	-	3
6	7	8	7	10	3	-

The objective of TSPPC is to find the shortest route by visiting each node once only by fulfilling the predetermined precedence. This problem has been formulated by (Yun, Chung, & Moon, 2013) as follows:

i, j node index, i, $j = 1, \ldots, n$ where n is the number of nodes.

t_{ij} traveling time from nodes i to j.

ta_i arrival time at node i.

p set of nodes (i, j), where node i is visited before node j.

q set of nodes (i, j), where nodes i and j can be visited in any feasible sequence.

l arbitrary large positive number.

The variable is brought out to adapt the problem as follows:

$x_{ij} = 1$; if node i is visited before j;

$x_{ij} = 0$; otherwise:

Let E be the finding a node with the maximal arrival time. The E can be represented as follows:

Minimize

$$E = \max_{\alpha i}\{ta_i\} \quad (1)$$

subject to

$$ta_j - ta_i \geq t_{ij} \quad \alpha(i,j) \in p \text{ and } i \neq j \quad (2)$$

$$ta_j - ta_i + lx_{ij} \geq t_{ij}x_{ij} + t_{ji}x_{ji} \quad \alpha(i,j) \in q \text{ and } i \neq j \quad (3)$$

$$X_{ij} + X_{ji} = 1 \quad \alpha(i,j) \in q \text{ and } i \neq j \quad (4)$$

$$X_{ij} = 0,1 \quad \alpha(i,j) \in q \text{ and } i \neq j \quad (5)$$

Equation (1) shows the maximum among the arrival times. Equation (2) ensures all the cities are visited once. Equation (3) explains that the difference between arrival time at a node and the next one plus its large arbitrary positive number must bigger than the difference of traveling times which the notes are visited once only. Equation (4) and (5) declares the rules of value resemblance by the way of combination.

2. Particle swarm optimizations with topological sort

Particles Swarm Optimization (PSO) is an algorithm which optimises a problem iteratively. It has a population of particles travelling in search space according to simple formulae over the particle's position and velocity. However in original form, this algorithm cannot be directly used to optimise TSPPC because the original PSO is purposely designed for continuous problem, while TSPPC is discrete combinatorial problem.

In order to accommodate PSO for TSPPC, redefinition of particle and velocity in PSO is required. For this purpose, the PSO particles are defined as weight that determines which cities have a priority over the others. In this case, the Topological Sort approach is added to the PSO with the purpose to ensure the feasibility of the visited cities sequence. The flow chart of PSO with Topological Sort is presented in Fig. 2.

3.1. Initialisation

The number of particles ($npar$) and maximum number of iteration ($iter_{max}$) are set in this step. Then, the initial population which is known as swarm is produced by generating $npar$ set of initial position (X) and velocity (V) that consist of n random numbers between 0 and 10. As an example, one of the particles from initial population is, x_1= [3.46 1.33 8.12 2.56 4.74 5.66] and v_1= [0.0 0.1 0.3 0.5 0.08 0.4] as shown in Table 2.

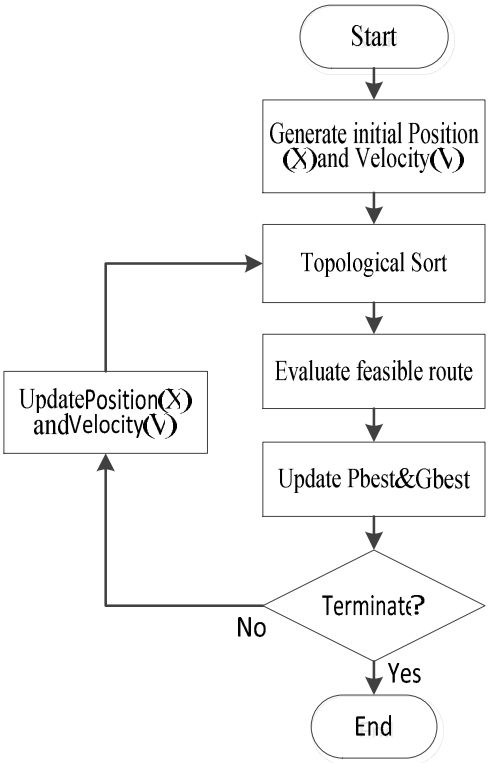

Fig. 2: Flowchart of PSO with topological sort

Table 2: Assigned position value for cities

Cities	1	2	3	4	5	6
x_1	3.46	1.33	8.12	2.56	4.74	5.66
v_1	0.00	0.10	0.30	0.50	0.08	0.40

3.2. Topological sort

Topological Sort is an approach used to generate feasible visiting route from the particle position (x_i) of the PSO. In this work, x_i value is assigned to each city and determines the visiting route.

The procedure of Topological Sort is presented as follow:

Procedure: Topological Sort
Begin
n; number of cities
st = 0; number of selected city
While $st \leq n$
Establish *Available Set*
$st = st + 1$
Select a city with highest weight from *available set* and place in st^{th} position of feasible route, F_r
Remove all outgoing arcs from selected city
Eliminate selected city from precedence graph
End While
End Procedure

For particle x_1= [3.46 1.33 8.12 2.56 4.74 5.66] and precedence graph in Fig. 1, this procedure is started by establishing Available Set (AS). AS consists of cities without incoming arc. For precedence graph in Fig. 1, only city 1 is available to be selected. Therefore, city 1 will be the first city to be visited in the visiting route.

Next, the selected city will be removed from the precedence graph, including the outgoing arc from this city. Then, the AS will be updated by removing the selected city from the list and inserting the new cities without incoming arc. For the considered example, the new AS is [2 3]. When there is more than one city in AS, the city with higher particle will be chosen as the next city to be visited. In this case, the x_1 value for city 2 is 1.33, while for city 3 the x_1 is 8.12. Therefore, city 3 is selected as the next city in the route. Next, the AS is updated and this procedure is repeated until complete. For the x_1, the feasible route is established as in Table 3. The feasible route generated from x1 is fr1= [1 3 5 2 4 6] as generated in Table 3.

Table 3: Example of topological sort

Iteration	Available set	Selected city
1	1	1
2	2, 3	3
3	2, 5	5
4	2	2
5	4	4
6	6	6

3.3. Evaluation

In this step, the feasible route is evaluated by using predefined objective functions. The objective functions are calculated using procedures and formulas in Eq. 1 - 5. For, fr_1= [1 3 5 2 4 6] the total traveling time are the summation of t_{13}, t_{35}, t_{52}, t_{24} and t_{46}.

3.4. Update Pbest and Gbest

Pbest is the best personal particle solution, while *Gbest* is the best solution among all particles. *Pbest* and *Gbest* will be saved in every iteration. The *Gbest* value in the final iteration is considered the best solution for the optimisation process.

3.5. Update position and velocity

The final step in PSO is to update swarm position and velocity. The purpose of this step is to establish new swarm set that follows the current *Pbest* and *Gbest*. In PSO, the position and velocity are updated using the following formulas:

$V_i(k+1) = wV_i(k) + c_1(P_{best} - X_i(k)) + c_2(G_{best} - X_i(k))$
$X_i(k+1) = X_i(k) + V_i(k+1)$
$X_i(k)$: i^{th} position at k^{th} iteration.
$V_i(k)$: i^{th} velocity at k^{th} iteration.
w : inertia coefficient
c_1 : Cognitive coefficient
c_2 : Social coefficient

For example, let
$x_1(1)$ = [3.46 1.33 8.12 2.56 4.74 5.66]
$v_1(1)$ = [0.0 0.1 0.3 0.5 0.08 0.4]
Pbest = [3.46 5.68 1.21 2.30 4.03 7.18]
Gbest = [1.46 0.44 1.81 0.55 1.26 0.72]
$w = c_1 = c_2 = 1.4$
$v_1(2) = 1.4(0.0) + 1.4(3.46-3.46) + 1.4(1.46-3.46)$

= -2.8

$v_1(2)$ = [-2.8 4.98 -18.08 -2.48 -5.75 -4.22]

$x_1(2)$ = 3.46 + (-2.8)

= 0.66

$x_1(2)$ = [0.66 6.31 -9.96 0.08 -1.01 1.43]

4. Computational experiment

For computational experiment purpose, seven test problems with different size ranging from 20 to 80 are used. These test problems are adopted from (Yun et al., 2013). In their work, they proposed a hybrid Genetic Algorithm (HGA) with adaptive local search to optimise the TSPPC problem. The optimisation results using PSO will be compared with HGA and two other algorithms that were also used in (Yun et al., 2013). These algorithms are GA with priority-based representation (GA-PB) (Gen, Altiparmak, & Lin, 2006) and GA with topological sort (GA-TS) (Yun & Moon, 2011).

The proposed PSO is coded by using MATLAB software and run on Intel Core i5-3230m, CPU @ 2.60 GHZ, RAM 8.00 GB and OS: Windows 7. The parameters in each measure are set as follows: total iteration number is 1000, particle size is 20 and C1 = C2 = C3 = 1.4. Seven cases of the TSPPC which consists of 20, 30, 40, 50, 60, 70 and 80 nodes respectively, are considered.

Fig. 3 shows the performance results of each method. The detail results are presented in Table 4. For TSPPC with 20 nodes, all algorithms generated a similar result in terms of the best value except the suggested PSO which yielded 110 as the best value. The best value of the proposed PSO was smaller than that those of other approaches. It showed that the time taken to visit all the cities was the shortest and the most preferable compared to the routes generated using other algorithms.

PSO also had superior performance compared to the other approaches for the problem with 30 nodes and 40 nodes. The GA-TS approach and HGA approach produced the same best value while the GA approach came out with the highest travelling time. It exhibited that the results from using TS-based strategy (in both PSO and GA) were better than the priority-based representation system. However, the PSO presented better performance even though GA-TS had also used topological sort strategy.

The suggested PSO method in the TSPPCs with 50, 60, 70 and 80 nodes showed the best results consistently compared to GA-based algorithms. The optimisation result in larger problem size showed the actual capability of the algorithm when dealing with extremely large number of possible solution. For example, in the problem with 50 nodes, there were roughly 3.05x1064 possible solutions including the infeasible solutions. Implementation of Topological Sort approach that ensures all particles will generate feasible solutions give better chance for PSO to explore the entire search space.

From Fig. 3, the proposed PSO came out with better travelling time in all problems. Meanwhile, the Hybrid Genetic Algorithm (HGA) was consistently in the second best position, followed by GA with Topological Sort (GA-TS), while GA-PB was in the last position in term of best final solution. This finding is highly related with better exploration capability in PSO compared to GA. This advantage has been discussed in many other applications (Hassan et al., 2005). Besides that, PSO allowed greater diversity because of two population (current and Pbest).

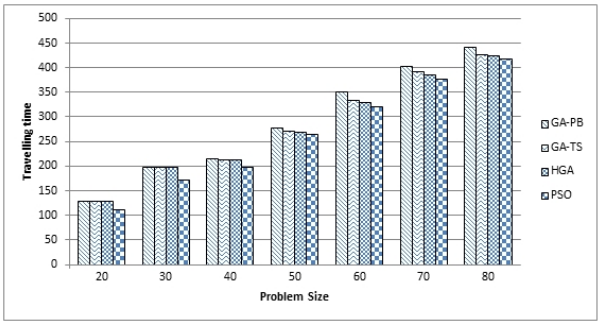

Fig. 3: Comparison of optimisation results

Table 4: Optimisation results using PSO

Problem size	Traveling time	Visiting route
20	110	1-4-3-2-6-5-8-7-9-14-13-17-12-16-19-11-10-15-18-20
30	170	1-4-3-2-6-7-10-5-9-12-8-11-15-14-16-17-13-18-22-19-20-21-23-24-25-27-28-26-29-30
40	197	3-2-1-4-5-6-7-10-12-9-11-8-15-14-16-13-17-20-19-22-21-18-24-25-26-27-30-23-29-31-32-28-33-35-34-36-37-39-38-40
50	264	2-18-3-7-27-32-28-33-8-13-17-12-4-38-37-36-41-11-1-6-5-10-9-14-15-16-20-42-43-23-22-21-26-19-25-24-29-30-31-48-35-34-39-40-44-46-49-45-47-50
60	319	1-4-3-7-2-6-8-5-9-11-10-12-13-15-14-16-17-18-21-22-20-23-19-24-25-26-27-31-32-28-30-29-33-34-37-36-38-35-42-43-41-40-44-39-46-49-47-48-45-52-51-50-53-54-59-58-57-55-56-60
70	377	3-4-8-1-9-13-17-12-18-23-28-27-2-7-6-5-10-11-14-15-16-22-20-21-26-19-25-24-29-30-35-34-31-32-33-38-39-40-37-36-41-47-42-43-46-45-44-48-50-49-55-54-61-60-59-51-52-53-58-64-63-68-57-65-56-62-66-69-67-70
80	416	2-3-5-1-4-8-7-9-6-10-13-11-12-14-19-16-15-18-17-21-22-20-24-23-28-25-26-27-29-34-32-33-31-30-35-37-36-38-42-43-41-40-44-39-45-46-49-51-48-47-54-52-50-53-58-59-57-60-56-55-66-61-62-65-64-63-68-70-69-67-74-71-73-75-72-78-76-79-77-80

5. Conclusions

The objective of this paper is to implement Particle Swarm Optimisation (PSO) for the Travelling

Salesman Problem with Precedence Constraint (TSPPC). Since PSO is developed for continuous problem, Topological Sort approach is used to accommodate PSO for discrete combinatorial problem like TSPPC.

Seven test problems of TSPPC consisted of various size nodes had been used to measure the performance of PSO compared to Genetic Algorithm (GA) based techniques. The optimisation results indicated that the proposed PSO approach had outperformed the GA-based algorithms in all problems. This finding is related with PSO features that allow better exploration and solution diversity. As for future recommendation, continuous effort will be put to increase PSO performance.

References

Alkaya, A. F., & Duman, E. (2013). Application of Sequence-Dependent Traveling Salesman Problem in Printed Circuit Board Assembly. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, *3*(6), 1063–1076.

Bencheikh, G., Boukachour, J., El, A., & Alaoui, H. (2011). Improved Ant Colony Algorithm to Solve the Aircraft Landing Problem. *International Journal of Computer Theory and Engineering*, *3*(2), 224–233.

Dao, S. D., & Marian, R. M. (2013). *Genetic algorithms for integrated optimisation of precedence-constrained production sequencing and scheduling* (pp. 65–80). Springer New York.

Gen, M., Altiparmak, F., & Lin, L. (2006). A genetic algorithm for two-stage transportation problem using priority-based encoding. *Or Spectrum*, *28*(3), 337–354.

Hassan, R., Cohanim, B., De Weck, O., & Venter, G. (2005). A comparison of particle swarm optimization and the genetic algorithm. In *Proceedings of the 1st AIAA multidisciplinary design optimization specialist conference* (pp. 18–21).

Hua, G., Zhou, X., & Ruan, X. (2007). GA-based synthesis approach for machining scheme selection and operation sequencing optimization for prismatic parts. *The International Journal of Advanced Manufacturing Technology*, *33*(5-6), 594–603.

Ropke, S., & Cordeau, J.-F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, *43*(3), 267–286.

Sarin, S. C., Sherali, H. D., Judd, J. D., & Tsai, P.-F. (Jennifer). (2014). Multiple asymmetric traveling salesmen problem with and without precedence constraints: Performance comparison of alternative formulations. *Computers & Operations Research*, *51*, 64–89.

Sugumar, R. (2012). Implementation of Genetic Algorithm in Traveling Salesman Problem . *African Journal of Science and Research*, *1*(1), 1–7.

Sung, J., & Jeong, B. (2014). An Adaptive Evolutionary Algorithm for Traveling Salesman. *The Scientific World Journal*, *2014*, 1–11.

Sung, J., Moon, C., & Jeong, B. (2008). An Adaptive Evolutionary Algorithm for the Operation Sequencing with Precedence Constraints. In *Innovative Computing Information and Control, 2008. ICICIC'08. 3rd International Conference on* (p. 113).

Yun, Y., Chung, H., & Moon, C. (2013). Hybrid genetic algorithm approach for precedence-constrained sequencing problem. *Computers & Industrial Engineering*, *65*(1), 137–147.

Yun, Y., & Moon, C. (2011). Genetic algorithm approach for precedence-constrained sequencing problems. *Journal of Intelligent Manufacturing*, *22*(3), 379–388.