Optim Eng (2009) 10: 219–237
DOI 10.1007/s11081-008-9059-x

A new spanning tree-based genetic algorithm for the design of multi-stage supply chain networks with nonlinear transportation costs

Ming-Jong Yao · Hsin-Wei Hsu

Received: 30 September 2008 / Accepted: 20 October 2008 / Published online: 6 November 2008
© Springer Science+Business Media, LLC 2008

Abstract The design of configuration and the transportation planning are crucial issues to the effectiveness of multi-stage supply chain networks. The decision makers are interested in the determination the optimal locations of the hubs and the optimal transportation routes to minimize the total costs incurred in the whole system. One may formulate this problem as a 0-1 mixed integer non-linear program though commercial packages are not able to efficiently solve this problem due to its complexity. This study proposes a new spanning tree-based Genetic Algorithm (GA) using determinant encoding for solving this problem. Also, we employ an efficient heuristic that fixes illegal spanning trees existing in the chromosomes obtained from the evolutionary process of the proposed GA. Our numerical experiments demonstrate that the proposed GA outperforms the other previously published GA in the solution quality and convergence rate.

Keywords Genetic algorithm · Non-linear transportation costs · Multi-stage supply chain networks · Spanning tree

1 Introduction

A supply chain is a network that consists of suppliers, manufacturing sites, distribution centers (DCs) and customer locations through logistics. Logistics is often defined

M.-J. Yao (✉)
Department of Transportation Technology and Management, National Chiao Tung University, Taiwan, ROC
e-mail: myaoie@gmail.com

H.-W. Hsu
Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Taiwan, ROC
e-mail: d9534802@oz.nthu.edu.tw

as the art of bringing the right amount of right products to the right place (Tilanus 1997). Due to the criticality of logistic systems, researchers and practitioners had addressed lots of efforts to the plan and management in supply chain networks (Marry and Vidyaranya 2005). Also, many factors influence the efficiency of the logistic system, and one of the most important issues is to determine the locations of plants and distribution centers so as to minimize costs or maximize profit and satisfy the customer demand in the supply chain. Tragantalerngsak et al. (1997) commented that the multi-stage model fits the decision making scenario for the cases where several plants should be sited between the suppliers to the customers in order to provide products or act as a distribution center.

In the literature, many researchers had addressed their efforts to solve the problem of designing a supply chain network. One may refer to Simchi-Levi et al. (2003) and our literature in Sect. 2 for the details. Some of them applied heuristic for solving the problem, e.g., Geoffrion and van Roy (1979). Most of studies recommended mixed integer programming (MIP) model for solving the problem. However, as Bramel and Simchi-Levi (1997) commented in their book, it is generally very difficult to obtain the optimal solution for these MIP models. Recently, Syarif et al. (2002) applied the concept of spanning tree to the design of multi-stage supply chain networks and utilized the characteristic of spanning tree to set up the encoding (viz. Prüfer encoding) in their genetic algorithm (GA) for solving the problem. Syarif et al. (2002) cared about the topics of the places of manufacturing sites and DCs should be opened. In this paper, we would like to extend their study by accommodating more complex decision-making scenarios (e.g., taking into accounts the fixed cost for subcontracting the transportation vehicles operating between any two neighboring stages) in the multi-stage supply chain network. In Sect. 3, we will show that the revised mathematical model may be no longer a linear, binary integer program and it is unable to be solved by commercial optimization packages, e.g., LINGO or CPLEX, etc.

In order to solve the supply chain design problem in this study, we propose a new spanning tree-based GA using different encoding. We note that the GA using Prüfer encoding in Syarif et al. (2002) still applies to solve the concerned problem in this study. Our numerical experiments will show that the proposed GA outperforms Syarif et al. (2002) solution approach.

The rest of this paper is organized as follows. Section 2 reviews the literature on the design of supply chain networks. Section 3 presents the mathematical model for the concerned problem in this study. Section 4 provides the details on the proposed GA. The numerical experiments in Sect. 5 demonstrate that the proposed GA obtains better solutions than Syarif et al. (2002). Finally, Sect. 6 gives the concluding remarks.

2 Literature review

In order to face the challenge from the global market, larger enterprise groups pay intensive attention to the design of supply chains since it grows to be one of the most important factors to gain advantage over other competitors. The management of flow of goods (including work-in-process and finished items) is one of the major concerns

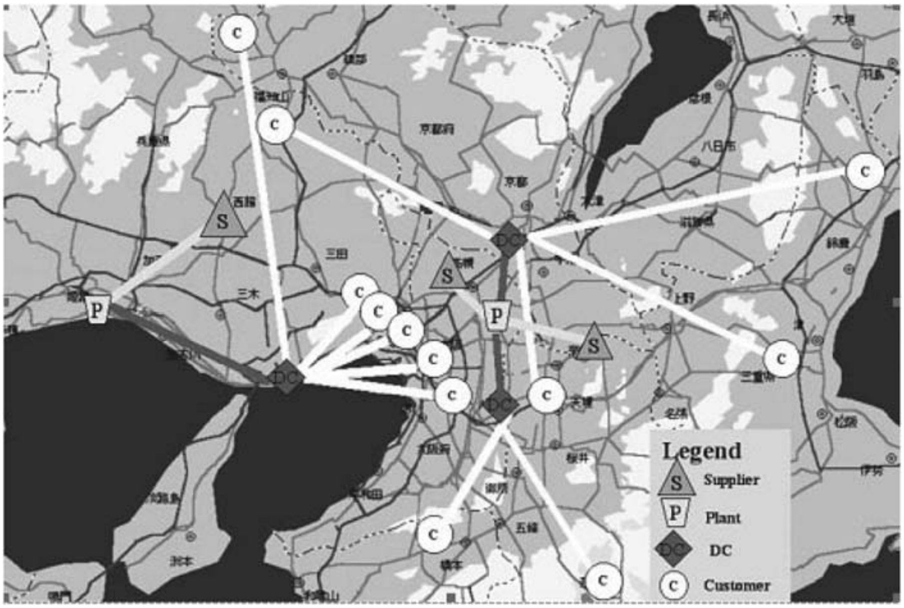

Fig. 1 Three-stage logistics system by Yu (1997)

to improve the efficiency of the supply chain to meet the demand with a minimum cost, or to fill demand for maximum profit. Therefore, the modeling and analysis of multi-stage supply chains have been a very prosperous area of research in the past two decades.

One may refer to Ro and Tcha (1984) and Pirkul and Jayaraman (1998) for the early studies on multi-stage logistic systems. An order plan model for a multi-stage production system was introduced by Azevedo and Sousa (2000). Such a system processes products and materials through different production units that are part of a logistics chain organized in several phases. In this case, they tried to determine, for each incoming order, an optimal path (concerning cost) through the network. Discussion of the production/distribution planning in relation to the supply chain management concept was given by Sim et al. (2000).

Before this study, Syarif et al. (2002) proposed a spanning tree-based GA to determine to choice the right locations for opening the plants and DCs and calculating minimum cost from suppliers to manufacturers, manufacturers to DCs and DCs to customers. One may refer to Fig. 1 (illustrated by Yu 1997) for the operations in a three-stage supply chain. We note that the mathematical model was formulated according to a simple and straightforward decision-making scenario. Therefore, their mathematical model is a linear, binary integer program that can be solved by LINGO or other commercial optimization packages. Later, Yeh (2005) proposed a revised mathematical model that corrected the fatal error appearing in Syarif et al. (2002) models. He also brought up with a brilliant and efficient Hybrid Heuristic Algorithm (HHA) for solving the revised model. His proposed HHA combines a greedy method, the linear programming technique and three local search methods (namely, a pair-wise exchange procedure, an insert procedure and a remove procedure). The computational results in his study demonstrate his HHA is very efficient in solving the problem with high quality solutions. We note that Yeh's (2005) HHA only applies to the models with linear cost terms since it needs to employ the linear programming technique, which serves as the most critical part in the HAA, to improve candidate solutions during its optimization process. Therefore, it is *not* applicable to the supply chain network design problem with nonlinear transportation costs concerned in this study.

Note that GA had been popularly applied to solve several NP-hard network optimization problems; e.g., Chou et al. (2001) applied GA to solve the degree-constrained minimum spanning tree problem presented in Narula and Ho (1980), Delbem et al. (2005) employed GA to solve the network reconfiguration problem in power distribution systems, and Zhou and Gen (2003) applied GA to solve the telecommunication network design problem. Gen et al. (2005) utilized GA for solving the following four network design problems: (1) degree-constrained minimum spanning tree problem, (2) capacitated minimum spanning tree problem, (3) fixed charge transportation problem, and (4) bicriteria local area network topological design problem. Also, Jo et al. (2007) employed GA for solving the nonlinear fixed charge transportation problem in two-stage supply chain networks (with the transshipment happening only between the plants and the customers), which may be recognized as a special case or a simplified version of the problem studied in this paper.

Gen and Cheng (1997) commented that there are two types of encoding, namely, direct and indirect, when employing GA to solve network optimization problems. With direct encoding, the strings can be translated directly. On the other hand, with indirect encoding, a decoding algorithm should be used to expand the strings into meaningful information for evaluation. Spanning trees have been used extensively in a variety of network optimization problems and various encoding methods have been used to represent trees. They can be classified broadly into three categories: edge, node, and edge-node encoding. Edge encoding has been found to be a poor representation, and in node-based or vertex-based encoding the nodes, rather than edges, are represented in the encoding. A popular encoding method for trees is based on Prüfer number, which represents a tree of n nodes with $n-2$ digits, where each digit is an integer between one and n. Syarif et al. (2002) adopted Prüfer encoding to solve the supply chain network problems. Since Prüfer encoding may lead to infeasible chromosomes, they also proposed a feasibility check and repairing procedure for not only decoding Prüfer representation and fix any infeasible chromosome. Note that the evolutionary process in their GA is done with a one-point crossover and inversion mutation. Besides, Gen et al. (2005) and Jo et al. (2007) also employed Prüfer encoding for solving network design problem in supply chains. These studies recommended Prüfer encoding since it may obtain solutions which are better than other GAs, e.g., matrix-based encoding GAs.

On the other hand, Abuali et al. (1995) suggested determinant encoding as an alternate node-based encoding for representing spanning trees, and compared Prüfer, determinant, and link and node biased encoding methods and found that determinant encoding provided better performance. Chou et al. (2001) showed that determinant encoding is superior to Prüfer encoding when solving the degree-constrained minimum spanning tree problem.

Inspiring from the results of Chou et al. (2001), we tried to apply determinant encoding for solving the concerned problem. Also, we propose an efficient heuristic to fix illegal spanning trees existing in the chromosomes obtained from the evolutionary process. We will present the details on the proposed GA in Sect. 4.

3 The mathematical model

In this section, we present three models for the problem of designing a supply chain network.

3.1 The basic model

First, we would like to introduce the basic model presented in Syarif et al. (2002). Before presenting the revised model, we first introduce the nomenclature used later.

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
f_j: fixed cost for operating plant j
g_k: fixed cost for operating DC k
s_{ij}: unit cost of transportation 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
P: an upper limit on total number of plants that can be opened
W: an upper limit on total number of DCs that can be opened

Variables

x_{ij}: amount shipped from supplier i to plant j
y_{jk}: amount shipped from plant j to DC k
z_{kl}: amount shipped from DC k to customer l

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

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

where $\lceil x \rceil$ denotes the ceiling operator that obtains the smallest integer larger than x. We present the mixed binary integer linear program in Syarif et al. (2002) as follows. Note that we call it as the problem (P$_0$) in this paper.

$$(\mathrm{P}_0)\quad \text{Minimize } TC = \sum_i \sum_j s_{ij} x_{ij} + \sum_j \sum_k t_{jk} y_{jk} + \sum_k \sum_l u_{kl} z_{kl} + \sum_j f_j p_j + \sum_k g_k w_k \tag{1}$$

$$\text{subject to} \quad \sum_j x_{ij} \le a_i \quad \text{for all } i \tag{2}$$

$$\sum_k y_{jk} \le b_i p_j \quad \text{for all } j \tag{3}$$

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

$$\sum_l z_{kl} \le c_k w_k \quad \text{for all } k \tag{5}$$

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

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

$$p_j, w_k = \{0, 1\} \quad \text{for all } j, k \tag{8}$$

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

The objective function (1) is the sum of the fixed costs for operating plants and DCs, the transportation costs between any two neighboring echelons in the multi-stage supply chain network. Constraints in (2) ensure that the suppliers' capacity is sufficient. Constraints (3) and (5) are the capacity constraints for the plants and DCs, respectively. Constraints (4) and (6) indicates that the opened plants and DCs do not exceed their upper limits, respectively. Constraints in (7) make sure that the demands of all customers are met.

3.2 An extended version

Here, we would like to present an extended version of Syarif et al. (2002) model. We derive this revised model, which will be named as the problem (P$_1$), by taking into accounts the fixed costs for subcontracting the transportation vehicles in the multi-stage supply chain network. When formulating this revised model, we adopt all the assumptions in Syarif et al. (2002) but employ two different assumptions as follows.

1. We assume that the operations of transportation will be subcontracted to a third party logistic (3PL) service provider. According to the cases in the real world, the 3PL service provider charges the transportation cost by taking into account

not only the unit cost for each shipped item, but also the fixed cost for assigned designated vehicles for the transportation between two neighboring echelons in a supply chain network.
2. We allow unlimited number of opened plants and DCs in the multi-stage supply chain network.

We define some notation before discussing the details on calculating the cost for subcontracting 3PL service provider. First, we define *FC* as the fixed cost for subcontracting a transportation vehicle and *VL* as the capacity limit of a transportation vehicle. Suppose that the total amount of shipment between two neighboring echelons in a supply chain network is *TA*. Following the first assumption above, the 3PL service provider will charge a fixed cost for subcontracting the transportation vehicles operating between the two neighboring echelons, which is given by

$$FC \cdot \left\lceil \frac{TA}{VL} \right\rceil \tag{10}$$

Following our second assumption, we remove the following two constraints, namely, $\sum_j p_j \le P$ and $\sum_k w_k \le W$ from Syarif et al. (2002) model. We consider that it would be more reasonable since we have included the fixed costs for operating plants and DCs in the objective function and the optimization process automatically determines the values of the upper bounds (i.e., P and W).

The mixed integer nonlinear program for the problem (P_1) is presented as follows.

(P_1) Minimize

$$\begin{aligned} TC = {} & \sum_i \sum_j s_{ij} x_{ij} + \sum_j \sum_k t_{jk} y_{jk} + \sum_k \sum_l u_{kl} z_{kl} + \sum_j f_j p_j \\ & + \sum_k g_k w_k + FC \cdot \left[\left\lceil \frac{\sum_{x_{ij} \ne 0} x_{ij}}{VL} \right\rceil + \left\lceil \frac{\sum_{y_{jk} \ne 0} y_{jk}}{VL} \right\rceil \right. \\ & \left. + \left\lceil \frac{\sum_{z_{kl} \ne 0} z_{kl}}{VL} \right\rceil \right] \end{aligned} \tag{11}$$

subject to (2), (3), (5), (7), (8), and (9).

We note that after adding the last cost term, the objective function in our mathematical model turns to a nonlinear function. Since the problem (P_1) is not a mixed integer linear program, we are no longer able to guarantee obtaining the optimal solution using commercial software, e.g., LINGO or CPLEX as the problem (P_0) did.

3.3 More complicated versions

We note that it could be very complicated to determine the transportation cost in the supply chain. We would use a case in Fig. 2 as an example for our discussion here.

We assume that there are three companies that take care of the transportation operations in the supply chain network in Fig. 2. Each company covers only a particular part of the map possibly because of the geometric reasons and the economic reasons

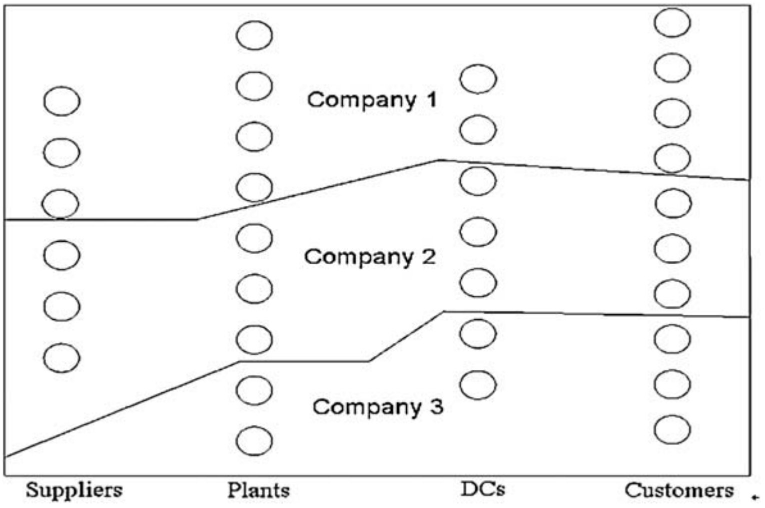

Fig. 2 A map for the 3PL companies responsible for the transportation

when establishing its transportation routes. Note that the transportation charge could be complicated in the real world. For example, the company may offer a price table with all-unit discount, incremental discount or a price table with indifference points, see Russell and Krajewski (1991) and Siajadi et al. (2005) for details. We note that it is almost impossible to learn the exact objective function value before one determines the values of the decision variables in the mathematical model. The source of such difficulty results from the total transportation costs paid to 3PL companies. Especially, one has to know the values of $x_{ij}, y_{jk}, z_{kl}, \forall i, j, k, l$ to determine the total transportation costs paid to a 3PL company c, which is denoted as $\tau_c(x_{ij}, y_{jk}, z_{kl}|\forall i, j, k, l)$. Then, we may have a complicated mixed integer nonlinear program such a case, which is named as the problem (P_2) as follows.

$$(P_2) \quad \text{Minimize } TC = \sum_j f_j p_j + \sum_k g_k w_k + \sum_c \tau_c(x_{ij}, y_{jk}, z_{kl}|\forall i, j, k, l) \qquad (12)$$

subject to (2), (3), (5), (7), (8), and (9).

Even though the problem (P_2) is a very complicated problem, GA may serve as an appropriate solution methodology since GA has the values of all the decision variables in the mathematical model during the evolutionary process.

4 The genetic algorithm

In this section, we propose a new spanning tree-based GA for solving the design of a multi-stage supply chain. The chromosome representation, the proposed heuristic that fixes the feasibility problem for the illegal chromosomes and the genetic operators in the proposed GA will be introduced in the following subsection.

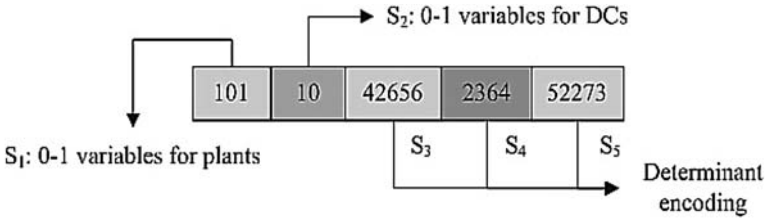

Fig. 3 The illustration of chromosome representation in the proposed GA

4.1 Chromosome representation and determinant encoding

In the problem of designing a supply chain network, each chromosome S in the proposed GA is a composition of five ordered sub-strings, $S_1, \ldots, S_5$. Namely, the first sub-string S_1 represents the decisions if the sites of plants should be opened, and the second sub-string S_2 represents the decisions if the sites of DCs should be opened. Also, the last three sub-strings, S_3, S_4, S_5 use determinant encoding to represent the transportation patterns between the suppliers and the plants, that between the plants and the DCs, as well as that between the DCs and the customers, respectively. Here, we take a small-size supply chain that has 3 supplies, 3 feasible plants, 2 feasible DCs and 4 customers as an example (which is depicted in Fig. 3) to illustrate our chromosome representation in the proposed GA.

Suppose that there are I suppliers, J plants, K DCs and L customers in the supply chain. Next, we will have further discussion on determinant encoding. We first take the transportation for the stage between the I suppliers and J plants as an example. (The determinant encoding for the transportation between the J plants and K DCs as well as that between K DCs and L customers are done in the same fashion.) There are a total of $I + J$ nodes considered in this stage. Then, the length of the substring S_3 using determinant encoding corresponding to this stage should be $(I + J - 1)$. (We have further discussions on the reasons why there should be $(I + J - 1)$ links for this stage in the Appendix.) Similarly, the length of the substrings S_4 and S_5 should be $(J + K - 1)$ and $(K + L - 1)$, respectively.

The decoding algorithm for determinant encoding treats each allele of chromosome to correspond to its position in the chromosome and the position represents its direct connecting node. The first gene is decoded as fixed-position 2, second as fixed-position 3, and so on. Using a substring S_g $(g = 3, 4, 5)$ with its length being λ, we elaborate the decoding for the determinant encoding in the following discussion. Denote S_q as the qth allele in substring S_q and $1 \le q \le \lambda$. Recall that the number of the nodes in this stage should be $\lambda + 1$. The decoding of a substring S_g using determinant encoding is done by the following procedure.

The decoding procedure for the substrings using determinant encoding

1. Set $q = 1$, if $0 < q \le \lambda + 1$, go to Step 2, else Stop.
2. Connect node $(q + 1)$ with node $S_g(q)$, set $q = q + 1$, go back to Step 1.

Next, we would like to show that there could be three types of illegal spanning tree generated from determinant encoding if one did not set any restriction on the range of

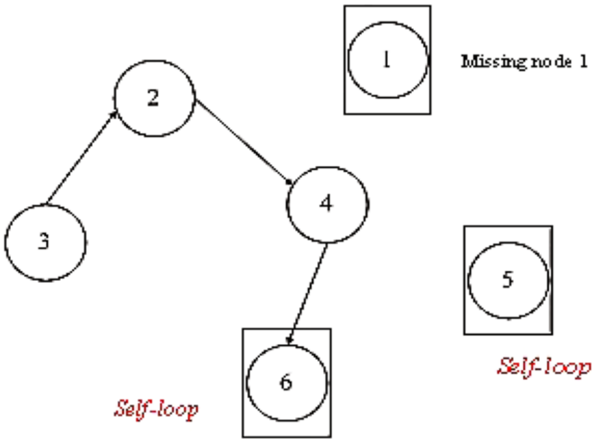

Fig. 4 The layout of the spanning tree from determinant encoding—an illegal tree

the encoding value. For example, let $s_g = [42656]$ represent a chromosome coded by determinant encoding. It implies that there are six nodes in the network corresponding to each fixed position [23456] with the links $(2, 4)$, $(3, 2)$, $(4, 6)$, $(5, 5)$ and $(6, 6)$ in the tree. (Each corresponding fixed position in the figure is equal to the order of the gene plus one. Each allele of chromosome can be connected to its corresponding fixed position as in Step 2 in the procedure presented above.) The final layout of the tree is shown in Fig. 4. The generated spanning tree may not be legal and need to be repaired by reallocating genes in appropriate positions to generate a legal tree. From Fig. 4, one may detect that there exists no connecting link to node 1, which is called "*missing node 1*". On the other hand, one may find "*self-loop*" in the tree since the links $(5, 5)$ and $(6, 6)$ leads to such a problem. Also, "*cycle*" occurs if a subset of links connect in a loop, returning to the original node, in which case one of the links may be unnecessary.

In order to ensure feasibility of a substring in determinant encoding, we should restrict the range of the encoding value corresponding to the transportation between the suppliers and the plants, that between the plants and the DCs and that between the DCs and the customers. We take the transportation between the I suppliers and J plants as an example. The first $(I - 1)$ genes for this stage correspond to the indices for the plants that suppliers 2 to I connected with. Obviously, they can connect only to the nodes indexed from $(I + 1)$ to $(I + J)$. Then, the next J genes for the J plants should take the encoding value the range in from 1 to I. The restrictions on the values of encoding also apply to the stage between the plants and the DCs and the stage between DCs and the customers. It will never happen the cases of *self-loops* and *cycles* with the restrictions on the range of the encoding value. Therefore, the restrictions on the range of the encoding value effectively prevent the possibility of producing illegal spanning trees. Only a single issue we must deal with is problem of *missing node 1*. Since this problem often happens when generating the initial population, we present our procedure for fixing illegal spanning trees in Sect. 4.2.

4.2 Initialization, ensuring feasibility and fitness evaluation

When implementing the initialization of the proposed GA, one part of the chromosomes in the initial population are randomly generated, but taking into accounts of the restrictions on the range of the encoding value. We set α as the portion of the chromosomes that are randomly generated. (We take the value of $\alpha = 10\%$ in our numerical experiments.) We denote Π as the population size. Then, the other chromosomes in the initial population are generated by a greedy heuristic. Let q be the index for the fixed position in each chromosome. The pseudo code of the initialization procedure is presented as follows.

The initialization procedure

1. (The randomly-generated part) Calculate the value of $\rho = \lceil \Pi \cdot \alpha \rceil$. For this set of ρ chromosomes, we randomly generate the values of $S_g(q)$ but take into accounts of the restrictions on the range of the encoding value for each chromosome S.
2. (The greedy-generated part) For the other $\Pi - \rho$ chromosomes, we do the following steps. Let $q = 2$.
   (a) For the qth fixed position, pick the link (q, r) with the *lowest* transportation cost per-unit but taking into accounts of the restrictions on the range of the encoding value. When there exist two or more tie links, we randomly pick one of these tie links.
   (b) Set $q = q + 1$. If $q \leq I + J$, then go to Step 2(a); otherwise, stop.

We have further explanation on Step 2(a) as follows. Suppose that a supply chain has I suppliers and J plants. Recall that the first fixed position corresponds to node 2, which is the supplier no. 2. For a substring S_3, we should pick the link $(2, \bar{j})$ with $\bar{j} = \arg\min_j \{s_{2j}\}$. We note that for a fixed position q, usually there exist many candidate links with the same transportation cost per unit since the nodes corresponding to these candidate links are located within a close range of transportation distance. Therefore, the random tie-breaking in Step 2(a) will prevent plenty copies of same chromosomes existing in the initial population.

Since the problem of "missing node 1" still happen, there may exist illegal trees in the initial population. For a given substring S_q with its length being λ, we may use the following procedure for fixing this problem.

The procedure for fixing missing node 1

1. Let $q = 1$ and flag $= 1$ (which defaults that the problem of missing node 1 exists).
2. Check the value of $S_g(q)$: If $S_g(q) = 1$, then set flag $= 0$. Go Step 3.
3. If $q < \lambda$, set $q = q + 1$ and go to Step 2; otherwise, go to Step 4.
4. If flag $= 0$, then stop (without the problem of missing node 1); otherwise, pick plant $\bar{r}$ among those candidate sites with $\bar{r} = \arg\min_r \{s_{1r}\}$ and assign 1 to the corresponding fixed position. (If there is a tie, randomly select a position.)

Although determinant encoding is an indirect encoding strategy, the decoding algorithm is very simple. Also, it enjoys the advantage that there exist no infeasible solution in the proposed chromosome representation. The only disadvantage is that it may generate illegal trees with the problem of missing node 1. But, from another point of view, it provides opportunities to improve the value of the objective function when fixing the illegal trees (that is not possible for Prüfer encoding).

After conducting decoding, we are ready to determine the values of $x_{ij}, y_{jk}, z_{kl}|\forall i, j, k, l$ for fitness evaluation. We note that the decisions on the values of different stages are independent, e.g., the decisions on the values of x_{ij}'s are independent of y_{jk}'s. We may use the following procedure to determine the values of $x_{ij}, y_{jk}, z_{kl}|\forall i, j, k, l$, in a stage-by-stage fashion. (Here, we employ the decision on the values of x_{ij} as an example in the transportation quantity assignment procedure. The values of y_{jk}'s and z_{kl}'s can be determined in a similar way.)

The transportation quantity assignment procedure

1. Mark all the links in the chromosome as "unlabeled".
2. Randomly pick an unlabeled link (i, j) that connects node i and node j in the supply chain, and set $x_{ij} = \min\{a_i, b_j\}$ where a_i and b_j are the available capacity of supplier i and plant j, respectively.
3. Mark the link (i, j) as "labeled" and update the available capacity by $a_i = a_i - x_{ij}$ and $b_j = b_j - x_{ij}$.
4. If there exists any unlabeled link, go to Step 2; otherwise, stop.

After determining the values of x_{ij}, y_{jk}, z_{kl}, $\forall i, j, k, l$, we are able to secure the value of the objective function by plugging them into the term $FC \cdot [\lceil \sum_{x_{ij}\neq 0} x_{ij}/VL \rceil + \lceil \sum_{y_{jk}\neq 0} y_{jk}/VL \rceil + \lceil \sum_{z_{kl}\neq 0} z_{kl}/VL \rceil]$ in (11) or the term $\sum_c \tau_c(x_{ij}, y_{jk}, z_{kl}|\forall i, j, k, l)$ in (12). The selection mechanism presented in the next subsection will be conducted based on the objective function obtained here.

4.3 Selection, genetic operators and termination

Some researchers prefer to use the enlarged sampling approach since it reduces the possibility of duplicate chromosomes entering the population during selection (Gen and Cheng 1997). Typically, there are two enlarged sampling strategies: $(\mu + \lambda)$ and (μ, λ). In $(\mu + \lambda)$ strategy, μ parents and λ offsprings compete for survival and the μ best solutions are selected for the next generation. In (μ, λ) strategy, we select the μ best solutions from out of λ offspring solutions. We used the stochastic $(\mu + \lambda)$ method as recommended in Chou et al. (2001).

On the genetic operators, we compare several combinations of crossover and mutation operators. In the GA braught by Syarif et al. (2002), they employ the *single-point* operator for crossover and the *inversion-displacement* operator for mutation. In *one-point crossover*, a random position is generated for a pair of chromosomes and the alleles of the first chromosome from this fixed position to the end are exchanged with the second chromosome in the same range. In this process, the alleles of the second chromosome are transferred to the respective alleles in the first chromosome.

In order to look for a more efficient implementation for the proposed GA, we include *two-point* and *uniform* operators for crossover and both *insertion* and *exchange* operators for mutation. *Two-point crossover* generates two random positions, head and tail. The alleles of the first chromosome from the head position to the tail are exchanged with the second chromosome in the same range. *Uniform crossover* is a dynamic and nondeterministic method where a set of positions, called a mask, is chosen for each of the chromosomes and their alleles are exchanged with each other based on the generated positions. There are two random decisions-the positions to replace and the number of genes to replace. *Insert mutation* randomly generates two positions in a given chromosome and inserts the gene from the first position in the second position and shifts all the genes to the right by one position. *Exchange mutation* randomly selects two positions in a given chromosome and exchanges both genes. Insert mutation causes greater changes to the chromosome compared to exchange mutation. This is particularly true in our study since the genes in a chromosome are related to their fixed positions.

Obviously, the genetic operators should be applied to the parts of the chromosome for different stages of the supply chain independently. Also, when implementing the genetic operators in the proposed GA, one should pay attention to the following two issues to ensure feasibility of the chromosomes.

1. We should follow the restrictions on the range of the encoding value discussed in Sect. 4.1 to ensure the feasibility of the chromosome. For example, supposed that we apply a crossover operator to the substrings corresponding to the transportation between the I suppliers and J plants for two chromosomes. In such a case, we first conduct the crossover operator to the first $(I-1)$ genes of the substrings so as to keep the values of the indices for the plants in their feasible range. Then, we do the crossover operator to the other J genes of the substrings for the J plants. (Mutation operators shall be applied in a similar way.)
2. We should fix those chromosomes experiencing the operations of genetic operators if they suffer the problem of missing node 1.

An important issue in the implementation of GA is the *termination criterion*. Several *termination criteria* are established from number of generations, computing time, and fitness convergence. Fitness convergence occurs when all the chromosomes in the population have the same fitness value. In this study, fitness convergence is selected as the termination criterion. Namely, we stop the evolutionary process in GA when the best chromosome on hand was not improved in the last 10 generations.

The population size, crossover rate, and mutation rate are three other important control parameters for GA. For the sake of having a fair stand for comparison, we employed the same settings as Syarif et al. (2002) did for the implementation of their GA.

5 Numerical experiments

In the first part of this section, we will demonstrate that the proposed GA using determinant encoding and our cost-reduction heuristic, which is abbreviated as Y&H, is superior to the GA using Prüfer encoding presented in Syarif et al. (2002), which is abbreviated as SYG for the rest of the paper. Also, we will bring up with the recommend genetic operators following the results of our experimental design on the

Table 1 The configurations of the four problems in our numerical experiments

Prob.	No. of suppliers	No. of manufacturers	No. of DCs	No. of customers
1	4	5	5	5
2	8	10	10	10
3	15	20	20	40
4	20	25	25	60

selection of genetic operators in the second part of this section. The third part of this section conducts further comparison between the Y&H and SYG using larger number of instances.

5.1 Comparison of two solution approaches

In order to compare the performance of the proposed GA and the GA using Prüfer encoding, we randomly generate four instances of the problem (P_1) with different configurations (suppliers, manufacturers, distribution centers and customers) of the supply chain networks. Table 1 summarizes the configurations of the supply chain networks in our experiments.

We note that the parameters of the instances used for comparison in this section are randomly generated. The capacity limit of each node in the supply chain, the fixed cost of the suppliers and DCs, and the transportation cost in each stage are generated from Uniform $[200, 1500]$, Uniform $[400, 2500]$ and Uniform $[1, 10]$, respectively. In our experiments, we have the fixed cost for subcontracting a transportation vehicle given by $FC = 500$ and the capacity limit of a transportation vehicle given by $VL = 300$. Recall that the total amount of shipment between two neighboring echelons is denoted as TA (which shall be calculated as the values of x_{ij}, y_{jk}, z_{kl} are given from the transportation quantity assignment procedure presented in Sect. 4.2). Therefore, the cost for subcontracting the transportation vehicles operating between the two neighboring echelons is given by $500 \cdot \lceil TA/300 \rceil$.

Our numerical experiments were done by a personal computer with a CPU of AMD Sempron(tm) 2400+, 1.67 GHz and 1 G RAM. In order to rule-out the effects from using different genetic operators, we use the same combination as the SYG's (i.e., *one-point* operator for crossover and *inversion-displacement* operator for mutation) in our experiments here. (Though after comparing different combinations of genetic operators, we find that two-point operator for crossover and exchange operator for mutation will be the most effective one for the proposed GA. We will present our discussions on the design of experiments in Sect. 5.2. Also, we will compare the Y&H with the SYG using our best combination of genetic operators in Sect. 5.3 later.) The crossover rate and the mutation rate are 0.4 and 0.2, respectively. The population size is 50. We run the Y&H and SYG for 30 times for each problem. Table 2 summarizes the objective function value and the run time for the comparison of the two solution approaches.

Table 2 "seems" indicating that the proposed GA (i.e., Y& H) is superior to SYG. In order to earn objective support from statistical analysis, we employed t-test to verify our intuition and formulated the following hypothesis:

Table 2 The comparison of the two solution approaches

Prob.	The value of the objective function					Run time (in sec.)	
	SYG		Y&H		Improvement of Y&H	SYG	Y&H
	Average	Best	Average	Best			
1	\$54,425	\$53,360	\$53,847	\$53,080	1.06%	10.73	9.85
2	106,104	104,750	105,582	104,660	0.49%	34.10	24.87
3	272,187	265,020	263,577	260,300	3.16%	59.65	38.60
4	406,631	397,870	389,657	381,800	4.17%	170.30	90.18

H_0: There exists no significant difference between the solution from Y&H and SYG.
H_1: Y&H is superior to SYG.

The results of our t-tests indicate that one may reject H_0 with a 95% confidence level (i.e., $\alpha = 0.05$). Also, since the p-values of all are less than 0.0026, we are strongly confident that the same conclusion still holds for those cases with smaller values of α. Therefore, we can draw our conclusion that Y&H is superior to SYG according to our numerical experiments.

Interestingly, one may observe that the larger the problem size, the better the performance of Y&H (from the determinant encoding). On the other hand, the solutions from Y&H outperform the solution from SYG in both the average and the best of the objective function value. Therefore, we may learn that the determinant encoding is superior to Prüfer encoding in our experiments. Also, one may notice that for the large-size problems (i.e., problem no. 3 and no. 4), the best solutions from SYG (using Prüfer encoding) is even worse than the average of the solutions of Y&H. Such an advantage may result from the fixing heuristic in Sect. 4.2, that plays the role of local search in the evolutionary process of GA, enhancing the ability of exploitation of the proposed GA.

On the other hand, one may observe that the run time of SYG becomes significantly longer than Y&H for the larger-size problems. The run time of SYG is almost twice of Y&H for problem no. 4. Therefore, we would comment that the proposed GA using determinant encoding performs better in the convergence rate of GA.

5.2 Experimental design on the selection of genetic operators

In this section, we would like to find the best combination of genetic operators for the proposed GA. Here, we compare three operators for crossover, namely, *one-point* (C1), *two-point* (C2) and *uniform* (C3) operators, and three operators for mutation, viz., *inversion-displacement* (M1), *insertion* (M2) and *exchange* (M3) operators. (Note that we define the alias of each genetic operator to facilitate our presentation later.) Similar to our experiments in Sect. 5.1, we set the crossover rate and the mutation rate to be 0.4 and 0.2, respectively, and the population size to be 50. We solved the problem (P_1) for 10 times, and summarized our results in Table 3. In each cell of Table 3, we show the average objective function value and the average run time of the proposed GA. The best solution is indicated in the parenthesis. For example, when using C1 and M1 as the genetic-operator combination, the average and the best

Table 3 The comparison of the two solution approaches

Cross.	Mut.	Prob. no.							
		1		2		3		4	
		Obj. value	Time (sec.)	Obj. value	Time (sec.)	Obj. value	Time (sec.)	Obj. value	Time (sec.)
C1	M1	$53,847	9.85	$105,582	24.87	$263,577	38.60	$389,657	90.18
		(53,080)		(104,660)		(260,300)		(381,800)	
	M2	54,017	9.64	106,117	24.67	267,725	48.55	397,910	71.65
		(53,150)		(105,200)		(263,660)		(388,380)	
	M3	54,064	*9.59*	106,274	14.16	263,137	*23.80*	389,683	*52.01*
		(53,370)		(105,110)		*(256,980)*		(382,390)	
C2	M1	54,171	13.2	106,288	19.80	264,686	42.04	389,063	82.30
		(53,550)		(105,460)		(259,930)		(380,340)	
	M2	54,114	13.45	106,411	19.01	268,097	34.07	397,580	71.98
		(53,430)		(104,860)		(261,940)		(387,870)	
	M3	53,738	11.52	*105,482*	*13.91*	*261,765*	28.87	*386,015*	68.29
		(53,080)		*(104,460)*		(257,630)		*(379,540)*	
C3	M1	*53,572*	22.06	106,209	28.01	263,606	72.95	390,572	132.95
		(53,080)		(105,840)		(259,020)		(385,640)	
	M2	53,925	15.76	107,155	18.48	267,507	58.92	397,917	95.92
		(53,310)		(106,260)		(262,600)		(390,580)	
	M3	53,886	9.65	105,644	22.01	262,051	48.04	388,502	81.41
		(53,150)		(104,550)		(257,340)		(380,950)	

objective function value obtained for the problem no. 1 is \$53,847 and \$53,080, respectively. Also, for each problem, the best objective function value or the shortest run time are shown in *italic fonts*.

From Table 3, one may observe that C2 + M3 (i.e., the combination of the *two-point* operator for crossover and the *exchange* operator for mutation) brought the best solution for most of the cases. Furthermore, it shows that the proposed GA converges in a shorter run time comparing with others. Especially, the larger the size of the problem, the better the recommended combination. We note that such an advantage is very important for the applications in the real world since the size of the problem is usually very large for the practical cases. Following our numerical experiments, we suggest using the *two-point* operator for crossover and the *exchange* operator for mutation when employing the proposed GA for solving the problem of designing a supply chain network.

5.3 Further comparisons

After comparing different combinations of genetic operators in Sect. 5.2, we find that two-point operator for crossover and exchange operator for mutation will be the most effective one for the proposed GA. We would like to use larger number of sample

Table 4 The improvement of the Y&H comparing with the SYG

Instance	Data set 1		Data set 2		Data set 3		Data set 4	
	Obj. value	Run time (sec.)	Obj. value	Run time (sec.)	Obj. value	Run time (sec.)	Obj. value	Run time (sec.)
1	0.06%	66.08%	7.29%	67.41%	17.34%	65.09%	23.82%	84.64%
2	4.24%	65.10%	8.41%	74.80%	15.77%	84.91%	21.56%	84.45%
3	2.74%	60.26%	5.70%	73.17%	16.17%	75.46%	16.23%	75.98%
4	0.11%	60.81%	7.08%	74.50%	14.30%	81.84%	16.90%	79.62%
5	2.60%	75.66%	7.18%	72.14%	15.08%	85.17%	16.68%	80.76%

problems with to demonstrate the superiority of the proposed approach by using our best combination of genetic operators here. There are four sets of experiments, and we randomly generate 5 instances for each data set. The instances in the first data set are generated using a similar way as we did for the problem no. 1 in Table 2. (The second set is generated similar to the problem no. 2, and so on.) The number of nodes in each stage of supply chain (which shall be an integer) are randomly generated with a discrete uniform distribution. For the first set of instances, the number of nodes in each stage is generated from Uniform [3, 8]. The uniform distributions for the second, the third, and the fourth sets are Uniform [8,14], Uniform [15, 40], and Uniform [20, 60], respectively. We collect the data of the objective function values and the run time for both the Y&H and SYG approaches for 30 runs for each instance. By comparing with the results of the SYG, we summarize the improvement of the Y&H for each data set in Table 4.

From Table 4, the improvement of the proposed approach (i.e., the Y&H) is impressively significant by using our best combination of genetic operators (with two-point crossover and exchange mutation). Also, the larger the size of the problem, the larger the improvement of the proposed approach. Therefore, we conclude that the proposed approach outperforms the approach by Syarif et al. (2002) in both aspects of the solution quality and the run time.

6 Conclusion

This study focuses on the design of configuration and the transportation planning in multi-stage supply chain networks. The decision makers need to determine the optimal locations of the hubs and the optimal transportation routes to minimize the total costs incurred in the whole system. We formulate two mathematical models by revising the model in Syarif et al. (2002) by taking into accounts more practical cases in the real world (e.g., considering the cost for subcontracting the transportation vehicles between any two neighboring echelons) in the supply chain network. We note that this problem is formulated as a 0-1 mixed integer non-linear program that is not able to be efficiently solved by commercial packages due to its complexity. This study proposes a new spanning tree-based Genetic Algorithm (GA) using determinant encoding for solving this problem. And, we derive an efficient heuristic that fixes illegal spanning trees existing in the chromosomes obtained from the evolutionary process. It also plays the role of a local search to improve the candidate solutions obtained in the evolutionary process of the proposed GA. Our numerical experiments demonstrate that the proposed GA outperforms the GA of Syarif et al. (2002) in the solution quality and convergence rate.

Recently, the authors are trying to apply the proposed methodology in this study to other network design problems in the literature.

Appendix: The number of possible links in one stage of the supply chain

In this section, we would like to explain the reasons why there should be $(I + J - 1)$ links between I suppliers and J plants in the supply chain.

Intuitively, there are possible links between I suppliers and J plants. Here, following the rationale of enumeration, we would like to show that there could be no more than $(I + J - 1)$ links in an optimal solution for the supply chain network design problem. For example, if there are 3 suppliers and 3 plants, the total number of possible links should be 9. We enumerate all the possible cases when optimality is reached as follows.

1. The supplier 1 is assigned to three plants. It is necessary that at least two of the plants consumed all of their capacity; otherwise, it does not reach optimality. That is, at most only one plant still has capacity available. After assigning the supplier 1 to these three plants, we have two suppliers and one plant left. No matter both suppliers or one of the suppliers were assigned to the plant (with capacity available), the total numbers of links will not exceed 5, which is $(I + J - 1)$.
2. The supplier 1 is assigned to two plants. It is necessary that at least one of the plants consumed all of its capacity; otherwise, it does not reach optimality. In such a case, we have a problem with two suppliers and two plants left. If the supplier 2 is assigned to two plants. It is necessary that at least one of the plants consumed all of its capacity; otherwise, it does not reach optimality. Then, we have only one supplier and one plant left. No matter when happen to the last case, the total numbers of links will not exceed 5.
3. The supplier 1 is assigned to only one plant.

The supplier 2 is assigned to three plants. It is necessary that at least two of the plants consumed all of their capacity; otherwise, it does not reach optimality. That is, at most only one plant still has capacity available. After assigning the supplier 2 to these three plants, we have one supplier and one plant left. Then, we have only one supplier and one plant left. The total numbers of links will not exceed 5.

The supplier 2 is assigned to two plants. It is necessary that at least one of the plants consumed all of its capacity; otherwise, it does not reach optimality. Then there are two plants and one supplier left where two links will be assigned at most. Therefore, the total numbers of links will not exceed 5.

The supplier 2 is assigned to only one plant. There could be three plant still available in such a case. The only supplier 3 may be assigned to all of these three plants at most. Therefore, the total numbers of links will not exceed 5.

We may easily generalize our rationale to all of the cases with I suppliers and J plants in the supply chain. We conclude that there are no more than $(I + J - 1)$ links between I suppliers and J plants when optimality is reached.

References

Abuali FN, Wainwright RL, Schoenefeld DA (1995) Determinant factorization: A new encoding scheme for spanning trees applied to the probabilistic minimum spanning tree problem. In: Eshelman LJ (ed) Proceedings of the sixth international conference on genetic algorithms. Morgan Kaufmann, San Mateo, pp 155–192

Azevedo AL, Sousa JP (2000) Order planning for networked make-to-order enterprises: A case study. J Oper Res Soc 51:1116–1127

Bramel J, Simchi-Levi D (1997) The logic of logistics: Theory, algorithms and applications for logistics management. Springer, New York

Chou H, Premkumar G, Chu CH (2001) Genetic algorithm for communications network design—An empirical study for the factors that influence performance. IEEE Trans Evolut Comput 5(3):236–249

Delbem ACB, de Leon Ferreira de Carvalho A, Bretas NG (2005) Main chain representation for evolutionary algorithms applied to distribution system reconfiguration. IEEE Trans Power Syst 20:425–436

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

Gen M, Kumar A, Kim JR (2005) Recent network design techniques using evolutionary algorithms. Int J Prod Econ 98(2):251–261

Geoffrion A, van Roy TJ (1979) Caution: Common sense planning methods can be hazardous to your corporate health. Sloan Manag Rev 20:30–42

Jo JB, Li YZ, Gen M (2007) Nonlinear fixed charge transportation problem by spanning tree-based genetic algorithm. Comput Ind Eng 53:290–298

Marry JM, Vidyaranya BG (2005) Global supply chain design: A literature review and critique. Transp Res Part E 41:531–550

Narula SC, Ho CA (1980) Degree-constrained minimum spanning tree. Comput Oper Res 7(4):239–249

Pirkul H, Jayaraman V (1998) A multi-commodity, multi-plant, capacitated location allocation problem: Formulation and efficient heuristic solution. Comput Oper Res 25:869–878

Ro H, Tcha D (1984) A branch and bound algorithm for two level uncapacitated facility location problem with some side constraint. Eur J Oper Res 18:349–358

Russell RM, Krajewski LJ (1991) Optimal purchase and transportation cost lot sizing for a single item. Dec Sci 22:940–954

Siajadi HR, Ibrahim N, Lochert PB, Chan WM (2005) Joint replenishment policy in inventory-production systems. Prod Plan Control 16:255–262

Sim E, Jang Y, Park J (2000) Study on the supply chain network design considering multi-level, multi-product, capacitated facility. In: Proceedings of Korean supply chain management society

Simchi-Levi D, Kaminsky P, Simchi-Levi E (2003) Designing and managing the supply chain: Concepts, strategies, and case studies, 2nd edn. McGraw-Hill, New York

Syarif A, Yun YS, Gen M (2002) Study on multi-stage logistic chain network A spanning tree-based genetic algorithm approach. Comput Ind Eng 43:299–314

Tilanus B (1997) Introduction to information system in logistics and transportation. Elsevier, London

Tragantalerngsak S, Holt J, Ronnqvist M (1997) Lagrangian heuristics for the two-echelon, single-source, capacitate location problem. Eur J Oper Res 102:611–625

Yeh WC (2005) A hybrid heuristic algorithm for the multistage supply chain network problem. Int J Adv Manuf Technol 26(5–6):675–685

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

Zhou G, Gen M (2003) A genetic algorithm approach on tree-like telecommunication network design problem. J Oper Res Soc 54(3):248–254