PERGAMON

Computers & Industrial Engineering 43 (2002) 299–314

computers & industrial engineering

www.elsevier.com/locate/dsw

Study on multi-stage logistic chain network: a spanning tree-based genetic algorithm approach

Admi Syarif[a,b], YoungSu Yun[a,c], Mitsuo Gen[a,*]

[a]*Department of Industrial and Information Systems Engineering, Ashikaga Institute of Technology, 268 Ohmae-cho, Ashikaga 326-8558, Japan*

[b]*Department of Mathematics, Lampung University, Bandar Lampung 35148, Indonesia*

[c]*School of Automotive, Industrial and Mechanical Engineering, Taegu University, Kyungbook 712-714, South Korea*

Abstract

In recent years, many of the developments in logistics are connected to the need of information of efficient supply chain flow. An important issue in the logistics system is to find the network strategy that can give the least cost of the physical distribution flow. In this paper, we consider the logistic chain network problem formulated by 0–1 mixed integer linear programming model. The design tasks of this problem involve the choice of the facilities (plants and distribution centers) to be opened and the distribution network design to satisfy the demand with minimum cost. As the solution method, we propose the spanning tree-based genetic algorithm by using Prüfer number representation. We design the feasibility criteria and develop the repairing procedure for the infeasible Prüfer number, so that it can work for relatively large size problems. The efficacy and the efficiency of this method are demonstrated by comparing its numerical experiment results with those of traditional matrix-based genetic algorithm and professional software package LINDO.

Keywords: Logistics; Supply chain management; Genetic algorithm; Production/distribution system

1. Introduction

Logistics is often defined as the art of bringing the right amount of the right product to the right place at the right time and usually refers to supply chain problems (Tilanus, 1997). The efficiency of the logistic system is influenced by many factors; one of them is to find the location of facilities to be opened and distribution network strategy in such a way that the customer demand can be satisfied at minimum cost or maximum profit. If the facilities are to distribute product directly to the customers, then single-

* Corresponding author. Tel.: +81-284-62-0605x376; fax: +81-284-64-1071.
E-mail addresses: admi@genlab.ashitech.ac.jp (A. Syarif), joy629@hitel.net (Y.S. Yun), gen@ashitech.ac.jp (M. Gen).

0360-8352/02/$ - see front matter
PII: S0360-8352(02)00076-1

Nomenclature

Indices

I	number of suppliers ($i = 1, 2, \ldots, I$)
J	number of plants ($j = 1, 2, \ldots, J$)
K	number of DCs ($k = 1, 2, \ldots, K$)
L	number of customers ($l = 1, 2, \ldots, L$)

Parameters

a_i	capacity of supplier i
b_j	capacity of plant j
c_k	capacity of DC k
d_l	demand of customer l
s_{ij}	unit cost of production in plant j using material from supplier i
t_{jk}	unit cost of transportation from plant j to DC k
u_{kl}	unit cost of transportation from DC k to customer l
f_j	fixed cost for operating plant j
g_k	fixed cost for operating DC k
W	an upper limit on total number of DCs that can be opened
P	an upper limit on total number of plants that can be opened

Variables

x_{ij}	quantity produced at plant j using raw material from supplier i
y_{jk}	amount shipped from plant j to DC k
z_{kl}	amount shipped from DC k to customer l

$$w_j = \begin{cases} 1, & \text{if production takes place at plant } j \\ 0, & \text{otherwise} \end{cases}$$

$$z_k = \begin{cases} 1, & \text{if DC } k \text{ is opened} \\ 0, & \text{otherwise} \end{cases}$$

stage model is appropriate. On the other hand, if several facilities are to be sited between the suppliers to the customers in order to produce product or act as regional distribution centers (DCs), then the multi-stage model is the appropriate model (Tragantalerngsak, Holt, & Ronnqvist, 1997).

The body of literature on multi-stage logistic problems including to the multi-stage facility location/allocation problems is large, dealing different models, relevant to various situation (Kaufman, Eede, & Hansen, 1977; Lee, 1993). However, in most cases, this problem is classified according to the capacity of the facilities. In the uncapacitated location/allocation problem, it is assumed that the facilities have no limit on capacity (Ro & Tcha, 1984). When the facilities have certain capacity, the problem is referred as a capacitated location/allocation problem. In this problem, a number of potential facilities with a certain capacity, such as service centers, plants, DCs are given and the problem is to assign facilities to the

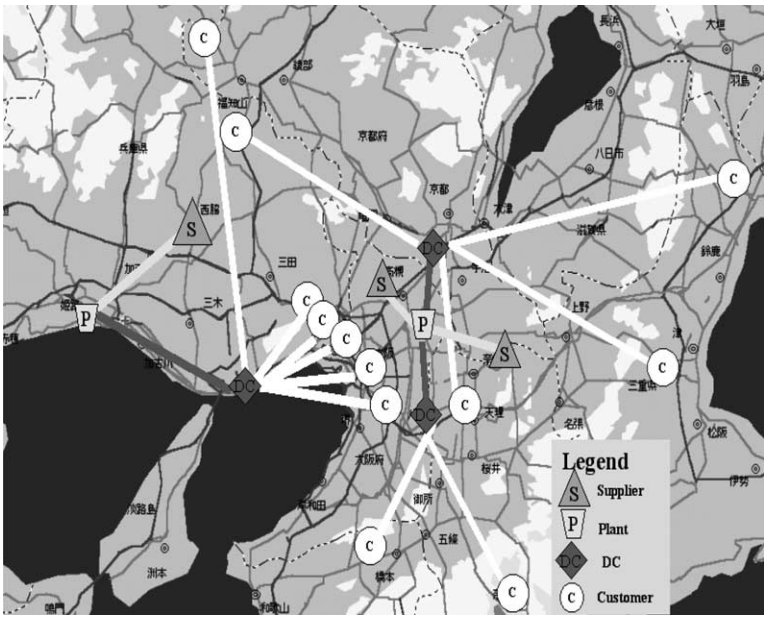

Fig. 1. The illustration of three-stage logistics system.

location in such a way that the sum of the fixed cost of opening facilities and variable cost of transporting the customer demand from facilities is minimized. Pirkul and Jayaraman (1998) studied the two echelon distribution problem with multiple plants and multiple capacitated DCs. The objective is to decide product flow quantity from plants to DCs and from DCs to customers in order to minimize system's overall cost. Azevedo and Sousa (2000) researched an order plan model for multi-stage production system. Such a system processes products and materials through different production units that are part of logistics chain organized in several phases. They try to determine, for each incoming order, an 'optimal' path (concerning cost) through the network. Kim and Lee (2000) discussed the multi-stage production/distribution planning in relation to supply chain management concept with capacitated facilities. The similar problem was also solved by Sim, Jang, and Park (2000) using heuristic method based on Lagrangian relaxation.

In this paper, we consider an extension multi-stage logistic problem by giving the maximum number of facilities (plants and DCs) to be opened as the constraints. This kind of problem can be viewed as the combination of multiple-choice Knapsack problem with the capacitated location–allocation problem simultaneously. So this problem is known to be NP-hard (Gen & Cheng, 1997). With these constraints, the difficulty and the real world applicability of this problem also increase significantly. The design tasks of this problem involve the choice of facilities (plants and DCs) to be opened and the distribution network design to satisfy the customer demand with least cost. We assume that the logistic system process for this problem is organized according to a three-sequence of stages as illustrated by Yu (1997) in Fig. 1.

As the solution method, we proposed a novel technique called spanning tree-based genetic algorithm (st-GA). We adopt the Prüfer number, that is, known to be an efficient way to represent various network problems (Gen & Cheng, 1997) and design the feasibility of the chromosome. To handle the infeasible

chromosome, here, we develop a repairing procedure. With this repairing procedure, we show that this method can be applied for relatively large size problems. The effectiveness and efficiency of the proposed method are demonstrated by comparing numerical experiment results of the proposed method with those of traditional matrix-based genetic algorithm (m-GA) and professional software package LINDO.

The outline of this paper is organized as follows: in Section 2, the mathematical formulation of this problem is given. We describe the feature of our method and its process to solve the problem in Section 3. In Section 4, numerical experiments and comparison with the results of traditional m-GA are given to demonstrate the efficiency of the proposed method. Finally, some concluding remarks are presented in Section 5.

2. Mathematical model

Now we present a comprehensive mathematical model that considers real world factors and constraints. We assume that the numbers of customers and suppliers, as well as their demands and capacities are known in advance. The numbers of potential plants and DCs, as well as their maximum capacities are also known. See Nomenclature for our formulations.

The problem is to choose the subset of plants and DCs to be opened and to design the distribution network strategy that can satisfy all capacities and demand requirements imposed by customers with minimum cost. We formulate the problem by using the following mixed integer linear programming model (MILP):

$$\min \sum_i \sum_j s_{ij} x_{ij} + \sum_j \sum_k t_{jk} y_{jk} + \sum_k \sum_l u_{kl} z_{kl} + \sum_j f_j w_j + \sum_k g_k z_k \tag{1}$$

subject to

$$\sum_j x_{ij} \le a_i, \qquad \text{for all } i \tag{2}$$

$$\sum_k y_{jk} \le b_j w_j, \qquad \text{for all } j \tag{3}$$

$$\sum_j w_j \le P \tag{4}$$

$$\sum_l z_{kl} \le c_k z_k, \qquad \text{for all } k \tag{5}$$

$$\sum_k z_k \le W \tag{6}$$

$$\sum_k z_{kl} \ge d_l, \qquad \text{for all } l \tag{7}$$

$$w_j, z_k = \{0, 1\}, \qquad \text{for all } j, k \tag{8}$$

$$x_{ij}, y_{jk}, z_{kl} \ge 0, \qquad \text{for all } i, j, k, l \tag{9}$$

Constraint (2) ensures the supplier capacities are enough. Constraints (3) and (5) are the capacity constraints for the plants and DCs, respectively. Constraints (4) and (6) ensure that the opened plants and opened DCs do not exceed their upper limits, respectively. Constraint (7) ensures that all demands are met. Without loss of generality, we can assume that this model satisfies the balanced condition, since we can change the unbalanced problem into the balanced one by introducing dummy suppliers or dummy customers.

3. A spanning tree-based genetic algorithm

Recently, there has been an increasing interest in using various evolutionary computation methods to solve hard optimization problems (Gen & Cheng, 1997). Probably, among them, genetic algorithm (GA) is the most well known class of evolutionary algorithms. Other variants of evolutionary algorithm, such as *Genetic Programming*, *Evolutionary Strategies or Evolutionary Programming* are less popular, though very powerful too. Genetic algorithms deal with a coding of the problem instead of decision variables. They require no domain knowledge—only the payoff or objectives for evaluating fitness after operating genetic operations (Goldberg, 1989). In addition, traditional methods use deterministic transition rules to guide the search, such as hill-climbing, neighborhood search. Another difference between traditional methods and genetic algorithms is the latter searches from a set of points, while the former from a single point. These make genetic algorithms more robust than traditional methods regarding their potential as optimization techniques to solve many real world problems (Gen & Cheng, 1997). Michalewicz (1994) was the first researcher, who used GA for solving linear and non-linear transportation/distribution problems. In his method, he represented each chromosome of the problem by using $m \times n$ matrix.

The use of st-GA for solving some network problems was introduced by Gen and Cheng (1997, 2000). They employed Prüfer number in order to represent a candidate solution to the problems and developed feasibility criteria of Prüfer number to be decoded into a spanning tree. The verification for the excellence of the Prüfer number was addressed by Zhou and Gen (1996). They noted that the use of Prüfer number is more suitable for encoding a spanning tree, especially in some research fields, such as: transportation problems, minimum spanning tree problems, and so on. Also, it is shown that we can use only $m + n - 2$ digit number to uniquely represent a distribution network with m suppliers and n destinations, where each digit is an integer between 1 and $m + n$. This means that Prüfer number representation is more efficient in the sense of required memory for computation than that of matrix-based representation. However, for the relatively large size problem, we found that their method cannot work. The reason is because it is very difficult to generate the chromosome that satisfies their feasibility criteria. In this paper, we design simple feasibility check and develop repairing procedure, so that it can be applied for large size problems.

3.1. Representation and feasibility of the chromosome

Genetic algorithm is known as a problem-independent approach, however, the chromosome representation is one of critical issues, when applying it to some optimization problems. Tree-based

representation is known to be one way for representing network problems. Basically, there are three ways of encoding tree:

1. Edge encoding (Gen & Cheng, 1997; Lo & Chang, 2000)
2. Vertex encoding (Gen & Cheng, 1997; Lo & Chang, 2000)
3. Edge-and-vertex encoding (Palmer & Kershenoaum, 1995).

We use here the vertex encoding using Prüfer number representation (Gen & Cheng, 1997). In 1889, Cayley (1889) proved that for a complete graph with p nodes, there are $p^{(p-2)}$ distinct labeled trees. Prüfer (1918) presented the simplest proof of Cayley's formula by establishing a one-to-one correspondence between the set of spanning tree and a set of $p-2$ digit with an integer between 1 and p inclusive (Gen & Cheng, 1997).

For this problem, we use sub-tree $I-J$, sub-tree $J-K$ and sub-tree $K-L$ to represent the distribution pattern for first stage, second stage, and third stage, respectively. Each chromosome in this problem consists of five parts. The first part is J binary digits to represent the opened/closed plants. The second part is K binary digits to represent opened/closed DCs. The last three parts are three Prüfer numbers representing the distribution pattern of each stage, respectively.

As the first step in generating the chromosome, we should generate the two 0–1 variables and check the number of open plants or open DCs for its upper limit constraint. If it exceeds the given upper limit, then close one of the opened plants or opened DCs that has minimum capacity. Also, we should check the total capacities of the opened plants and DCs to satisfy the customer demands. If the total capacity is less than the total demand, then close the facility with minimum capacity and then open the closed facility with maximum capacity. This process is done, until the total capacity of facilities is enough to satisfy the customer demands. The three Prüfer numbers are generated after this step.

For the sub-tree $I-J$, denote the suppliers $1,2,\ldots,I$ as the component of set $S=\{1,2,\ldots,I\}$ and define plants $1,2,\ldots,J$ as the component of the set $D=\{I+1,I+2,\ldots,I+J\}$. Obviously, this distribution graph has $I+J$ nodes, which means that we need $I+J-2$ digits Prüfer number in the range $[1,I+J]$ to uniquely represent this sub-tree $I-J$. When generating the Prüfer number, it is also possible for us to generate an infeasible Prüfer number that cannot be adapted to generate the transportation sub-tree. To this reason, we need to check its feasibility by using the following feasibility condition.

3.1.1. Feasibility check for Prüfer number

Let R_i denotes the number of appearances of node i in the Prüfer number $P(T)$ and L_i denotes the number of connections of the node i for all $i \in S \cup D$, the Prüfer number $P(T)$ is said to be feasible if

$$\sum_{i=1}^{m} L_i = \sum_{i=m+1}^{m+n} L_i \tag{10}$$

The Prüfer number that does not satisfy the above feasibility criteria is called infeasible chromosome. There are two ways to handle with this infeasible chromosome: first is to reject that chromosome and generate a new one until satisfying the above criteria. However, this technique will take a large computational time, especially when the difference of the supplier number and destination number is very large. Another way is to develop the repairing procedure for the infeasible chromosome. Here, we design the

feasibility check and repairing procedure for the Prüfer number to be decoded into spanning tree as follows:

Procedure: feasibility check and repairing procedure for Prüfer number.
Repeat the following steps, until $\sum_{i\in S} L_i = \sum_{i\in D} L_i$ (feasibility condition)

Step 1: Determine R_i for $i \in S \cup D$ from $P(T)$
Step 2: $L_i = R_i + 1$
Step 3: If $\sum_{i\in S} L_i > \sum_{i\in D} L_i$, then select one digit in $P(T)$ which contains node i ($i \in S$) and replace it with the number j ($j \in D$). Otherwise, select one digit in $P(T)$, which contains node i ($i \in D$) and replace it with the number j ($j \in S$) then go to Step 1.

After checking for their feasibility, the Prüfer numbers can be decoded into spanning trees in order to determine the distribution pattern in each stage. The procedures of encoding the network problem into Prüfer number and also decoding Prüfer number into the network graph are given as follows.

Procedure: encoding $I - J$.

Step 1: Let node i be the smallest labeled leaf node in a labeled sub-tree $I - J$.
Step 2: Let j be the first digit in the encoding as the node j incident to node i is uniquely determined. Here, we build the encoding by appending digit to the right, and thus the encoding is built and read from left to right.
Step 3: Remove node i and the link from i to j; thus we have a sub-tree with $I + J - 1$ nodes.
Step 4: Repeat the above steps, until one link is left. We produce a Prüfer number or an encoding with $I + J - 2$ digits between 1 and $I + J$ inclusive.

Procedure: decoding $I - J$.

Step 1: Let P be the original Prüfer number and let $\bar{P}$ be the set of all nodes not included in P, which designates as eligible nodes for consideration in building a tree.
Step 2: Repeat the process (2.1)–(2.5), until no digits left in P.
(2.1) Let i be the smallest label node $\bar{P}$. Let j be the leftmost digit of P.
(2.2) If i and j are not in the same set O or D, add the edge from i to j into the tree. Otherwise, select the next digit k from P that not included in the same set with i. Exchange j with k, add the edge (i, k) to the tree.
(2.3) Remove j (or k) from P and i from $\bar{P}$. If j (or k) does not appear anywhere in the remaining part of P, put it into $\bar{P}$. Designate i as no longer eligible.
(2.4) Assign the available amount of units to $x_{ij} = \min\{a_i, b_j\}$ or $x_{ij} = \min\{a_i, b_k\}$
(2.5) Update the availability $a_i = a_i - x_{ij}$ and $b_j = b_j - x_{ij}$
Step 3: If no digits remain in P, there are exactly two nodes, r and s, still eligible for consideration. Add link from r to s into the tree and form a tree with $I + J - 1$ links.
Step 4: If no available amount of units to assign, then stop. Otherwise, there are remaining supply of node r and demand of node s, add edge (r, s) to tree and assign the available amount of units $x_{rs} = \min\{a_r, b_s\}$ to edge. If there exists a cycle, then remove the edge that assigned zero flow. A new spanning tree is formed with $m + n - 1$ edges.

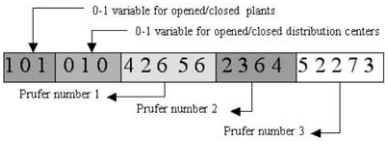

Fig. 2. The illustration of feasible chromosome.

By using similar ways, we can generate two other Prüfer numbers representing $J - K$ and $K - L$ subtrees. These two Prüfer numbers consist of $J + K - 2$ and $K + L - 2$ digits, respectively. The procedures of encoding ($J - K$ and $K - L$) and decoding ($J - K$ and $K - L$) for the second and third stages are also similar. The process of generating the feasible chromosome is done *pop_size* times in order to generate the initial population.

As an example, we consider the problem that has 4 suppliers, 3 feasible plants, 3 feasible DCs and 4 customers. We decide 2 plants and 2 DCs as the upper limit of open plants and open DCs for this problem. Fig. 2 gives an illustration of a feasible chromosome representation.

As defined before, this chromosome consists of five sub-strings. The first and second sub-strings are 3 binary digits representing opened/closed plants and DC, respectively. The last three sub-string are Prüfer numbers consist of 5, 4 and 5 digits to represent the distribution pattern for each stage, respectively. The above representation shows that two plants and one DC are opened. When decoding the chromosome, firstly we should change the capacity of the closed facilities to be zero. Then, after generating the Prüfer numbers and checking their feasibility, we can decode them by using the decoding procedure to find the distribution pattern of this chromosome and compute the fitness value of the chromosome. Here, we give the step-by-step for decoding of the first Prüfer number in the above representation as follows: Firstly, for the Prüfer number $P = [4\ 2\ 6\ 5\ 6]$, we have $\bar{P} = [1\ 3\ 7]$. Node 1 is the smallest node number in $\bar{P}$ and Node 4 is left most digit of P. However, since these two nodes are in the same set, we select the next node in P, which is not in the same set (node 6). Add edge (1, 6) to the tree, remove node 1 from $\bar{P}$ and node 6 from P leaving $P = [4\ 2\ 5\ 6]$ and $\bar{P} = [3\ 7]$. Assign $x_{16} = \min\{a_1, b_6\} = 0$. Update $a_1 = a_1 - x_{16}$ and $b_6 = b_6 - x_{16}$. Secondly, node 3 is the lowest number in $\bar{P}$ and node 5 is left most digit of P, which is not in the same set. Add edge (3, 5) to the tree, remove node 3 from $\bar{P}$. Since node 5 is no longer appear in the remaining part of P, add it to $\bar{P}$. $P = [4\ 2\ 6]$ and $\bar{P} = [5\ 7]$. Assign $x_{35} = \min\{a_3, b_5\} = 7$. Update

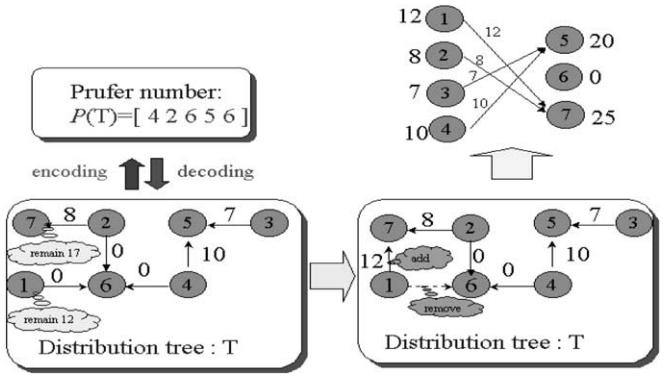

Fig. 3. The illustration of Prüfer number decoding procedure.

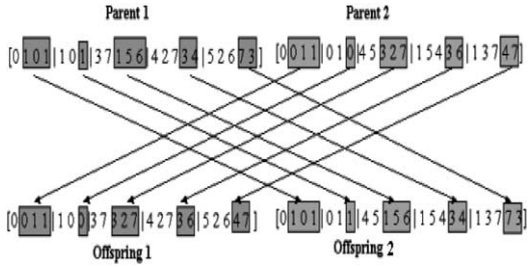

Fig. 4. The illustration of one point crossover process.

$a_3 = a_3 - x_{16} = 0$ and $b_5 = b_5 - x_{16} = 13$. Repeat this process until finally P is empty and only node 2 and 7 in $\bar{P}$. Add edge (2, 7) to the tree. Since there is still available source in node 7 and demand in node 1, we add the edge (1, 7) and remove node (1, 6). Fig. 3 shows the illustration of the Prüfer number and its spanning tree. The decoding procedures of the second and third Prüfer numbers are similar.

3.2. Genetic operations

3.2.1. Crossover

The crossover is done by exchanging the information of two parents to provide a powerful exploration capability. We begin with defining a parameter P_c of an evolutionary system as the probability for crossover. This probability gives us the expected number $p_c \cdot pop\_size$ of chromosomes that undergo the crossover operation. First, we generate a random real numbers in the range of [0, 1]; the first chromosome will go for crossover if $r < P_c$. By repeating this process *pop_size* times for next chromosomes, we shall averagely generate $p_c \cdot pop\_size$ offsprings. In this paper, we employ a one-cut-point crossover operation, which randomly selects a one-cut-point and exchanges the right parts of two parents to generate offspring. Prüfer numbers resulted by this crossover operation is always feasible in the sense that they can be decoded into a corresponding transportation tree due to the feasibility criteria $\sum_{i \in S} L_i = \sum_{i \in D} L_i$ is unchanged after these operations. However, before decoding all Prüfer numbers, we still need to check whether the capacity of the opened facilities is enough to satisfy the demand or not. If it is not enough, we use the similar procedure as in the initialization to satisfy the demand. Fig. 4 shows the one-point crossover process.

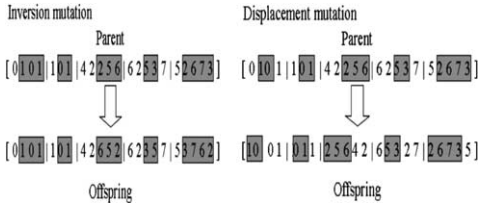

Fig. 5. The illustration of the mutation processes.

3.2.2. Mutation

Modifying one or more of the gene values of an existing individual, mutation creates new individual to increase the variability of the population. We use inversion and displacement mutation operations. The inversion mutation selects two positions within a chromosome at random and then inverts the sub-string between these two positions. The displacement mutation selects a sub-string at random and inserts it in a random position. This two mutation operators will also generate feasible Prüfer numbers if the parents are feasible. However, we also need to check the feasibility of the opened facilities to satisfy the demand. The same process is used, when the capacity of the opened facilities in the offspring is not enough to satisfy the demand. Fig. 5 shows the illustration of those mutation processes.

3.3. Evaluation and selection

As in nature, it is necessary to provide driving mechanism for better individuals to survive. Evaluation is to associate each chromosome with a fitness value that shows how good it is based on its achievement of the objective function. The higher fitness value of an individual, the higher its chances for survival for the next generation. Selection is to select individuals to be parents in the next generation according to their fitness value. So the evaluation and selection play a very important role in the evolutionary process. For this problem, the evaluation procedure can operate as follows:

Procedure: evaluation.

Step 1: Convert Prüfer numbers into spanning trees
Step 2: Calculate the total cost and the total fixed cost of trees according to the objective function
Step 3: Repeat the procedure for all individuals.

As to selection, we adopt the elitist selection strategy (Gen & Cheng, 1997) in which duplicate chromosomes are prohibited.

3.4. Overall procedure of the proposed method

In this section, we show the overall procedure for solving the problem as follows:

Step 1: Set the initial values and the parameters of genetic algorithms:
Set population size *pop_size*, crossover rate p_c, mutation rate p_m, and maximum generation *max_gen*.
Step 2: Generate the initial population
Generate feasible chromosome following procedure given in Section 3.1. Repeat this process *pop_size* times.
Step 3: Genetic operators
(1) Crossover
(2) Mutation
Step 4: Check the feasibility of the offspring and repair the infeasible offspring.
Step 5: Evaluation
Calculate the evaluation value for the offspring.
Step 5: Selection

Table 1
The size of test problems

Problem	Number of suppliers, m	Number of plants, n	Number of DCs, K	Number of customers, L	Number of maximum opened plants, P	Number of maximum opened DCs, W
1	3	5	5	4	4	4
2	10	10	10	21	6	6
3	20	15	12	50	9	7
4	10	6	8	100	4	5

Elitist strategy

Step 6: Termination condition

Increase the generation number. If it is less than or equal to *max_gen*, then go to Step 3.

Step 7: Output the solution

4. Numerical examples

We implement our proposed method using Visual C language. The performance of the proposed method is tested by using four different size of test problems as given in Table 1. To see the efficiency and the effectiveness of the method, we also develop m-GA based on Michelewicz's approach by using the same program language. We also use LINDO to get optimal solution for relatively small size problem. Both st-GA and m-GA were run on the Pentium 700 PC with the same GA parameters $P_c = 0.4$ and $P_m = 0.2$. To have more information about these algorithms, we divided each test problem into three numerical experiments by giving different size of *pop_size* and *max_gen*. Each of numerical experiment is 10 times. We note its best, worst and average results.

At first, we begin with small size problem. For this problem, the capacities, demands, fixed costs and shipping costs for each stage are given in Tables 2 and 3. We set the maximum numbers of opened plants and opened DCs to be 4.

In Fig. 6, we show the best distribution pattern for this problem given by our proposed method. The average objective value for both methods in the generation is illustrated in Fig. 7. From those results, it can be seen that the best solution is reached by opening only 3 plants and 3 DCs though it is allowed to open 4 plants and 4 DCs.

Next, we also have the numerical experiments for relatively larger size problems as in the test problem 2, 3, and 4. We developed all capacities and demands in considering all constraint in the model. The distribution cost for these problems are randomly generated integer number in the range [4,9]. We summarize the results of all of the test problems in the following Tables 4 and 5.

It is shown in the above tables that, for small size problem, our proposed method can give the optimal solution in all of the time and has better average solution than that of m-GA. For relatively large-scale problems, our proposed algorithm can give better heuristic solution than m-GA. It is also shown that our proposed method has better computational time and memory required for computation. However, since the search space for this kind of problem is so large, it is very important to set the experiment with reasonable population size and maximum generation in order to ensure that we have a good result.

Table 2
Capacity, demand and fixed cost for the test problem 1

Supplier capacity	Plant		DC		Customer demand
	Capacity	Fixed cost	Capacity	Fixed cost	
500	400	1800	530	1000	460
650	550	900	590	900	330
390	490	2100	400	1600	450
	300	1100	370	1500	300
	500	900	580	1400	

Table 3
Shipping cost value for each stage for the test problem 1

Source	Plants				
	1	2	3	4	5
1	5	6	4	7	5
2	6	5	6	6	8
3	7	6	3	9	6
Plant	DC				
	1	2	3	4	5
1	5	8	5	8	5
2	8	7	8	6	8
3	4	7	4	5	4
4	3	5	3	5	3
5	5	6	6	8	3
DC	Customer				
	1	2	3	4	
1	7	4	5	6	
2	5	4	6	7	
3	7	5	3	6	
4	3	5	6	4	
5	4	6	5	7	

5. Conclusion

In this paper, we proposed a st-GA approach to find the best production/distribution design in multi-stage logistics system. We utilize the Prüfer number that is known to be an efficient way to represent various network problems. Even though the structure of the proposed methods is very simple, experimental results show that our algorithm not only can give better heuristic solutions in almost all of the

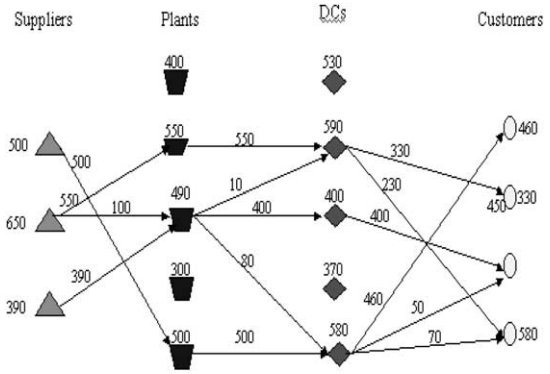

Fig. 6. Illustration of the optimal distribution pattern.

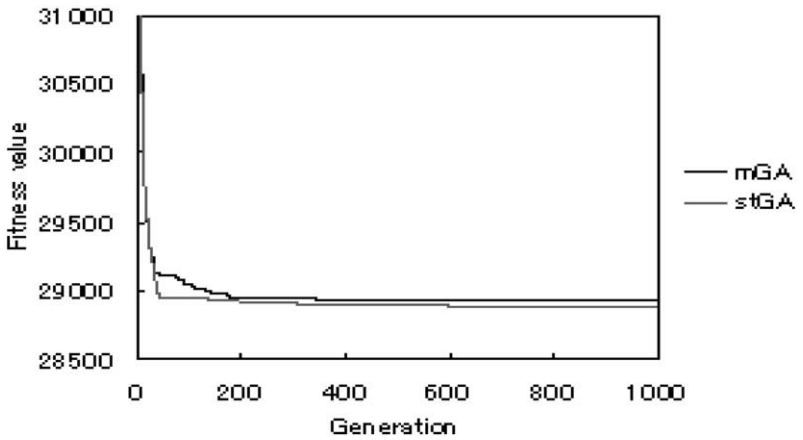

Fig. 7. The average fitness value in the generation.

time, but also has better performance in the sense of computational time and required memory for computation than those of m-GA. For relatively small size problem, we show that our proposed method can search the optimal solution in almost all of the time. So we believe this method will be an efficient and robust method to solve this kind of multi-stage logistic chain design problems.

Acknowledgements

This research was supported by the International Scientific Research Program, the Grant-in-Aid for Scientific Research (No. 10044173) by the Ministry of Education, Science and Culture, the Japanese Government. The work of the first author was also supported by Development Undergraduate Education Project, The University of Lampung, Indonesia (Indonesian Government and World Bank Project). The authors also would like to thank the reviewer for his valuable comments and suggestions on this paper.

Table 4
Comparative result of the test problems using m-GA and st-GA

Problem	*pop_size*	*max_gen*	m-GA			st-GA			Optimal solution[a]
			Best	Worst	Mean	Best	Worst	Mean	
1	25	750	28 870	28 960	28 942	28 870	28 960	28 920	
	30	1000	28 870	28 960	28 924	28 870	28 920	28 875	
	50	1500	28 870	28 960	28 915	28 870	28 870	28 870	28 870
2	200	2000	35 316 233	36 205 426	35 658 718	34 578 733	36 000 837	35 644 650	
	250	2500	34 647 365	36 077 077	35 542 572	34 148 415	36 081 877	35 379 480	
	300	3000	33 866 577	35 935 038	35 511 341	33 118 733	35 924 136	34 626 132	33 118 733
3	200	3000	603 110	615 257	603 516	601 510	611 273	602 347	
	250	4000	601 269	609 632	601 962	600 195	607 195	601 237	
	300	6000	600 217	603 748	601 094	594 555	601 739	600 336	593 175
4	200	4000	1 011 680	1 012 735	1 012 039	1 010 951	1 012 417	1 011 912	
	250	6000	1 011 379	1 012 637	1 011 750	1 010 188	1 012 321	1 011 549	
	300	10 000	1 011 057	1 012 172	1 011 151	1 010 141	1 011 988	1 010 957	

[a] Given by LINDO.

Table 5
Computational time and memory used by m-GA and st-GA (ACT, average computational time in second; memory, required unit memory space to represent the chromosome)

Problem	*pop_size*	*max_gen*	m-GA		st-GA	
			ACT	Memory	ACT	Memory
1	25	750	1.6	60	1.5	21
	30	1000	2.7	60	2.6	21
	50	1500	5.8	60	5.7	21
2	200	2000	74.8	410	67.5	75
	250	2500	103.6	410	93.3	75
	300	3000	216.4	410	167.2	75
3	200	3000	467.8	1080	259.3	118
	250	4000	579.6	1080	322.4	118
	300	6000	947.5	1080	546.3	118
4	200	4000	606.7	908	384.2	132
	250	6000	1235.6	908	685.7	132
	300	10 000	2896.4	908	1468.3	132

References

Azevedo, A. L., & Sousa, J. P. (2000). Order planning for networked make-to-order enterprises—a case study. *Journal of Operational Research Society*, *51*, 1116–1127.

Cayley, A. (1889). A theorem on tree. *Quarterly Journal of Mathematics*, *23*, 376–378.

Gen, M., & Cheng, R. (1997). *Genetic algorithms and engineering design*, New York: Wiley.

Gen, M., & Cheng, R. (2000). *Genetic algorithms and engineering optimization*, New York: Wiley.

Goldberg, D. (1989). *Genetic algorithm in search, optimization and machine learning*, Reading, MA: Addison-Wesley.

Kaufman, L., Eede, M. V., & Hansen, P. (1977). A plant and warehouse location problem. *Operational Research Quarterly*, *28*, 547–554.

Kim, H. K., & Lee, Y. H. (2000). Production/distribution planning in SCM using simulation and optimization model. *Proceedings of Korean Supply Chain Management Society*.

Lee, C. Y. (1993). A cross decomposition algorithm for a multi-product multi-type facility location problem. *Computer and Operations Research*, *20*, 527–540.

Lo, C. C., & Chang, W. H. (2000). A multi-objective hybrid genetic algorithm for the capacitated multipoint network design problem. *IEEE Transaction on System, Man and Cybernetic*, *30* (3), 461–470.

Michalewicz, Z. (1994). *Genetic algorithms + data structures = evolution programs*, (2nd ed). New York: Springer.

Palmer, C. C., & Kershenoaum, A. (1995). An approach to a problem in network design using genetic algorithms. *Networks*, *26*, 151–163.

Pirkul, H., & Jayaraman, V. A. (1998). Multi-commodity, multi-plant, capacitated location allocation problem: Formulation and efficient heuristic solution. *Computer and Operation Research*, *25* (10), 869–878.

Prüfer, H. (1918). Neuer bewis eines saizes uber permutationen. *Archives of Mathematical Physics*, *27*, 742–744.

Ro, H., & Tcha, D. (1984). A branch and bound algorithm for two level uncapacitated facility location problem with some side constraint. *European Journal of Operational Research*, *18*, 349–358.

Sim, E., Jang, Y., & Park, J. (2000). A study on the supply chain network design considering multi-level, multi-product, capacitated facility. *Proceedings of Korean Supply Chain Management Society*.

Tilanus, B. (1997). Introduction to information system in logistics and transportation. (pp. 7–16). In B. Tilanus, *Information systems in logistics and transportation*, Amsterdam: Elsevier.

Tragantalerngsak, S., Holt, J., & Ronnqvist, M. (1997). Lagrangian heuristics for the two-echelon, single-source, capacitate location problem. *European Journal of Operational Research*, *102*, 611–625.

Yu, H. (1997). ILOG in the supply chain. *ILOG Technical Report*.

Zhou, G., & Gen, M. (1996). A note on genetic algorithms approach to the degree-constrained spanning tree problem. *Networks*, *30*, 105–109.