An efficient genetic algorithm for the traveling salesman problem with precedence constraints

Article *in* European Journal of Operational Research · August 2002

DOI: 10.1016/S0377-2217(01)00227-2 · Source: RePEc

4 authors, including:

Chiung Moon
Yonsei University
25 PUBLICATIONS **757** CITATIONS
SEE PROFILE

Yoonho Seo
Korea University
40 PUBLICATIONS **492** CITATIONS
SEE PROFILE

ELSEVIER

European Journal of Operational Research 140 (2002) 606–617

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Discrete Optimization

An efficient genetic algorithm for the traveling salesman problem with precedence constraints

Chiung Moon [a,*], Jongsoo Kim [a], Gyunghyun Choi [a], Yoonho Seo [b]

[a] *Department of Industrial Engineering, Hanyang University, Ansan 425-791, Republic of Korea*
[b] *School of Industrial Engineering, University of Ulsan, Ulsan 680-749, Republic of Korea*

Received 2 June 2000; accepted 28 February 2001

Abstract

The traveling salesman problem with precedence constraints (TSPPC) is one of the most difficult combinatorial optimization problems. In this paper, an efficient genetic algorithm (GA) to solve the TSPPC is presented. The key concept of the proposed GA is a topological sort (TS), which is defined as an ordering of vertices in a directed graph. Also, a new crossover operation is developed for the proposed GA. The results of numerical experiments show that the proposed GA produces an optimal solution and shows superior performance compared to the traditional algorithms. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Traveling salesman problem with precedence constraints; Genetic algorithm; Topological sort; Optimization

1. Introduction

The traveling salesman problem (TSP) is a typical example of a very hard combinatorial optimization problem. The problem is to find the shortest tour that passes through each vertex in a given graph exactly once. The TSP has received considerable attention over the last two decades and various approaches are proposed to solve the problem, such as branch-and-bound [6], cutting planes [14], 2-opt [12], simulated annealing [9], neural network [19], and tabu search [7]. Some of these methods are exact algorithms, while the others are near-optimal or approximate algorithms. The exact algorithms include the integer linear programming approaches with additional linear constraints to eliminate infeasible subtours [1,5,8,13,16,21]. On the other hand, network models yield appropriate methods that are flexible enough to include the precedence constraints [6,11]. More recently, genetic algorithm (GA) approaches are successfully implemented to the TSP [3]. Potvin [16] presents survey of GA approaches for the general TSP.

When there exist an order in which the vertex should be visited, the problem is referred to as the traveling salesman problem with precedence

* Corresponding author.
E-mail addresses: cumoon@hanyang.ac.kr, cumoon@hanmail.net (C. Moon).

PII: S0377-2217(01)00227-2

constraints (TSPPC). The TSPPC can be applied to many industrial problems, such as scheduling, routing decision, process sequencing, etc. But, as pointed out by Savelsbergh and Sol [18], much less attention has been devoted to the TSPPC. The TSPPC is harder than the general TSP because the model formulations are too complex and the algorithms for solving these models are difficult to implement. Since the TSPPC belongs to the class of NP-hard problem, optimal solution to the problem cannot be obtained within a reasonable computational time for a large size problem.

Savelsbergh and Sol [18] present the TSPPC model to solve the Dial-A-Ride problem where a vehicle should transport a number of passengers. Each passenger should be transported from a given location to a given destination. Kusiak and Finke [10] presents a branch-and-bound technique for the flexible forging machine scheduling problem with sequence-dependent changeover costs and precedence constraints. Renaud et al. [17] propose a heuristic model to solve the pickup and delivery TSP which is formulated as the TSPPC model. Mingozzi et al. [15] present dynamic programming strategies for the TSP with time windows and precedence constraints. Fagerholt and Christiansen [4] consider a TSPPC with time window and allocation to solve the bulk ship scheduling problem. The model is solved as a shortest path problem on a graph.

In this paper we propose an efficient GA approach to solve the TSPPC which is formulated by network model. Despite the abundance of contributions to the TPS in the GA research area, we cannot find any results for the TSPPC. The traditional GA approaches are not suitable to solve the TSPPC because the precedence constraints cannot be handled adequately. In order to efficiently handle the precedence constraints, a new encoding scheme is developed. The key concept of the new encoding to use a topological sort (TS), which is defined as an ordering of vertices in a directed graph. The merit of the new encoding is that any common genetic operations on this encoding will not generate infeasible solution so that it need not any modification both for the initial population and for the offspring generated in the evolutionary process. Therefore, the proposed GA approach does not generate any infeasible solutions for the TSPPC. A new crossover operator is also developed to efficiently search towards the optimal solution.

2. Network model for TSPPC

To formulate the TSPPC, the two-commodity network flow model can be used [6]. Let c_{ij} be the travel time from vertex v_i to v_j and s be the first selected vertex in the graph. Suppose that there are two distinct commodities p and q in the network with J nodes or cities. While commodity p is supplied by $(J-1)$ units at a selected starting node and used by one unit at each node that is not the starting node, commodity q is consumed by $(J-1)$ units at the starting node and supplied by one unit at the other nodes. Such network flow of the commodities are characterized by two properties: First, the sum of commodities p and q in any feasible tour should be equal to $J-1$. And, the quantity of commodity p (or q) outbounded from a node is decreasing as the tour proceeds. These properties are used to model the precedence relations for the TSPPC.

Three variables are introduced for the two-commodity network flow model: y_{ij}^p are the quantity of commodity p from vertex v_i to v_j, y_{ij}^q are the quantity of commodity q from vertex v_i to v_j.

$$y_{ij} = \begin{cases} 1 & \text{if vertex } j \text{ is visited immediately after vertex } i, \\ 0 & \text{otherwise.} \end{cases}$$

The two-commodity network flow model for the TSPPC can be described as follows:

$$\text{Minimize} \sum_{i=1}^{J} \sum_{\substack{j=1 \\ i \neq j}}^{J} \frac{1}{(J-1)} c_{ij} \left(y_{ij}^p + y_{ij}^q \right), \tag{1}$$

subject to

$$\sum_{j=1}^{J} y_{ij}^p - \sum_{j=1}^{J} y_{ji}^p = \begin{cases} J-1 & \text{for } i = s, \\ -1 & \text{otherwise,} \end{cases} \tag{2}$$

$$\sum_{j=1}^{J} y_{ij}^{q} - \sum_{j=1}^{J} y_{ji}^{q} = \begin{cases} -(J-1) & \text{for } i = s, \\ +1 & \text{otherwise,} \end{cases} \tag{3}$$

$$\sum_{j=1}^{J} \left(y_{ij}^{p} + y_{ij}^{q}\right) = J - 1 \quad \forall i, \tag{4}$$

$$y_{ij}^{p} + y_{ij}^{q} = (J-1)y_{ij} \quad \forall i \text{ and } j, \tag{5}$$

$$\sum_{j=1}^{J} y_{uj}^{p} - \sum_{j=1}^{J} y_{vj}^{p} \geqslant 1 \quad \text{for } (v_u \to v_v)\ (v_v \neq s), \tag{6}$$

$$y_{ij}^{p} \geqslant 0 \quad \forall i \text{ and } j, \tag{7}$$

$$y_{ij}^{q} \geqslant 0 \quad \forall i \text{ and } j, \tag{8}$$

$$y_{ij} \in \{0, 1\} \quad \forall i \text{ and } j. \tag{9}$$

The objective function (1) expresses total travel times for all vertices, since the sum of commodities p and q between vertices v_i to v_j on any feasible sequence (i.e., $y_{ij} = 1$) is equal to $J - 1$ (i.e., $y_{ij}^{p} + y_{ij}^{q} = J - 1$).

Constraints (2) and (7) are used to ensure the feasibility of flow of commodity p. Similarly, constraints (3) and (8) are for feasibility of commodity q. Constraint (4) ensures a feasible tour, i.e., feasible sequence. Constraint (5) explains that, if $y_{ij} = 1$, the sum of commodities p and q between v_i and v_j be $J - 1$. From constraint (5), we see that the $(1/(J-1))c_{ij}(y_{ij}^{p} + y_{ij}^{q})$ in objective function is the same meaning as the $c_{ij}y_{ij}$. Constraint (6) is for the precedence relationship between vertices.

3. Development of solution method

A GA approach based on TS is presented in this section. GA is one of the evolutionary search methods that can provide optimal or near optimal solutions for the combinatorial optimization problems. The most attractive features of GA is the flexibility of handling various kinds of objective functions with fewer requirements on fine mathematical properties [3]. It has been applied to a number of fields like engineering, biology, computer science, and social sciences. The main issues in developing a GA-based algorithm are chromosome representation, initialization of the population, evaluation measure, crossover, mutation, and selection strategy. Also, the genetic parameters such as population size *pop_size*, number of generation *max_gen*, probability of crossover p_c, and probability of mutation p_m, should be determined before execution of a GA.

Since the TSPPC has many precedence constraints, the traditional representation schemes generate infeasible solutions. Because most sequences in the search space would not correspond to feasible tours. To resolve this problem, fitness function with penalty terms, and repair operators to transform infeasible solutions into feasible ones, have been presented to handle infeasible solutions. However, these methods cannot handle the precedence constraints efficiently and are not satisfactory in speed and in accuracy of the generated solution. In this section, we developed an efficient representation scheme to resolve this difficulty.

3.1. Topological sort

While developing a solution method for TSP using GA, a critical issue we usually encounter is the development of a simple scheme to represent a feasible solution. The simplicity and clarity of the representation scheme can lessen the complexity of algorithm. In order to be a good representation scheme, it should be able to generate all possible topological orders for a given network.

The TS is an ordering of vertices in a directed graph, such that if there is a path from v_i to v_j, then v_j appears after v_i in the ordering. It is clear that a topological order is not possible if the graph has a cycle. The TS procedure to sort the vertices into topological order is straightforward. The first step is to select any vertex without incoming edges and then store the vertex and its position. Then, the vertex and all the edges leading out from the vertex are removed from the graph.

In a directed graph G, the vertices represent activities and the edges represent the precedence relations between activities. A directed edge $\langle v_i, v_j \rangle$ indicates that vertex v_i must be completed before

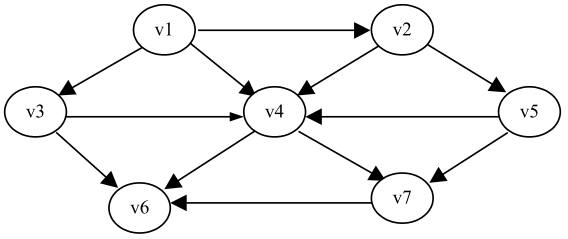

Fig. 1. An example of directed graph with precedence constraints.

vertex v_j. An example of directed graph to represent the precedence constraints is illustrated in Fig. 1.

In the directed graph, a path is complete if the path visits all the vertices, and feasible if it does not violate the precedence constraints. A topological sort technique can be used to obtain all the feasible paths in a directed graph [20]. But more than single sequence of vertices can be derived from a directed graph using the TS technique. For example, paths v_1–v_2–v_5–v_4–v_7–v_3–v_6 and v_1–v_2–v_5–v_4–v_3–v_7–v_6 are both topological orders that can be derived from the above directed graph. To overcome this pitfall, and to derive the feasible complete paths from a directed graph, an ordering technique using the topological sort and random assignment of priority is proposed in this paper. By associating with the genetic algorithm, this ordering technique can be effectively used to find the shortest tour to visit all vertices exactly once.

3.2. Representation

But, sole application of TS may result in the selecting problem if there are multiple vertices with predecessors. In this paper, to derive a unique sequence from directed graph, a random priority assignment technique to randomly assign a differentiated priority to each vertex in the graph is used. Therefore, a string of priorities can represent a feasible path.

For instance, for the example in Fig. 1, a chromosome structure can be represented as shown in Fig. 2. In Fig. 2, the second row means the selection priority in case of multiple vertices with no incoming edges. The priority for each vertex is generated at random within $[1, J]$ exclusively, where J is the number of vertices or machining processes. With this representation scheme, changing the positional sequences of the priority can derive all the feasible solutions from a directed graph.

A procedure to generate a feasible path from a directed graph using TS and random priority assignment technique is described in Fig. 3.

Let us see how to generate a feasible path from the representation scheme. We can consider the TSP network in Fig. 1. In the network, the first operation to be picked is v_1, since it is the only one with no predecessors. Then, v_1 is stored in queue and v_1 and the edges $v_1 \rightarrow v_2$, $v_1 \rightarrow v_3$ and $v_1 \rightarrow v_4$ are removed. In the resulting network, v_2 and v_3 have no predecessor. The vertex v_3 is selected as the next position since its priority is higher than

Vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7
Priority	5	1	7	2	4	6	3

Fig. 2. Priority representation structure.

procedure: a feasible path generation
  input: directed graph.
  while (any vertex remains) **do**
    if every vertex has a predecessor, **then** the network is infeasible: **stop.**
    else pick a vertex v with the highest priority among vertices with no predecessors;
    $que \leftarrow v$;
    delete v and all edges leading out of v from the directed graph;
  end_while.
end_procedure.

Fig. 3. Feasible path generation algorithm.

that of v_2. Continuing this procedure, a final feasible path $(v_1, v_3, v_2, v_5, v_4, v_7, v_6)$ is uniquely obtained for the string of priority '5172463'.

The TS-based representation employed here belongs to indirect representation, and is m-to-1 mapping. Generally, when using m-to-1 mapping encoding, we should consider diversity of the chromosome population. Hence a larger population size is used to increase the population diversity.

3.3. Initialization

The initialization of the population of chromosomes can be done by a random or well-adapted generation method. Since the random approach is used in this paper, the chromosomes as many as the population size *pop_size* are generated randomly. Each chromosome is represented as a string of integers. Each digit of the string means the priority of the gene and ranges between one and the number of genes. As described in the previous section, the process to generate initial population using TS and random assignment of priority technique does not yield any illegal chromosome.

3.4. Generation of offspring population

To create the next generation, a new set of chromosomes called offspring is formed by the execution of genetic operators such as selection,

procedure: moon crossover
begin
$osp \leftarrow$ null;
$k \leftarrow 0$;
Select two random chromosomes p_a and p_b, where $p_a = g_1\ g_2\ g_3\ g_4 \ldots\ g_J$, and $p_b = q_1\ q_2\ q_3\ q_4 \ldots\ q_J$;
Select two genes from the p_a at random. The substrings defined by the two genes;
$osp \leftarrow$ the substring between g_i and g_j selected from the p_a;
if the length of $osp = J$ **then end**;
else $sub\_p_b \leftarrow$ the remaining substring results from the deleting genes which are already selected from the p_a, *i.e.*, $sub\_p_b = p_a - osp$;
end_if
while (length of $osp \neq J$) **do**
  if $i = 1$ **then** $i = J+1$;
    $i \leftarrow i - 1$;
    $k \leftarrow k + 1$, $k = 1, 2, \ldots$, length of sub_p_b;
    if $g_i \neq q_k$, **then** $osp \leftarrow <osp, g_i, q_k>$;
    else $g_i = q_k$, **then** $osp \leftarrow < osp, g_i >$;
  else if $j = J$ **then**
    $i \leftarrow i - 1$;
    $k \leftarrow k + 1$, $k = 1, 2, \ldots$, length of sub_p_b;
    if $g_i \neq q_k$, **then** $osp \leftarrow < q_k, g_i,$ osp >;
    else $g_i = q_k$, **then** $osp \leftarrow < g_i, osp >$;
      else
    $i \leftarrow i - 1$;
    $k \leftarrow k + 1$, $k = 1, 2, \ldots$, length of sub_p_b;
    if $g_i \neq q_k$, **then** $osp \leftarrow <g_i, osp, q_k>$;
    else $g_i = q_k$, **then** $osp \leftarrow < g_i, osp >$;
      end_if
  end_while.
end_procedure.

Fig. 4. Procedure of moon crossover.

crossover and mutation. In particular, the crossover operator acts as a main operator and exercises a great influence on the performance of the GA approach, while the mutation operator acts as a background operator.

A new crossover operator we are proposing is developed. The proposed crossover is called the *moon crossover* because it is very similar to the change of the moon such as waxing moon→half-moon→gibbous→full moon. For example, a subtour can be compared to the waxing moon or the half-moon. The procedure of the moon crossover operator is described in Fig. 4.

In Fig. 4, $\langle \text{osp}, g_i \rangle$ is the concatenation operator to add g_i after the substring osp. An example of the *moon crossover* is illustrated in Fig. 5. Suppose that two chromosomes are $p_a = [5\ 1\ 7\ 2\ 4\ 6\ 3]$ and $p_b = [3\ 6\ 1\ 4\ 2\ 5\ 7]$. First, select the substring from p_a at random. In this example, the substring is selected as $\text{osp} = [7\ 2]$. Then, we can obtain $sub\_p_b = [3\ 6\ 1\ 4\ 5]$ from p_b. Next, $g_2 = 1$ and $q_1 = 3$ because $i \leftarrow 3 - 1$ and $k \leftarrow 0 + 1$. The offspring becomes $\text{osp} = [1\ 7\ 2\ 3]$. In the same way, add g_1, q_2, and the offspring becomes $\text{osp} = [5\ 1\ 7\ 2\ 3\ 6]$. Now the next is $g_7 = 3$ and $q_3 = 1$ and the cities have already appeared in the offspring, so we cannot add these cities into the offspring osp. Finally, q_4 is picked and the offspring becomes $\text{osp} = [5\ 1\ 7\ 2\ 3\ 6\ 4]$.

Using the same procedure, we can produce the second offspring as [3 6 1 4 5 7 2].

The swap mutation operator is introduced here. The swap scheme is select two genes within a chromosome at random and then swap these contents as shown in Fig. 6.

3.5. Evaluation and selection

To improve the solutions, each chromosome in the population is evaluated using a measure of fitness. The fitness measure we are using the objective function value in Eq. (1). In this way, we can find the best chromosome that corresponds to the shortest path. The objective function for each chromosome in the population can be obtained through the TS algorithm presented in the previous section.

Selection strategy is concerned with choosing chromosomes from population space. It may create a new population for the next generation based on either parent and offspring or part of them. A mixed strategy based on the roulette wheel and elitist selection is adopted as the selection procedure.

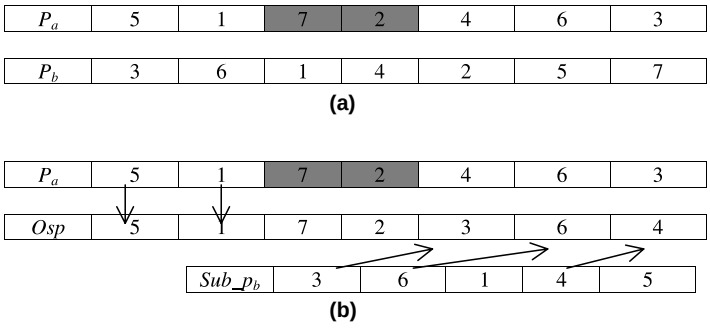

Fig. 5. Illustration of the moon crossover: (a) select the substring from p_a at random; (b) produce a offspring.

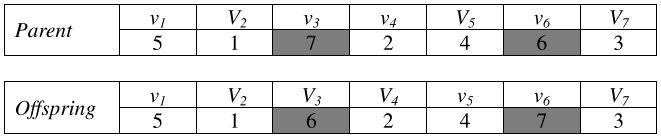

Fig. 6. The swap mutation operator.

procedure: Applied Genetic Algorithm

initialization

$t \leftarrow 0$;

set population size *pop_size,* number of generation *max_gen*, probability of crossover p_c,

and probability of mutation p_m;

initialize parent population *P(t)*;

evaluate *P(t)* and select the best solution σ^* with the minimum objective function among *P(t)*;

while (no termination criteria) **do**

regenerate *C(t)* from *P(t)* by applying the crossover and mutation operations;

evaluate *C(t)* and select the current best solution σ with the minimum objective value among *C(t)*;

update the best solution σ^*, *i.e.,* **if** $\sigma < \sigma^*$, **then** $\sigma^* = \sigma$;

select *P(t+1)* from *P(t)* and *C(t)*;

$t \leftarrow t + 1$;

end_while.

end_procedure.

Fig. 7. Applied genetic algorithm.

3.6. Genetic algorithm application

Let $P(t)$ and $C(t)$ be, respectively, populations for parent and offspring in generation t. Overall procedure of the proposed genetic algorithm is described in Fig. 7.

4. Experiments

Numerical experiments have been provided in order to demonstrate the effectiveness and efficiency of the proposed GA approach. The first experiment has 6 vertices and 6 precedence constraints as shown in Fig. 8. We impose the transition time between the vertices i and j as in Table 1.

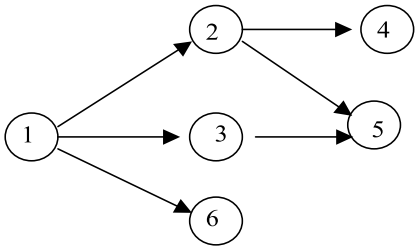

Fig. 8. TSP network with precedence constraints.

Table 1
Transition time between vertices

v_j	1	2	3	4	5	6
1	–	7	5	6	10	9
2	7	–	14	6	10	8
3	5	14	–	16	16	10
4	6	6	16	–	10	6
5	10	10	16	10	–	12
6	9	8	10	6	12	–

To solve the problem using the proposed GA, the genetic parameters are set as: maximum generation, $max\_gen = 20$; population size, $pop\_size = 10$; crossover probability, $p_c = [0.8, 0.5]$; mutation probability, $p_m = [0.1, 0.4]$; run by 5 times. Based on these setting, we found that the best settings for them are $p_c = 0.6$ and $p_m = 0.2$.

In the experiment, the proposed GA approach can find the optimal solutions on all runs. The optimal path and optimal value are 1–3–6–2–4–5 and 39, respectively. Also, we solved the problem using branch-and-bound technique. The detailed solution step is given in [10]. The optimal solution we obtain is $y_{13} = 1$, $y_{36} = 1$, $y_{62} = 1$, $y_{24} = 1$ and $y_{45} = 1$. From the above results, we see that two methods generate the same optimal solution.

We do experiment on a more complex problem with 20 vertices and 31 precedence constraints. The precedence relation network is shown in Fig. 9. We impose the transition time between vertices as given in Table 2. In this table, the transition time is symmetric as Table 1.

To solve the second problem, the parameters are set as: $max\_gen = 500$; $pop\_size = 150$; $p_c = 0.5$; $p_m = 0.2$; run by 10 times. By using the proposed GA and the branch-and-bound technique, we can obtain a same solution throughout the trials. The optimal path is 6–1–2–7–5–11–4–3–18–12–10–9–16–17–8–14–13–19–15–20 with the corresponding length 61. In the same parameters setting

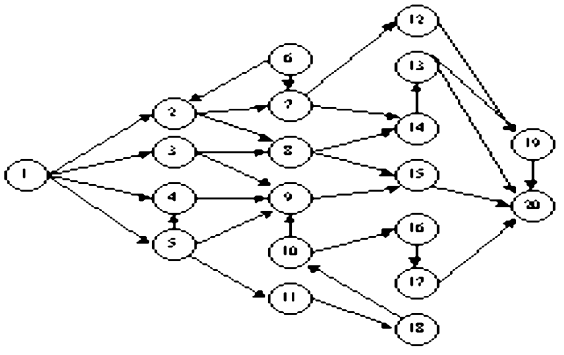

Fig. 9. TSP network with 20 vertices and 31 precedence constraints.

environment, the problem is also solved using the traditional GA which has order representation scheme. Since the traditional method generates infeasible solutions for the problem with constraints, it is needed to prevent the infeasible paths. For the comparison, we use the penalty technique proposed by Gen and Cheng [2]. The penalty technique transform the constrained problem into unconstrained problem by penalizing infeasible solutions. Fig. 10 shows the evolutionary process of the two approaches, which is plotted according to the average results of 10 trials.

From Fig. 10, we see that the proposed GA shows a better evolutionary search behavior to reach the optimal solution than the GA with penalty strategy does. The former reaches the optimal solution 10 times in 10 run times while the latter has 4 times in 10 run times. In all run times, we see that the proposed GA provides high quality solutions more quickly than the GA with penalty strategy does. Therefore we may conclude that the proposed GA is more suitable for the TSPPC than the GA with penalty strategy.

For a larger problem, we consider a network with 40 vertices and 56 precedence constraints. The precedence relation network is shown in Fig. 11. The transition times between operations are randomly generated within [1,15].

To solve the problem, the genetic parameters for the proposed GA are set as: $max\_gen = 500$; $pop\_size = 150$; $p_c = 0.5$; $p_m = 0.2$; run by 10 times. In this experiment, we can reach an optimal or near optimal solution through all trials. Fig. 12 illustrates the evolutionary process of the proposed GA approach, which is based on the average results of 10 trials.

We also compare the performance of *moon crossover* and traditional crossover operators, such as OX crossover and position-based crossover, using the second experiment data with 20 vertices and 31 precedence constraints. For this comparison, we use the same genetic parameter set as follows: $max\_gen = 500$; $pop\_size = 150$; $p_c = 0.5$; $p_m = 0.2$; run by 10 times. The evolutionary process of the average results of 10 trials is shown in Fig. 13.

Table 2
Transition time between vertices

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1		2	10	4	10	2	11	9	1	3	7	12	5	8	8	8	13	7	2	12
2			12	12	4	6	2	8	6	7	14	10	11	9	2	6	13	14	1	3
3				5	8	2	5	9	7	8	13	6	9	2	6	6	14	2	4	5
4					6	6	11	12	5	11	4	5	11	1	3	10	17	10	14	14
5						7	2	13	3	10	6	7	14	8	7	7	5	1	8	13
6							2	6	14	6	9	3	7	13	13	3	13	10	13	3
7								11	14	6	4	10	7	6	12	9	10	8	5	4
8									14	14	3	11	1	1	3	10	6	5	14	14
9										3	12	12	2	12	4	2	14	13	11	7
10											8	4	8	5	4	4	3	6	12	11
11												5	9	4	9	9	6	12	14	11
12													7	6	10	13	7	1	6	8
13														1	1	4	1	12	4	6
14															4	9	12	4	9	13
15																1	1	1	1	5
16																	8	5	2	14
17																		14	7	10
18																			5	6
19																				14
20																				

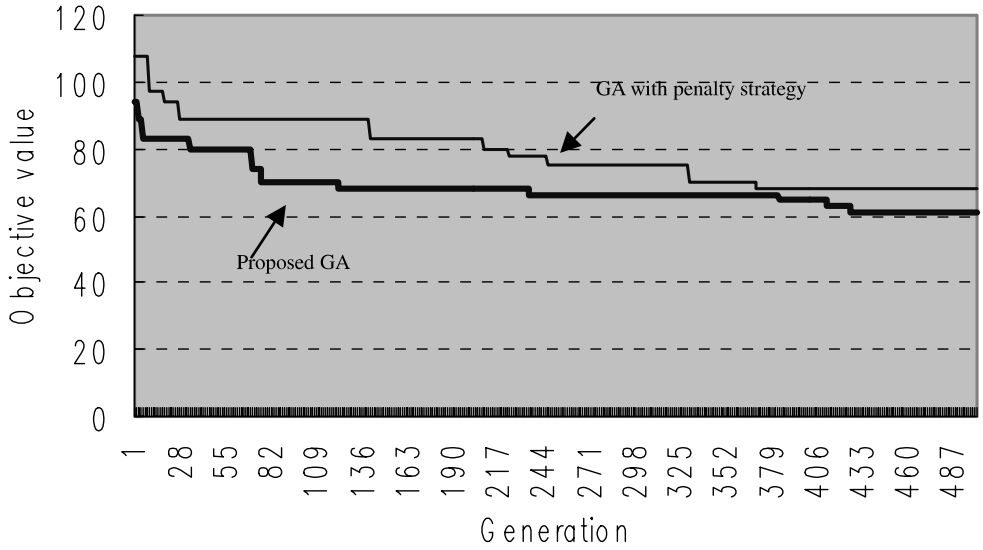

Fig. 10. Comparison of evolutionary process between two methods.

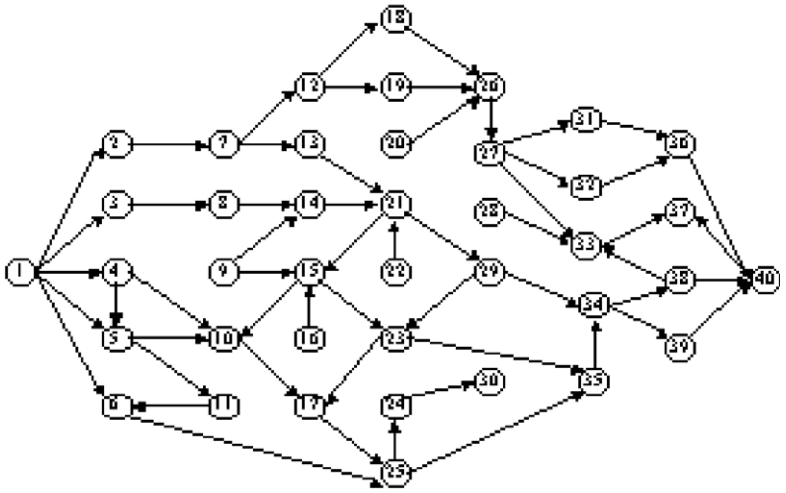

Fig. 11. Network for the problem with 40 vertices and 56 precedence constraints.

From Fig. 13, we see that the performance of moon crossover, OX crossover and position-based crossover is almost same, but OX and position-based crossovers cannot reach an optimal solution for all trials.

From the experiment results, we see that the proposed GA shows superior performance than the previous methods such as branch-and-bound or GA with penalty strategy. Through it is difficult to guarantee if the proposed GA has got the optimal solution to the TSPPC problem, we still can get some conclusion about the solution by GA approach. We can sure that the proposed GA approach can get the optimal solution with

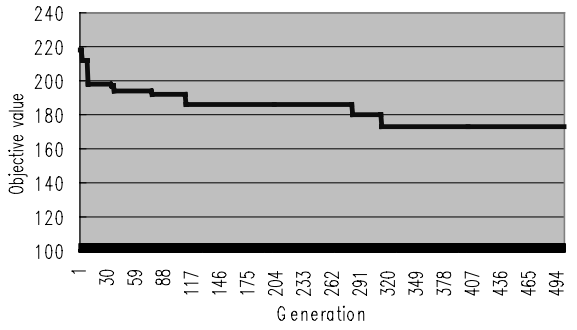

Fig. 12. Illustrating the evolutionary process of the proposed GA approach.

great probability when the problem size is not too large.

5. Conclusion

In this paper, we developed a TSPPC model using the two-commodity network flow model. To solve the model, we proposed an efficient GA approach based on TS concept. The proposed GA approach shows superior performance compared to the traditional algorithms. For small and medium size problems, we obtained optimal solutions. On a larger size problem, the proposed GA approach generated best solutions. Thus, we may conclude that the proposed GA approach is efficient methodology for the TSPPC.

References

[1] G. Carpaneto, P. Toth, Some new branching and bounding criteria for the asymmetric traveling salesman problem, Management Science 26 (1980) 736–743.
[2] M. Gen, R. Cheng, Interval programming using genetic algorithms, in: Proceedings of the First International Symposium on Soft Computing for Industry, 1996, pp. 113–116.
[3] M. Gen, R. Cheng, Genetic Algorithms and Engineering Design, Wiley, New York, 1997.
[4] K. Fagerholt, M. Christiansen, A travelling salesman problem with allocation time window and precedence constraints an application to ship scheduling, International Transactions in Operational Research 7 (2000) 231–244.
[5] M. Fischetti, P. Toth, An additive bounding procedure for combinatorial optimization problems, Operations Research 37 (1989) 319–328.
[6] G. Finke, A. Claus, E. Gunn, A two-commodity network flow approach to the traveling salesman problem, Congressus Numerantium 41 (1984) 167–178.
[7] F. Glover, Artificial intelligence, heuristic frameworks and tabu search, Managerial & Decision Economics 11 (1990) 365–378.
[8] L. Gouveia, J.M. Pires, The asymmetric travelling salesman problem and a reformulation of the Miller–Tucker–Zemlin constraints, European Journal of Operational Research 112 (1999) 134–146.
[9] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Configuration space analysis of travelling salesman problem, Journal Physique 46 (1985) 1277–1292.

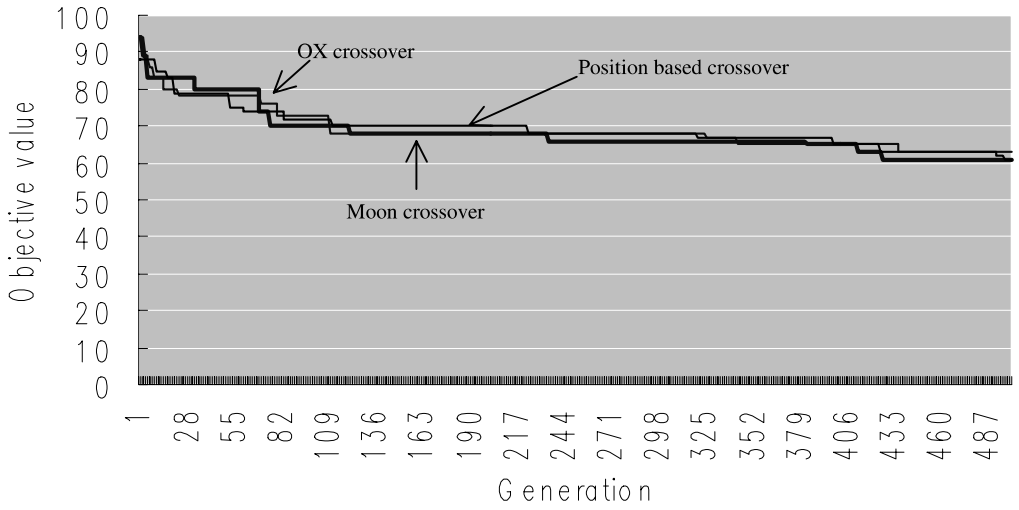

Fig. 13. Comparison of evolutionary process on the crossover.

[10] A. Kusiak, G. Finke, Modeling and solving the flexible forging module scheduling problem, Engineering Optimization 12 (1) (1987) 1–12.
[11] A. Langevin, F. Soumis, J. Desrosiers, Classification of travelling salesman problem formulation, Operational Research Letters 9 (1990) 127–132.
[12] S. Lin, B.W. Kernighan, An effective heuristic algorithm for travelling salesman problem, Operations Research (1973) 498–516.
[13] J. Lysgaard, Cluster based branching for the asymmetric traveling salesman problem, European Journal of Operational Research 119 (1999) 314–325.
[14] P. Miliotis, Using cutting planes to solve the symmetric travelling salesman problem, Mathematical programming 15 (1978) 177–188.
[15] A. Mingozzi, L. Bianco, A. Ricciardelli, Dynamic programming strategies for the TSP with time windows and precedence constraints, Operations Research 45 (1997) 365–377.
[16] J.Y. Potvin, Genetic algorithms for the travelling salesman problem, Annals of Operations Research 63 (1996) 339–370.
[17] J. Renaud, F.F. Boctor, J. Ouenniche, A heuristic for the pickup and delivery traveling salesman problem, Computers and Operations Research 27 (2000) 905–916.
[18] M.W.P. Savelsbergh, M. Sol, The general pickup and delivery problem, Transportation Science 29 (1995) 17–29.
[19] B. Shirrish, J. Nigel, M.R. Kabuka, A boolean neural network approach for the travelling salesman problem, IEEE Transactions on Computers 42 (1993) 1271–1278.
[20] M.A. Weiss, Data Structures and Algorithm Analysis, Benjamin/Cummings, Redwood City, 1993.
[21] R. Wong, Integer programming formulations of the travelling salesman problem, in: Proceedings of the IEEE International Conference of Circuits and Computers, 1980, pp. 149–152.