Computers & Operations Research 37 (2010) 376 – 389

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

A closed-loop logistic model with a spanning-tree based genetic algorithm

Hsiao-Fan Wang*, Hsin-Wei Hsu

Department of Industrial Engineering and Engineering Management, National Tsing Hua University, R721, Engineering Building 1, 101, Section 2, Kuang-Fu Road, Hsinchu 300, Taiwan, ROC

ARTICLE INFO

Available online 11 June 2009

Keywords:
Integer linear programming
Genetic algorithm
Closed-loop supply chain
Logistics and location decisions
Spanning-tree

ABSTRACT

Due to the problem of global warming, the green supply chain management, in particular, closed-loop logistics, has drawn the attention of researchers. Although there were logistics models that were examined in the literatures, most of them were case based and not in a closed-loop. Therefore, they lacked generality and could not serve the purposes of recycling, reuse and recovery required in a green supply chain. In this study, the integration of forward and reverse logistics was investigated, and a generalized closed-loop model for the logistics planning was proposed by formulating a cyclic logistics network problem into an integer linear programming model. Moreover, the decisions for selecting the places of manufactories, distribution centers, and dismantlers with the respective operation units were supported with the minimum cost. A revised spanning-tree based genetic algorithm was also developed by using determinant encoding representation for solving this NP model. Numerical experiments were presented, and the results showed that the proposed model and algorithms were able to support the logistic decisions in a closed-loop supply chain efficiently and accurately.

Statement of scope and purposes This study concerns with operations of 3*R* in the green supply chain logistics and the location selection optimization. Based on 'cradle to cradle' principle of a green product, a "closed-loop" structure of a network was proposed in order to integrate the environmental issues into a traditional logistic system. Due to NP-hard nature of the model, a Genetic Algorithm, which is based on spanning tree structure was developed. Test problems from the small size for accuracy to the large scale for efficiency have been demonstrated with comparison. The promising results have shown the applicability of the proposed model with the solution procedure.

1. Introduction

Due to the awareness of the environmental protection, how to reduce the utilization of the materials by reusing and remanufacturing the used products has been a critical issue for an enterprise. This induces the concept of the green supply chain management and has led to a problem of the closed-loop supply chain management. Different from a conventional supply chain, planning a green supply chain requires an additional function of recycling and thus, a closed-loop chain is a necessary infrastructure for a material flow. Then, with well-managed reverse logistics, 3*R* of *r*educe, *r*ecovery and *r*euse for the environmental protection can be achieved with cost savings in the procurement, the disposal and the transportation [19].

Technically, the closed-loop logistics comprises two parts: forward logistics and reverse logistics. For the forward logistics, as a conventional logistics, after manufactory, the distributors will deliver the final products to the customers to satisfy their demands and the position of the customers is typically the end of the process. For the reverse logistics, the flow of used products is processed from the customers back to the dismantlers to do the sorting or disassembling for recovery, reuse or disposal [3,15,23,26,29]. The closed-loop logistics management is to ensure the least waste of the materials by following the Conservation Law along the life cycles of the materials.

Generally speaking, for a network planning problem, there are three issues needed to be considered: the validity of the model, the efficiency of the solution and the applicability. For a closed-loop logistics problem, the status quo of these issues can be summarized as follows:

(1) For modeling: Most of the studies only discussed the reverse logistics model. Some studies have proposed the closed-loop models, but the loops were considered as a "prolonged supply chain" by lacking of the relation between forward and reverse flows [9,28,31]. Therefore, instead of sharing the same capacity, the models often assumed the unlimited capacities for the reverse logistics or did not state the relation between forward

* Corresponding author. Tel.: +886 3 5742654; fax: +886 3 5722204.
E-mail address: hfwang@ie.nthu.edu.tw (H.-F. Wang).

0305-0548/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.06.001

and reverse flows, which are not valid for representing the real situations.

(2) For applications: Because of the assumed unlimited capacities for the facilities of distribution centers or dismantlers in the reverse logistics, it has resulted in unrealistic route design for the used materials from the customers for recovery, reused or disposal [25].

(3) For solutions: A logistics planning problem with the location selection is an NP-hard problem [8,14,18,30]. Efficient solution procedure remains a challenge for researchers and practitioners.

To cope with these issues, in this study, we shall investigate the features and purposes of a green logistics management and in particular, the difference of a "prolonged supply chain" and a closed-loop supply chain for green products. Based on the findings, we shall develop a comprehensive green logistics model to support decisions of facility locations and material flows through a closed-loop capacitated supply chain when the realistic applications of 3*R* green materials logistics are expected with minimum total cost. Furthermore, an efficient algorithm will be developed and evaluated.

After the literature review of closed-loop logistics and Genetic Algorithms in Section 2, a mathematical model for closed-loop logistics will be proposed with the specification of its properties in Section 3. Estimation of its complexity will be done with numerical illustrations. In Sections 4 and 5, a revised spanning-tree based genetic algorithm is proposed to resolve this model, which is evaluated using large-scale problems. Finally, in Section 6, the conclusion will be drawn.

2. Literatures review

In this section, we probe the literature and categorize studies into two. The first one is the closed-loop logistics, and the second is the Genetic Algorithms used in a logistics network.

2.1. Closed-loop logistics

Closed-loop logistics refers to all those activities associated with the transformation and the flows of goods and services with their information from the sources of the materials to the end users. Management refers to the integration and treatment of these activities, both internal and external of a firm [4]. Therefore, an integrated supply chain management aims to close material cycles and prevent the leakage of the materials from the chain using the minimal costs to achieve maximal value [6].

Fleischmann et al. [10] pointed out that a closed-loop supply chain may include traditional manufacturers, retailers, with logistics service providers in the forward channel; and specialized parties of secondary material dealers and material recovery facilities in the backward channel. One of the key aspects in the closed-loop management is the simultaneous improvement of both economic and environmental performance throughout the chain by establishing long-term relationships between buyers and suppliers [35]. Therefore, building a stable closed-loop logistics in a chain is necessary when the objectives of minimizing both the total cost and the involved environmental impacts are desired by the companies.

Therefore, for a closed-loop logistics problem, in addition to the conventional logistics which is described by a network of suppliers, manufacturing sites, distribution centers (DCs) and customer locations through logistics, an important module, dismantlers or recyclers, is incorporated into a supply chain network. These dismantlers handle the recovered resources into many different types for further use or disposal [3]. That is, if the recycled resources can be used again at dismantler sites, the resource should be shipped to a manufacture for reproduction; otherwise, the un-useful resources must be land filled [29]. This behavior in a closed-loop system is regarded as *reverse logistics*.

In reality, a distribution center often plays such role as a collector in a recovery system. Therefore, in this study, depending on the needs, a DC in a closed-loop logistics network can act as a distributor only or both distributor and collector.

The framework of closed-loop logistics can be viewed as Fig. 1.

The integration of the forward and reverse logistics model is called the *closed-loop logistics model (CLL)*. Few studies have considered this issue. Fleischmann et al. [9] designed a reversed logistics network by considering the forward flow together with the reverse flow which has no capacity limit. Extending Fleischmann et al.'s model, Salema et al. [28] proposed a general model that has been applied to an Iberian company. However, when suspending the logistics between dismantlers and plants, both Fleischmann et al. and Salema et al.'s models did not consider the supplier side and lacked the relations between forward and reverse flows. Sometimes, the DCs also play the role of the collected centers. Thus, the capacity of DC is used for both distribution and collection. When the amounts of the collection are larger, then the amounts of the distribution must decrease under the same capacity. Similarly, if we consider the supply side, the plants must allow the materials flow from both forward (suppliers) and reverse (dismantlers) under the same capacity. If the amounts of returns are larger in a certain plant, the amounts of orders from suppliers will decrease. These interactions are the character of a closed-loop supply chain and the model cannot be separated into two parts independently. Without considering such kind of relations, the model is simply a "prolonged supply chain" including forward and reverse chains but not a loop.

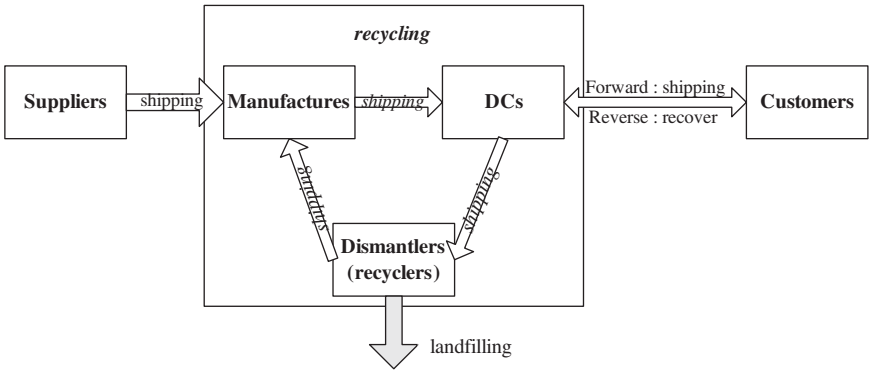

Fig. 1. Framework of green supply chain logistics.

In addition, Salema et al.'s model left the unsatisfied demand and did not consider the relations between demands, recovery amounts, and landfilling amounts, which is undesirable and unrealistic in management.

Üster et al. [31] considered a closed-loop network design problem, and solved by Benders decomposition. Üster et al. separated manufacturing from remanufacturing and assumed a single source for the customer supplying. Treating the model as the problem of the simple assignment has prevented it from general applications. The realistic logistics solution should provide the best combination of routings, and each node should be able to connect all nodes in the next stage, but not exclusively assigned.

The problem with the location selection of conventional logistics planning has been discussed extensively, and this kind of problem is an NP-hard problem [11,30]. Therefore, to overcome the shortages of the existing models and tackle the features of closed-loop logistics, how to develop an efficient solution procedure is another aim of our study.

2.2. Genetic algorithms in logistic networks

Genetic Algorithm is a commonly used optimizing tool for engineering calculation. It was proposed by Professor John Holland of the University of Michigan in 1975. Dengiz et al. [7] offered many examples on GA, which showed that it can be applied to a wide variety of applicative domains. For the fundaments of GA, one can refer to Gen and Cheng [11]. In the reverse logistics, Min et al. [24] also successfully used GA to develop a multi-echelon reverse logistics network for product returns.

The concept of applying a spanning tree to supply chain network problems was first proposed by Syarif et al. [30]. To utilize the characteristic of a spanning tree to set up the code of a genetic algorithm, Syarif et al.'s method was able to determine the locations of manufacturing sites and DCs. This kind of problem is a kind of fixed charge transportation problem (FCTP), and both Jo et al. [18] and Gottlieb and Paulmann [14] successfully adopted spanning tree-based GA with Prüfer encoding to solve FCTP.

In the literature, various encoding methods have been used to represent trees, and they can be classified broadly into three categories: edge, node, and edge-node encodings. Edge encoding has been found to be a poor representation, while in *node-* or *vertex*-based encoding, the nodes rather than the edges are represented in the encoding [5]. A popular encoding method for trees called Prüfer encoding is based on the Prüfer number [16], which represents a tree of n nodes with $n-2$ digits, where each digit is an integer between 1 and n. However, empirical investigations have shown that Prüfer encoding is a poor method in evolutionary algorithms and thus should be avoided [13,20,34].

Abuali et al. [1] based on nodes to offer Determinant Encoding, and they proved that the code is better than Prüfer's. Both the works of Chou et al. [5] and Yao and Hsu [33] for a logistic network further confirmed this conclusion.

There are two issues that are needed to be addressed for the initialization of a GA: the population size and the procedure to initialize the population. For the population size, Goldberg [12] has shown the need of increasing population exponentially with the length of the chromosome string in order to generate good solutions. Regarding the generation of the initial population, there were two ways proposed in the literature: random initialization and heuristic type initialization. Doris et al. [27] used alternative approach to generating initial solutions in place of random method and got a better result. Baker and Ayechew [2] compared random, heuristic type, and mixed population containing both random and heuristic initial solutions, and concluded that an initial population of heuristic solutions will lead to high-quality solutions in a relatively small number of generations of the GA. However, a possible drawback is that such a population will lack of diversity needed to obtain near-optimal solutions.

2.3. Conclusion

Reviewing the above-mentioned literature on closed-loop logistics, it is noted that most of the existing models lack of the relations between forward and reverse logistics embedded in a closed-loop system. Since 3*R* of *recovery*, *recycle* and *reuse* are activities to fulfill the basic principle of "cradle to cradle" in closed-loop management, therefore, Conservation Law should be obeyed at each node or state and will be satisfied in our proposed closed-loop logistics model.

In addition, the closed-loop logistics is an NP-hard problem. Thus an efficient algorithm is essential for enterprises. From the literatures, we can note that GAs have been successfully applied to a wide range of domains including reverse logistics. Among varieties of versions, spanning-tree based GAs with Determinant Encoding technique has shown their potential in solving forward logistics with good outcomes. Therefore, we shall adopt such technique to develop a GA to solve the proposed closed-loop logistic model.

3. A mathematical programming model for closed-loop supply chain logistics

From the concepts we described above, we know that the closed-loop supply chain is different from a conventional supply chain. The problems involved are more complex, and need more than double efforts to analyze both forward and reverse logistics simultaneously.

To measure the effectiveness of the logistics in a closed-loop network, the cost is normally considered by a company. Besides, in a multistage supply chain network problem, the following conditions should be satisfied in modeling [17,30,33,34].

(a) The demand of each customer must be satisfied.
(b) The flow is only allowed to be transferred between two consecutive stages.
(c) The number of facilities that can be opened and their capacities are both limited.

Because they are also the basic conditions for closed-loop logistics, we shall consider them as our assumptions in modeling.

Note that there are essentially five stages along a green logistic network: suppliers, manufacturers, DCs, customers, and dismantlers. Apart from the common conditions of the satisfied demand in (a), and limited capacities in (c); from Assumption (b), it can be noted that there are no flows between the facilities at the same stage.

One special issue of closed-loop logistics is the recycling rate, including the recovery and landfilling rates. Laan et al. [21] pointed out that in the recovery systems; a common assumption is that the amounts of the returned products depend on the demand of the products. To adopt this assumption, the recovery amount is assumed to be a percentage of the customer demand in our model. Then, this leads to the fourth assumption as

(d) The recovery and landfilling rates are given.

The goal of this study is to design a closed-loop supply chain logistics system that can minimize the total transportation and the operation costs by determining locations of the facilities and the flows of the operation units along each capacity-constrained stage when the demand of customers and the recycling rates are given. This closed-loop system is meant to support long-term steady-state

logistics decisions. Therefore, from the economic point of view, we can suggest the minimal cost flows and opening facilities in the system.

3.1. The proposed closed-loop logistics model

Consider the integer-valued basic logistics units in our system, in this section, based on four assumptions and the network structure; we shall propose a mathematical model to describe such logistic system.

Before modeling, we define the related parameters and notations as below:

Indices

I	the number of suppliers with $i = 1, 2, \ldots, I$
J	the number of manufactories with $j = 1, 2, \ldots, J$
K	the number of DCs with $k = 1, 2, \ldots, K$
L	the number of customers with $l = 1, 2, \ldots, L$
M	the number of dismantlers with $m = 1, 2, \ldots, M$

Parameters

a_i	capacity of supplier i
b_j	capacity of manufactory j
Sc_k	total capacity of forward and reverse logistics in the DC k
pd_k	the percentage of total capacity for reverse logistics in DC k
pc_l	recovery percentage of customer l
pl_m	the landfilling rate of dismantler m
d_l	demand of the customer l
e_m	capacity of dismantler m
s_{ij}	unit cost of production in manufactory j using materials from supplier i
t_{jk}	unit cost of transportation from each manufactory j to each DC k
u_{kl}	unit cost of transportation from DC k to customer l
v_{km}	Unit cost of transportation from DC k to dismantler m
w_{mj}	unit cost of transportation from dismantler m to manufactory j
Ru_{lk}	unit cost of recovery in DC k from customer l
f_j	fixed cost for operating manufactory j
g_k	fixed cost for operating DC k
h_m	fixed cost for operating dismantler m
φ	fixed cost for landfilling per unit

Variables

x_{ij}	quantity produced at manufactory j using raw materials from supply i
y_{jk}	amount shipped from manufactory j to DC k
z_{kl}	amount shipped from DC k to customer l
o_{km}	amount shipped from DC k to dismantler m
Rd_{mj}	amount shipped from dismantler m to manufactory j
Rz_{lk}	quantity recovered at DC k from customer l

$$\alpha_j = \begin{cases} 1 & \text{if production takes place at manufactory } j \\ 0 & \text{otherwise} \end{cases}$$

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

$$\delta_m = \begin{cases} 1 & \text{if dismantler } m \text{ is opened} \\ 0 & \text{otherwise} \end{cases}$$

Because the recovery and landfilling rates are the estimated proportional values of the demand and the recovery amount, they are non-integral. Similarly, the percentage of the capacity for the reverse logistics in DC is also estimated proportionally with real numbers. Therefore, in order to maintain integral properties, Gauss symbol is used in our mathematical model and the model is shown below with TC as the total cost:

Object function:

$$\min \quad TC = \sum_i \sum_j s_{ij}x_{ij} + \sum_j \sum_k t_{jk}y_{jk} + \sum_k \sum_l u_{kl}z_{kl} + \sum_k \sum_m v_{km}o_{km} + \sum_m \sum_j w_{mj}Rd_{mj} + \sum_l \sum_k Ru_{lk}Rz_{lk} + \sum_j f_j\alpha_j + \sum_k g_k\beta_k + \sum_m h_m\delta_m + \varphi \sum_m \left\lfloor pl_m \sum_k o_{km} \right\rfloor \tag{1}$$

Subject to

$$\sum_j x_{ij} \le a_i, \quad \forall i \tag{2}$$

$$\sum_k y_{jk} \le b_j\alpha_j, \quad \forall j \tag{3}$$

$$\sum_i x_{ij} + \sum_m Rd_{mj} = \sum_k y_{jk}, \quad \forall j \tag{4}$$

$$\sum_l z_{kl} + \sum_m o_{km} \le Sc_k\beta_k, \quad \forall k \tag{5}$$

$$\sum_j y_{jk} = \sum_l z_{kl}, \quad \forall k \tag{6}$$

$$\sum_m o_{km} \le \lfloor pd_k Sc_k\beta_k \rfloor, \quad \forall k \quad \lfloor\rfloor : \text{floor for Gauss's symbol} \tag{7}$$

$$\sum_l Rz_{lk} = \sum_m o_{km}, \quad \forall k \tag{8}$$

$$\sum_k Rz_{lk} \ge \left\lceil pc_l \sum_k z_{kl} \right\rceil, \quad \forall l \quad \lceil\rceil : \text{ceiling for Gauss's symbol} \tag{9}$$

$$\sum_k z_{kl} \ge d_l, \quad \forall l \tag{10}$$

$$\sum_j Rd_{mj} + \left\lfloor pl_m \sum_k o_{km} \right\rfloor \le e_m\delta_m, \quad \forall m \quad \lfloor\rfloor : \text{floor for Gauss's symbol} \tag{11}$$

$$\sum_k o_{km} = \sum_j Rd_{mj} + \left\lfloor pl_m \sum_k o_{km} \right\rfloor, \quad \forall m \quad \lfloor\rfloor : \text{floor for Gauss's symbol} \tag{12}$$

$$\alpha_j, \beta_k, \delta_m \in \{0, 1\}, \quad \forall j, k, m \tag{13}$$

$$x_{ij}, y_{jk}, z_{kl}, o_{km}, Rd_{mj}, Rz_{lk} \in N \cup \{0\} \quad \forall i, j, k, l, m \tag{14}$$

The objective is to minimize the total cost of the transportation and the operations, and the objective function (1) represents this goal. The constraints mainly contain two types: one is for limited capacities and the other is for the law of the flow conservation. Constraints (2) and (3) represent the limit of the capacity for suppliers and manufactories in forward logistics. Constraint (5) shows that the total flows of forward and backward cannot exceed the total capacity of DC. Constraints (7) and (11) mean the reverse limit of the

capacity for DCs and dismantlers. Constraint (9) describes the customer recovery relationship with the recovery rate. Constraints (4), (6), (8) and (12) satisfy the law of the flow conservation by in-flow equal to out-flow. Constraint (10) is to satisfy the customer demand. Constraint (13) denotes the binary variables, and Constraint (14) is the non-negative, integral condition in our model.

The parameter of pd_k is used to describe the role of DC k. If $pd_k=0$, DC k has a sole duty for distribution in the forward logistics; and when $pd_k = 1$, DC k may play a singular role for Collection Center. If pd_k falls in between zero and one, it means DC k not only can be a distribution center, but also a collection center. The concept is similar for Manufactories. If a Manufactory accepts the resources from the dismantlers for reuses, it acts as both Manufacture and Remanufacture; otherwise it only uses raw materials for manufactory. These concepts have been presented in our general closed-loop logistics model.

Because the variables denote the basic units of logistics, apart from 0–1 decision variables, all other variables are all integers and thus Gauss' symbols are introduced in the model. The floor and ceiling of Gaussian are defined below:

Definition 3.1. The floor or the ceiling of a real number x is an integer denoted by $\lfloor x \rfloor$ and $\lceil x \rceil$, respectively, and defined, respectively, as below:

$$\lfloor x \rfloor = \{\underline{x} | x \geq \underline{x}, \underline{x} \geq y, \forall y, \underline{x} \in I\}$$
$$\lceil x \rceil = \{\overline{x} | x \leq \overline{x}, \overline{x} \leq y, \forall y, \overline{x} \in I\}$$

Because of Gaussian, the model will be further transformed into a linear model.

3.2. The transformed integer linear programming model

To transform into a computable model, we propose the following process to transform two rates into linear forms as below:

Let us consider the Constraints (7) and (9) first. Since $\lfloor pd_k Sc_k \beta_k \rfloor$ and $\lceil pc_l \sum_k z_{kl} \rceil$ are the joint given input values of all parameters: pd_k, pc_l, and Sc_k; and $\sum_k z_{kl}$ equals to the demand of customer l, because customer demand must be satisfied by Constraint (10) and the optimal solution exists if and only if Constraint (10) equals to the lower bound, therefore, by rewriting $SP_k = \lfloor pd_k Sc_k \rfloor$ and $ZP_l = \lceil pc_l \sum_k z_{kl} \rceil$, Constraints (7), (9), and (10) can be transformed, respectively, into (7a), (9a) and (10a) as below:

$$\sum_m o_{km} \leq SP_k \beta_k, \quad \forall k \tag{7a}$$

$$\sum_k Rz_{lk} \geq ZP_l, \quad \forall l \tag{9a}$$

$$\sum_k z_{kl} = d_l, \quad \forall l \tag{10a}$$

As regards Constraints (11) and (12) which are different from the situation above with decision value, $\sum_k o_{km}$ are not known in advance, and also pl_m are the parameters related to the estimated landfilling amounts. To transform these two constraints with Gauss's symbol, three additional inequalities are needed as defined below.

$$OP_m \leq pl_m \sum_k o_{km}, \quad \forall m \tag{15}$$

$$OP_m \geq pl_m \sum_k o_{km} - \varepsilon, \quad \forall m \text{ where } \varepsilon \to 1^- \tag{16}$$

$$OP_m \in N \cup \{0\} \quad \forall m \tag{17}$$

$$\sum_j Rd_{mj} + OP_m \leq e_m \delta_m, \quad \forall m \tag{11a}$$

$$\sum_k o_{km} = \sum_j Rd_{mj} + OP_m, \quad \forall m \tag{12a}$$

where a very small real number $\varepsilon > 0$ is given to ensure inequality holds for integer solutions.

Then, the original model with Gauss's symbol can be transformed into an integer linear program (ILP) which is summarized as below:

Closed-loop logistics model (CLL model):

$$\begin{aligned} \min \quad TC = & \sum_i \sum_j s_{ij} x_{ij} + \sum_j \sum_k t_{jk} y_{jk} + \sum_k \sum_l u_{kl} z_{kl} \\ & + \sum_k \sum_m v_{km} o_{km} + \sum_m \sum_j w_{mj} Rd_{mj} \\ & + \sum_l \sum_k Ru_{lk} Rz_{lk} + \sum_j f_i \alpha_j + \sum_k g_k \beta_k \\ & + \sum_m h_m \delta_m + \varphi \sum_m OP_m \end{aligned} \tag{1}$$

Subject to

$$\sum_j x_{ij} \leq a_i, \quad \forall i \tag{2}$$

$$\sum_k y_{jk} \leq b_j \alpha_j, \quad \forall j \tag{3}$$

$$\sum_i x_{ij} + \sum_m Rd_{mj} = \sum_k y_{jk}, \quad \forall j \tag{4}$$

$$\sum_l z_{kl} + \sum_m o_{km} \leq Sc_k \beta_k, \quad \forall k \tag{5}$$

$$\sum_j y_{jk} = \sum_l z_{kl}, \quad \forall k \tag{6}$$

$$\sum_m o_{km} \leq SP_k \beta_k, \quad \forall k \tag{7a}$$

$$\sum_l Rz_{lk} = \sum_m o_{km}, \quad \forall k \tag{8}$$

$$\sum_k Rz_{lk} \geq ZP_l, \quad \forall l \tag{9a}$$

$$\sum_k z_{kl} = d_l, \quad \forall l \tag{10a}$$

$$OP_m \leq pl_m \sum_k o_{km}, \quad \forall m \tag{15}$$

$$OP_m \geq pl_m \sum_k o_{km} - \varepsilon \quad \forall m \tag{16}$$

$$\sum_j Rd_{mj} + OP_m \leq e_m \delta_m, \quad \forall m \tag{11a}$$

$$\sum_k o_{km} = \sum_j Rd_{mj} + OP_m, \quad \forall m \tag{12a}$$

$$\alpha_j, \beta_k, \delta_m \in \{0,1\}, \quad \forall j,k,m \tag{13}$$

$$x_{ij}, y_{jk}, z_{kl}, o_{km}, Rd_{mj}, Rz_{lk}, \quad OP_m \in N \cup \{0\}, \tag{14}$$

$$\varepsilon \to 1^- \quad \forall i,j,k,l,m \tag{17}$$

Table 1
The size and estimated constants of the example.

Suppliers	Manufactories	DCs	Customers	Dismantlers	pd_k (%)	pc_l (%)	pl_m (%)	φ
3	5	3	4	2	10	10	10	5

Table 2
Capacity, demand (in unit) and fixed cost (US$).

Supplier	Manufactory		DC		Customer	Dismantler	
Capacity	Capacity	Fixed cost	Capacity	Fixed cost	Demand	Capacity	Fixed cost
500	400	1800	870	1000	500	540	900
650	550	900	890	900	300	380	800
390	490	2100	600	1600	400		
	300	1100			300		
	500	900					

Table 3
Unit shipping cost for each stage (US$).

Supplier	Manufactory				
	1	2	3	4	5
1	5	6	4	7	5
2	6	5	6	6	8
3	7	6	3	9	6
Manufactory	DC				
	1	2	3		
1	5	8	5		
2	8	7	8		
3	4	7	4		
4	3	5	3		
5	5	6	6		
DC	Customer				
	1	2	3	4	
1	7	4	5	6	
2	5	4	6	7	
3	7	5	3	6	
Customer	DC				
	1	2	3		
1	3	7	4		
2	8	5	5		
3	4	3	4		
4	3	2	5		
DC	Dismantler				
	1	2			
1	3	2			
2	2	5			
3	3	3			
Dismantler	Manufactory				
	1	2	3	4	5
1	2	3	4	2	5
2	3	4	6	3	4

In this closed-loop logistics model, there are (I+2J+4K+2L+4M) constraints, and ($I\times J+J\times K+K\times L+L\times K+K\times M+M\times J+J+K+2M$) variables including ($J$+$K$+$M$) binary variables, in which additional 2M variables and M constraints are derived from transformation. With this structure, the number of variables is always more than that of constraints and thus the model is always feasible.

Table 4
The optimal solution of numerical example.

Objective value					29 848.00
x_{ij}	$x_{13}=40$	$x_{15}=460$	$x_{22}=415$	$x_{23}=60$	$x_{33}=390$
y_{jk}	$y_{22}=550$	$y_{31}=490$	$y_{51}=319$	$y_{52}=141$	
z_{kl}	$z_{12}=109$	$z_{13}=400$	$z_{14}=300$	$z_{21}=500$	$z_{22}=191$
o_{km}	$o_{11}=61$	$o_{21}=89$			
Rd_{mj}	$Rd_{12}=135$				
Rz_{lk}	$Rz_{11}=50$	$Rz_{22}=30$	$Rz_{32}=40$	$Rz_{41}=11$	$Rz_{42}=19$
α_j	$\alpha_2=1$	$\alpha_3=1$	$\alpha_5=1$		
β_k	$\beta_1=1$	$\beta_2=1$			
δ_m	$\delta_1=1$				
OP_m	$OP_1=15$				

3.3. An illustrative example

In this paragraph, we shall use a small example to illustrate the properties of the problem and the model.

Tables 1–3 are the given data. The example contains 3 suppliers, 5 manufactories, 3 distribution centers, 4 customers and 2 dismantlers. Five types of roles are involved with the respective numbers (recovery, landfilling, and percentage of capacity for reverse in DC) as shown in Table 1, and three rates are assumed to be equal with respect to each customer l, dismantler m, and DC k, respectively. Tables 2 and 3 list all the unit costs of operation and transportation, respectively.

In this example, with $I=3$, $J=5$, $K=3$, $L=4$ and $M=2$, there are 41 constraints, and 82 variables. Using both LINGO 8.0 and ILOG-CPLEX 7.0 with at most 1(s) elapsed time, we obtained the optimal solution as shown in Table 4 and Fig. 2:

From this numerical example, it can be seen that with the conservation law, all logistic units were reserved in the system of the forward and backward flows. Also, from this solution, it can be seen that only three manufactory sites out of five, two distribution centers out of three, and one dismantler out of two are needed to meet the overall demands of four customers and recycling. Thus, the model is able to serve optimal green supply chain management from economic viewpoint.

In the closed-loop logistics planning problem, the recovery and landfilling rates are the most critical yet uncertain factors. In this numerical example, they are assumed to be 10%. Hsu and Wang [17] have carried out parameter analysis to obtain the ranges of landfilling and recovery rates with the same optimal logistics pattern. Also, they pointed out that because of the recovery rate is more sensitive than

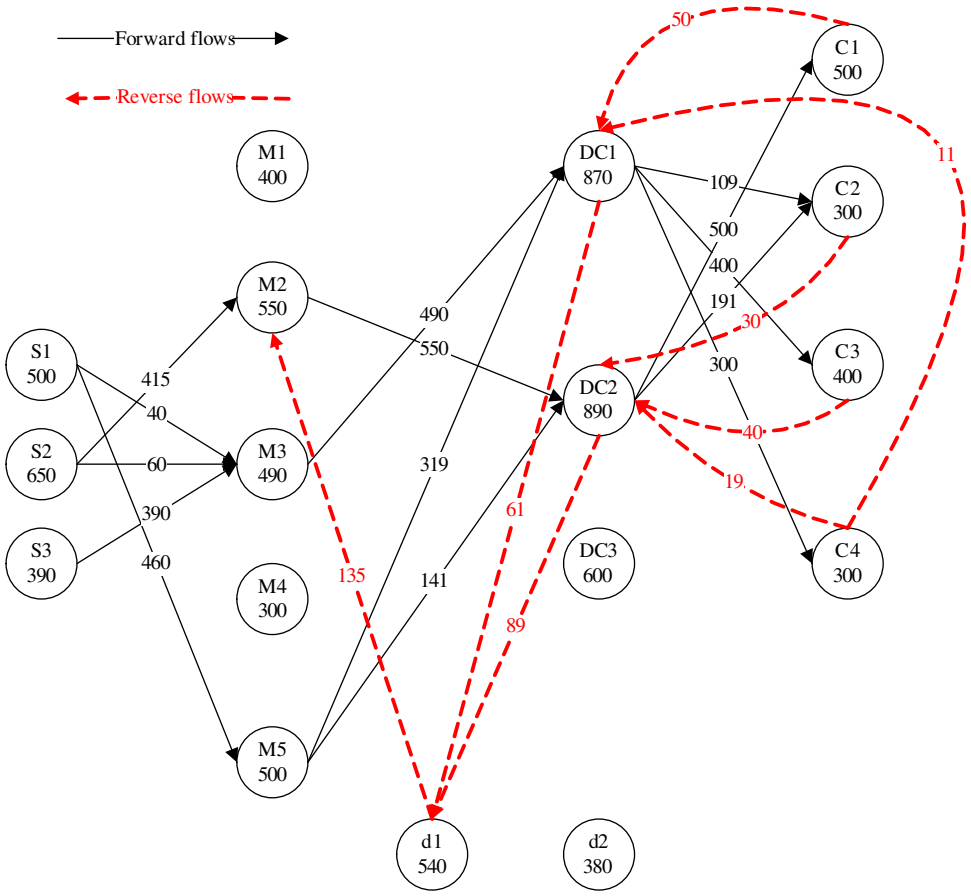

Fig. 2. Optimal distribution pattern of the illustrative example.

the landfilling rate, therefore, with a smaller tolerance range, any change of recovery rate should be given more attention in control and management.

4. Revised spanning-based genetic algorithm

Since the CLL model for the green logistics problem is a capacitated location-allocation problem; and also can be viewed as a multiple-choice Knapsack problem, it is known to be NP-hard [8,11,14,18]. Furthermore, the model is neither total unimodular [32], nor decomposable, therefore, an efficient algorithm should be developed to solve this model, which is the aim of this section.

From an enterprise' viewpoint, the logistics management should aim at minimizing cost or maximizing profit. To achieve this purpose, the strategy is to choose the right number and right locations of the facilities (e.g., manufactories, DCs, and dismantlers) to open. Although this kind of problem can be formulated into an integer linear program, it cannot provide a good solution for large-scaled problems within a short time. This becomes severe when the software programs use the Simplex-based algorithm with "exponential-time" complexity.

Based on the reviewed literature, we can conclude that two major shortcomings of the existing spanning-tree-based genetic algorithm needed to be improved and revised for our purposes of solution. The first is related to spanning-tree based encoding, and the second is genetic operations. Fig. 3 shows the flow diagram of our algorithm, and each state in the figure will be discussed in details below.

In the following illustrations, reference to the superscript numbers in Fig. 3 (like [a,b]) will be followed.

4.1. Revised determinant encoding

The spanning-tree based encoding normally is used in an acyclic problem. Although our problem lies in a cyclic logistics network, it is divided into several spanning-trees based on encodings between two consecutive stages which are not cyclic and the encoding is used to present the network structure. Since the property of our problem is not a real spanning-tree problem (we just use the encoding of spanning tree), we can relax the rules of spanning-tree properties and at the same time, improve the solution. The details of our encoding process will be explained as follow:

In a multistage supply chain network problem, it can be easily presented by a spanning tree. Fig. 4 shows the first two consecutive stages which have been presented by the spanning tree between suppliers and manufactories.

From the literatures, we have learned that both Prüfer and determinant encoding can be used for the encodings of the spanning tree problem in the framework of a GA. However, because the determinant encoding is a simple node-based indirect encoding strategy to overcome the bottlenecks of Prüfer encoding [1], we shall adopt determinant encoding in our study.

4.1.1. The initial chromosome and determinate encoding (Fig. 3[a])

The determinate encoding contains two parts: encoding and decoding.

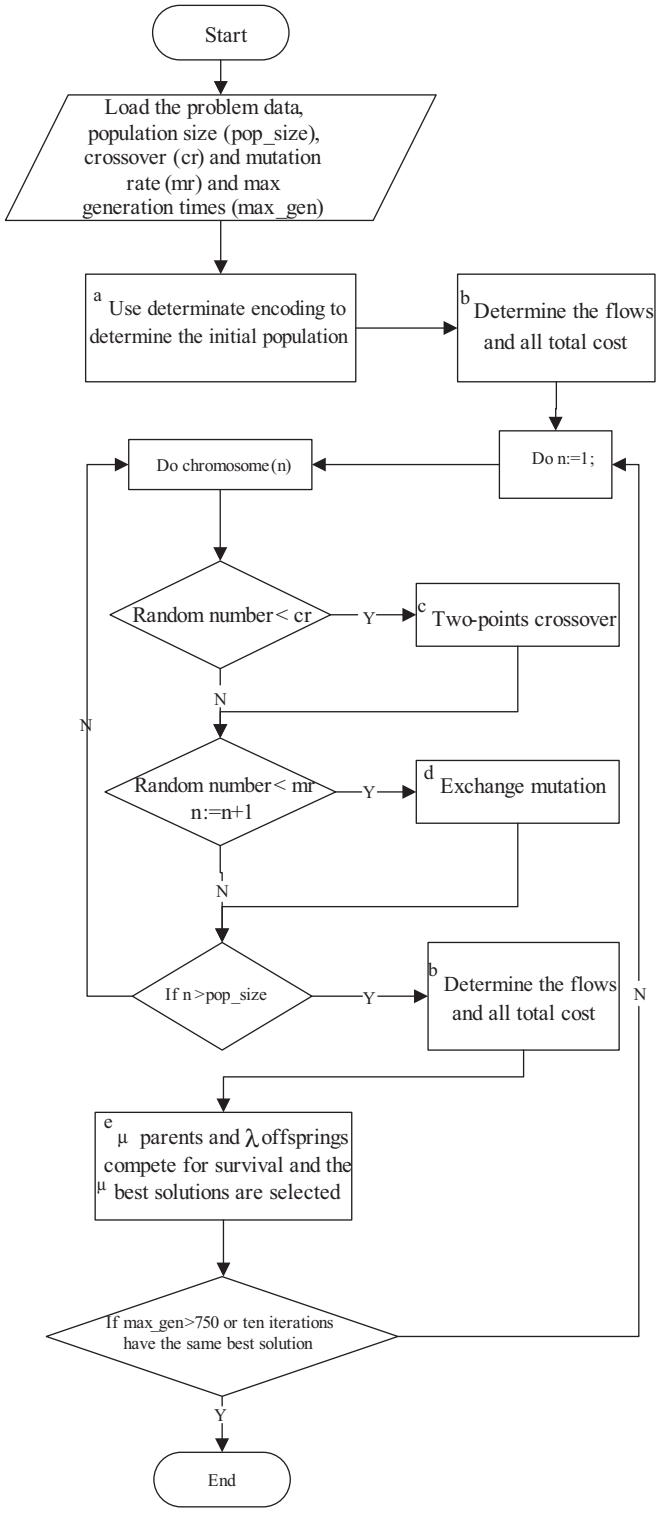

Fig. 3. The flow diagram of revised spanning-tree based GA.

Encoding:

Assume there are N nodes,

Step 1. Generate an $N-1$ length of determinate encoding;
Step 2. Use the *random* or *heuristic* method to set the codes. (The random or heuristic method will be explained at the initialization procedure below.)

Decoding:

The decoding algorithm treats each allele of the gene 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, the second as fixed-position 3, and so on.

The procedure of determinate decoding process is as follows:

Step 1. Let C be the given determination string and l be its length. If $C(j)$ is the jth allele in chromosome and $1 \leq j \leq l$, the number of the nodes in the given graph G is l+1, where a node is denoted as node (x) and $1 \leq x \leq l+1$.
Step 2. Set $j=1$, if $0 < j < l$+1, go to Step 3, else Stop.
Step 3. Connect node $(j$+1) with node $C(j)$, Set $j=j$+1, go back to Step 2.

For example, let $C=[3\ 4\ 2\ 5\ 8\ 5\ 3\ 9]$ represent a chromosome coded by the determinant encoding. It implies that there are nine nodes in the network corresponding to each fixed position [2 3 4 5 6 7 8 9] with the links (2,3), (3,4), (4,2), (5,5), (6,8), (7,5), (8,3) and (9,9) in the tree. Each of the fixed corresponding position is equal to the order of the gene plus one as shown in Fig. 5. The generated tree may not be legal and need to be repaired by reallocating genes at appropriate positions to generate a legal tree [5,33].

There are three '*illegal*' situations in the determinant encoding, and we can find all cases in Fig. 5. The first concerns "cycling," of which nodes 2, 3, and 4 are connected with a cycle. This means that a route starting from node 2 or 3 or 4 returns to itself and it will make the spanning tree illegal. The second is related to "reflexivity." It means that the node connects to itself as nodes 5 and 9, and it also makes the tree illegal. The last one is called "missing node 1," and it takes place when the chromosome does not contain node 1. Based on our proposed determinate encoding, only the last problem of the missing node 1 may happen in our problem, but it is comparatively easy to solve and will be discussed later.

After choosing the encoding type, we can start GA by evaluating the initial chromosome. Fig. 6 is an example of our chromosome presentation.

There are two parts in a chromosome that we must consider. Part 1 refers to the places that should be opened, and Part 2 is the evaluation of the determinant encoding for setting up the flows.

Part I*: Are the facilities open or not*?

Step 1. Randomly make the 0–1 values for the first three digits (for manufactories, distributions and dismantlers).
Step 2. Feasibility Test 1: If the total opened capacity satisfies the total customer demand, then the code of part I is finished and go to part II. Otherwise, go back to Step 1.

Part II*: Determinant encoding for setting up the flows*

We used a heuristic of revised determinant encoding to set up the flows in our problem. Based on the assumption (b), the units are restricted to flow only between different stages of the network. In order to ensure feasibility of a substring in determinant encoding, we should restrict the range of the encoding value. Using the flows between I suppliers and J manufactories as an example, the determinant encoding has $N-1$ chromosome length, with N nodes, and there are I+$J-1$ genes in a chromosome between suppliers and manufactories. For the first $I-1$ genes, let it be the number in between I+1 and I+J, and the last J genes be the number between 1 and I. The cases of "reflexivity" and "cycling" will never occur provided that the range of the encoding value is restricted to certain values. Therefore, there is only one case which needs to deal with and is the problem of "missing node 1". "Missing node 1" in determinant

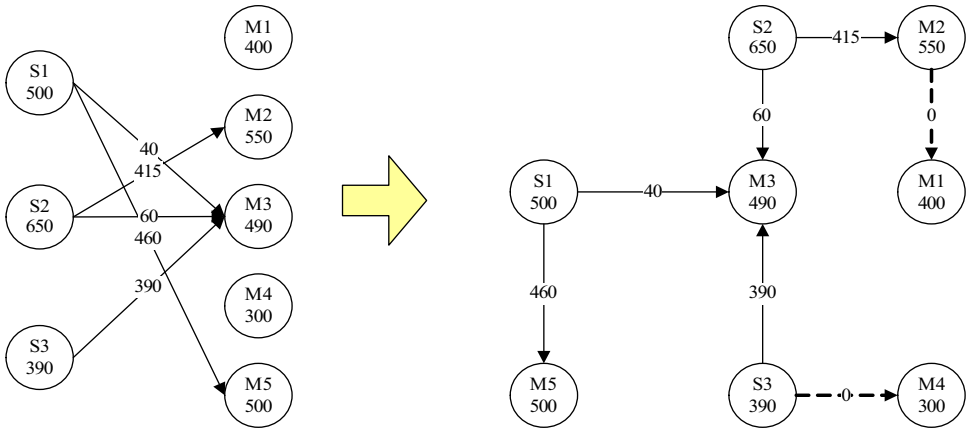

Fig. 4. The solution presented by a spanning tree.

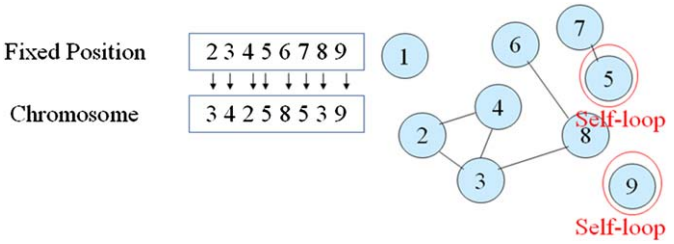

Fig. 5. Determinant encoding and the decoding process.

encoding will cause 'illegal' case of the first node absent from our code, so we must proceed validity test for it.

Validity test: In our problem, only without node 1 in the determinant encoding is counted as illegal and needed to be repaired. As the nodes correspond to the gene in the chromosome, the repairing process is to test the nodes at the fixed positions with the costs connecting to node 1, and the one with the minimum cost will be replaced by node 1. Thus, *missing node* 1 problem is resolved.

After encoding, the initial chromosome can be obtained by random and heuristic generation with the restrictions on the range of the encoding value as below.

The initialization procedure:

We use two kinds of algorithms for setting determinate encoding: one is to randomly set the genes by this process, and the other is a heuristic method. Let ρ% of the population be randomly generated, and $(100-\rho)$% of the population be heuristically generated.

1. By random generation: For the first ρ% chromosomes, we randomly generate the value of the positions in the chromosome but take into accounts of the restrictions on the range of the encoding value for each chromosome.
2. By heuristic generation: For the first $(100-\rho)$% percent chromosomes, we do the following steps. Let $q=2$.

(a) The heuristic method begins from the qth fixed position in the determinant encoding, of which the gene with the minimum cost is allocated to this fixed position. If more than one of the minimum points exists, any one will be arbitrarily chosen.

(b) Set $q=q+1$. If $q \leq I+J$ (If it is in the first stage of I suppliers and J manufacturers), then go to step 2(a); otherwise, stop.

In our research, the ratio for the use of heuristic setting and random setting is 9:1 ($\rho\% = 10\%$) in the initial population. The heuristic setting can effectively help us to find a good solution, and the random setting is used to avoid optimum generated from local population.

In conclusion, although the determinant encoding is an indirect encoding strategy, the decoding algorithm is very simple, and the only thing needed to be aware is to repair the "missing node 1" [33]. This step described above not only repairs the illegal node; but by connecting the node in the fixed position with a minimum cost, also provides an opportunity to improve the solution. This is impossible for Prüfer encoding.

4.1.2. The flows and cost of determinate encoding (Fig. 3^b)

We generate a random number stream to determine the order of flows and the details will be described by using the example between suppliers and manufactories with a stream set up by the random number between 1 and $i+j-1$.

Step 1. Use the random stream to determine which of the fixed positions will be chosen, then select the smaller capacity to be the flow between the fixed positions and the corresponding gene in the chromosome. This means that there is a need to assign the available amount of units to $x_{ij}=\min\{a_i, b_j\}$.

Step 2. Update the availability $a_i = a_i - x_{ij}$ and $b_j = b_j - x_{ij}$.

Step 3. If there is no available amount of units to assign, then stop; otherwise, there is a remaining supply of node r and demand of node s, then add edge (r, s) to the tree and assign the available amount of units $x_{rs}=\min\{a_r, b_s\}$ to the edge.

Determinate encoding with n nodes will have $(n-1)$ routings. In other words, I suppliers and J manufactories have at the most $(I+J-1)$ connection lines. Although the total possible routing number is $I \times J$, the objective of the optimal cost saving will result in minimal routing number. Based on the result, a spanning tree can be used to present the logistics network and avoid unnecessary routings and thus computation efficiency can be improved.

In a closed-loop supply chain, there are two kinds of systems, push for reverse and pull for forward. In the beginning of our GA, we must start from the push system to determine the reverse amounts, and then we can know how much the suppliers should assign to the manufactories in the pull system.

Push system: the reverse supply chain

The reverse logistics is a push system, and the flows are determined by the customers' recovered rates. Then starting from the DCs and dismantlers stages, the location and allocation will be ended at the manufactories and dismantlers stages.

Pull system: the forward supply chain

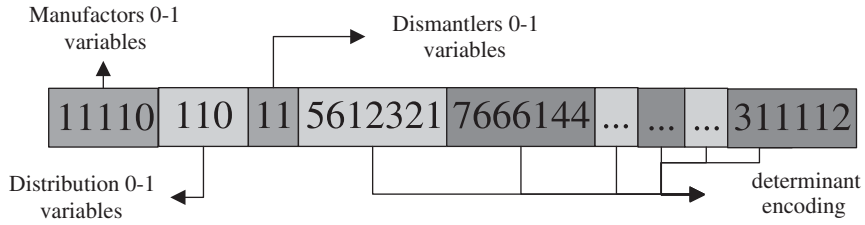

Fig. 6. The chromosome in revised spanning-tree-based GA.

The forward logistics is a pull system, and the flows are determined by the customer demand. Then through DCs and customers, the flows are allocated to the stages of manufactories and DCs exchange.

The stages of suppliers and manufactories have to be done after the pull and push processes are implemented. The flows will be evaluated at all stages, and the cost can be calculated by Eq. (1).

4.2. Genetic operations

Genetic operations are very important in genetic algorithms, and we focused on the crossover and mutation method, crossover and mutation rate, population size, fitness method or selection method, and termination condition for discussion. Research on these issues is enormous, and we adopt the one mostly commonly used in spanning tree problems in the following illustrations:

Based on the works of Yao and Hsu [33], Chou et al. [5], and Booker [22], the proper methods of crossover and mutation types for spanning tree problems are two-point crossover and exchange mutation.

The *two-point* crossover (Fig. 3[c]):

(1) Generates two random positions; head and tail.
(2) The alleles of the first chromosome from the head position to the tail are exchanged with the second chromosome in the same range.

The method of *exchange mutation* (Fig. 3[d]):

(1) Randomly selects two positions in a given chromosome.
(2) Exchanges both genes from these two random positions.

In order to make sure the feasibility after applying a crossover or mutation, we also used a sample heuristic here which can be referred to Appendix A.

From the experiments of Syarif et al. [30], the crossover rate is suggested to be 0.4, the mutation rate is equal to 0.2, and the population size is set to 100. For the population size, we will especially discuss the influence when the problem size increases in our numerical examples in the next section.

The selection method in Fig. 3[e] adopts the $(\mu+\lambda)$ method suggested by Chou et al. [5], of which μ parents and λ off springs compete for survival, and the μ best solutions are selected for the next generation.

Several termination conditions 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. Simultaneously, if the number of generations is greater than 750, we also stop the algorithms.

4.3. Summary of the proposed algorithm

Based on the above description, we use revised determinate encoding with heuristic initial population to overcome the bottlenecks of Prüfer encoding in a spanning-tree-based GA, and determine the cost and flows from reverse to forward logistics. By applying *two-point crossover* and *exchange mutation* with the $(\mu+\lambda)$ selection method, we can achieve better generation. This process repeats until the termination condition is satisfied. This revised algorithm can obtain better feasible solution in terms of high efficiency and reasonable accuracy. This evidence can be supported from the following examples when the comparison is done with optimization algorithms of LINGO 8.0 and CPLEX 7.0.

5. Evaluation of the algorithm

To test the accuracy and efficiency of the proposed algorithm, previous example was adopted as a base for comparison. To test the efficiency, different sizes of the test problems were used through doubling the numbers of the nodes at each stage as shown in Table 5; and running 30 times for each problem. A total of 150 experiments were executed by our algorithm. The results were compared with ILOG-CPLEX. These experiments were all done by a PC with Intel® Pentium® M processor 1.86 GHz, 1.0G RAM.

The test problem 1 is the illustrative example above of which $I=3$, $J=5$, $K=3$, $L=4$ and $M=2$. In test problem 2, $I=6$, $J=10$, $K=6$, $L=8$ and $M=4$, there are 82 constraints, and 304 variables (including 20 binary variables), and optimal solution is 58 306 by LINGO. In test problem 3, $I=12$, $J=20$, $K=12$, $L=16$ and $M=8$, there are 164 constraints, and 1168 variables (including 40 binary variables). The problem size increased to 328 constraints and 4576 variables in problem 4, and problem 5 reaches to 656 constraints and 18 112 variables. We can observe that the size of the problem increased immensely.

5.1. Preliminary evaluation

To evaluate the efficiency and accuracy of our algorithm, LINGO was first adopted for these purposes as it is the most commonly used software. The results are shown in Table 6. For Test Problem 3, after 10^9 iterations and 20 min(s) of elapsed runtime, LINGO failed to obtain the final solution, and so did test problem 4. For the problem 5, LINGO failed to find a feasible solution before 35 min, and after 40 min, a feasible solution was found. With our revised spanning-tree-based GA, two cases were considered: one was done with same population size of 100 for five test problems regardless of the problem size, and the other was done by increasing the population size with the problem size in order to obtain more accurate results. Table 6 summarizes the test results.

From Table 6, it can be observed that LINGO fails to solve such kind of large-scale problems, whereas our algorithm is capable of doing so. Besides, with our algorithm, increasing the population size with problem size only improved slight accuracy of the problem, yet

Table 5
The size of test problems.

Test problems	Suppliers	Manufactories	DCs	Customers	Dismantlers	pd_k (%)	pc_l (%)	pl_m (%)	φ
1	3	5	3	4	2	10	10	10	5
2	6	10	6	8	4	10	10	10	5
3	12	20	12	16	8	10	10	10	5
4	24	40	24	32	16	10	10	10	5
5	48	80	48	64	32	10	10	10	5

Table 6
Problem size with the same and different population sizes.

30 Times each problems	Test problem					
	1	2	3	4	5	
LINGO	Optimal (US$)	29 848	58 306	114 805 (feasible solution)	231 136 (feasible solution)	464 373 (feasible solution)
	Time (s)	6	13	> 1200	> 1200	> 2400
Revised ST-GA (population size = 100)	Min_cost (US$)	29 848	58 368	115 866	235 309	469 089
	[deviation]	[0]	[62]	[1061]	[4173]	[4716]
	Ave_cost (US$)	29 966.8	58 999.6	117 524.9	237 820	470 310
	Ave_time (s)	2.04	6.35	22.49	72.74	356.28
	[percentage of time]	[34%]	[48.85%]	[< 3.74%]		
Revised ST-GA (population size increase with problem size)	Pop_size	50	100	200	400	400
	Min_cost (US$)	29 848	58 368	114 926	233 546	463 956
	[deviation]	[0]	[62]	[121]	[2410]	[−417]
	Ave_cost (US$)	29 986.1	58 831.9	116 184.9	236 940	466 776
	Ave_time (s)	1.12	6.84	63.09	464.31	2324.42
	[percentage of time]	[18.67%]	[52.62%]	[< 10.52%]		

Deviation = ST-GA Min_cost–LINGO optimal.

Percentage of time = ST-GA Ave_time/LINGO time.

requires large computation time. Therefore, we do not have to use large population size to implement our algorithm as the problem size increases.

5.2. Advanced evaluation

While LINGO cannot solve large-scale problems with its branch-and-bound method; CPLEX is considered to be more efficient by branch-and-cut method, and thus was adopted for further experiments. The comparisons of our algorithm and CPLEX with error rates are shown in Table 7, of which the values in boldface are the best solutions.

In Table 7, we can see that although the CPLEX software is more efficient, it still cannot find the optimal solution in large problems. To evaluate accuracy, the error rates of our revised ST-GA with respect to the optimal solutions or out-of-memory feasible solutions obtained from CPLEX are less than about 1%. In test problem 5, although the operation time is higher, but we can even get a better result than CPLEX.

With regard to efficiency, because tests 4 and 5 ran out of memory earlier than test 3, we normalized run time with respect to the CPLEX time of test 3 in percentage. Fig. 7 shows the results of the evaluation: the left diagram shows the error rate of the revised ST-GA and the transformed time percentage of the revised ST-GA and CPLEX; the right diagram is an estimated situation for test 4 compared with tests 1, 2 and 3. From this evaluation, it is evident that even though the problem sizes were large, our algorithm showed a smaller error rate and less operation time than CPLEX. Therefore, the revised ST-GA can provide sufficiently accurate solutions with the efficient computation time for our closed-loop logistics problem.

In the previous experiments, we increased the population size in order to obtain a solution that is near-optimal or a feasible solution when out of memory incurred by the adopted software. However, increasing the population size rapidly increases operation time. The error rates of tests 3, 4 and 5 without increasing population size are shown in Table 8. The results showed that the error rates of population size increased a little, but the average operation time(s) decreased substantially without increasing population size. We can make the process more efficient and just sacrifice accuracy slightly when a population size equal to 100 is used. This facilitates real problem applications. For example, if an error is 2% is acceptable by a manager, then we may choose the population size of 100 for finding solution.

5.3. Discussion and summary

In order to investigate the influence of the problem structures on the solution performance, we further generated two test problems. That is, (1), 20 suppliers, 15 manufactories, 12 DCs, 50 customers and 5 dismantlers, and (2) 10 suppliers, 6 manufactories, 8 DCs, 100 customers and 5 dismantlers, respectively. These two test problems have 1852 variables and 1802 variables, respectively.

As LINGO cannot solve them, comparison with CPLEX is done and shown in Table 9. It can be seen that at the first test problem of 1852 variables, our algorithm obtains optimal solution. The error rate of our algorithm with the second test problem is less than 0.1%. However, the run times of both problems are less than that of CPLEX. From these additional experiments, we may confirm the accuracy and efficiency of the proposed algorithm.

Table 7
Comparisons of CPLEX and the revised ST-GA.

Problem	ILOG-CPLEX		Revised ST-GA	
	Min_cost (Ave)	Ave_time (s)	Ave_time (s)	Min_cost (Ave) [error rate]
Test1 (82 variables) (pop_size = 50)	**29 848**	**0.05**	1.12	29 848 (29 986.1) [0%]
Test2 (304 variables) (pop_size = 100)	**58 306**	**0.98**	6.84	58 368 (58 831.9) [0.106%]
Test3 (1168 variables) (pop_size = 200)	**11 4803** (feasible solution)	> 3923.8 (Out of memory)	**63.09**	114 926 (116 184.9) [0.107%]
Test4 (4576 variables) (pop_size = 400)	**231 120** (feasible solution)	> 983.13 (Out of memory)	**464.31**	233 546 (236 940) [1.05%]
Test5 (18112 variables) (pop_size = 400)	463 958 (feasible solution)	> **927.88** (Out of memory)	2324.42	**463 956** (466 776) [---]

Error rate = (GA–(CPLEX optimal or feasible solution when out of memory))/(CPLEX optimal or feasible solution when out of memory).

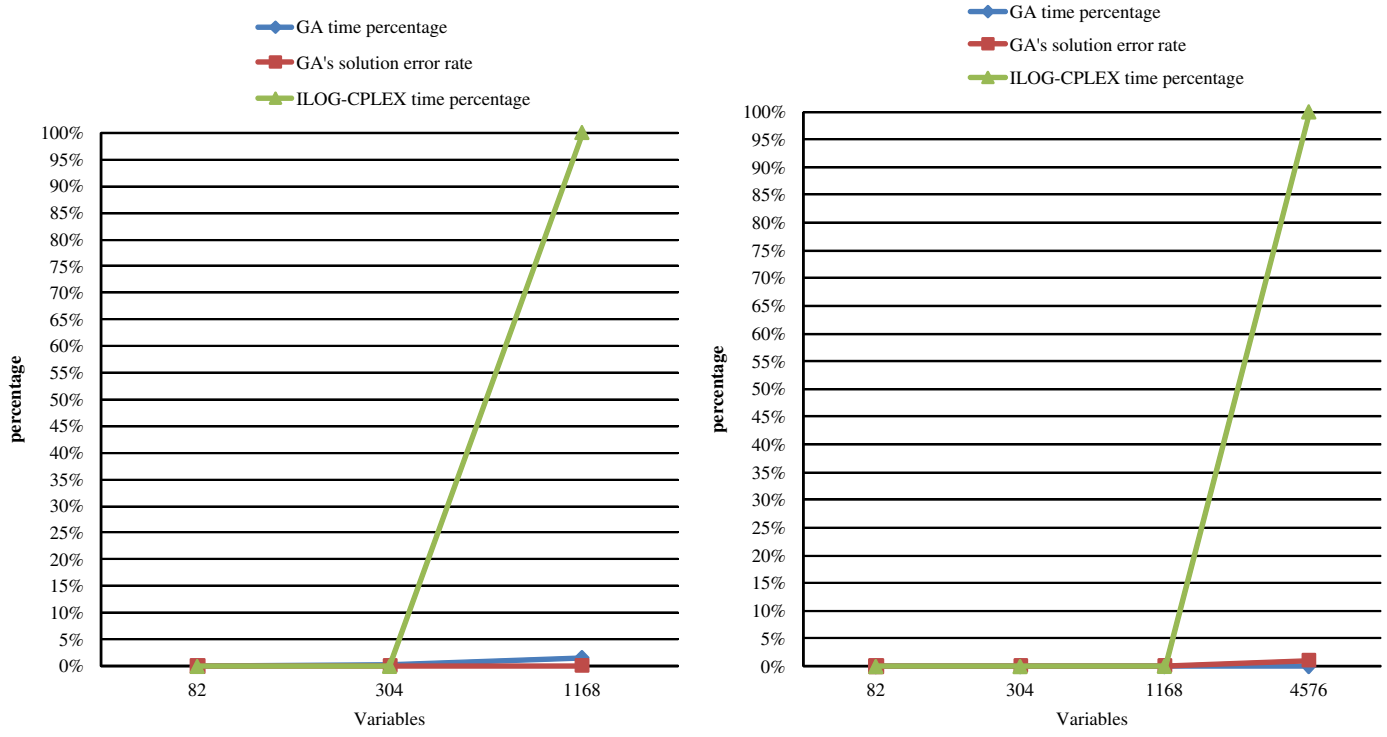

Fig. 7. Comparisons of error rate and time by percentage.

Table 8
The error rates of equal and different population sizes in tests 3 and 4.

Problem	Revised ST-GA (population size = 100)		Revised ST-GA (population size increase with problem size)	
	Error rate (%)	Ave_time	Error rate (%)	Ave_time
Test 3	0.926	22.49	0.107	63.09
Test 4	1.812	72.74	1.05	464.31
Test 5	1.11	356.28	0	2324.42

Furthermore, from our experiments, we have observed that although a larger population size can improve the solution, it consumes huge computation time. The "trade-off" between these is to find a suitable population size in the consideration of the error rate and time. Therefore, if we set the acceptable error rate in advance, the respective population size can be determined. In our experiments, 2% of the acceptable error rate was assumed, and thus the population size of 100 was used. In reality, to control the error and make

Table 9
Different problem structures compare between CPLEX and the revised ST-GA.

Problem	ILOG-CPLEX		Revised ST-GA	
	Min_cost (Ave)	Ave_time (s)	Min_cost (Ave) [error rate]	Ave_time (s)
(1) (1852 variables) (pop_size = 150)	**136178**	30.05	**136178** (137326.8) [0%]	**22.6**
(2) (1802 variables) (pop_size = 150)	**554713**	45.98	554809 (555264.3) [0.0173%]	**36.33**

Error rate = (GA−(CPLEX optimal or feasible solution when out of memory))/(CPLEX optimal or feasible solution when out of memory).

an effective decision are the most important concerns of a company, which can then be achieved by setting a suitable population size under a required accuracy using our algorithm.

In summary, the proposed revised spanning-tree based GA has demonstrated its performance in terms of both accuracy and efficiency.

6. Conclusion and future studies

Based on green issues, closed-loop logistics have become more and more important in recent years, and their resolution technologies have been critical for production companies. The reduction of primary resource use, pollution prevention, waste management, and policies governing sustainable products have thus become the focuses of modern industrial societies and environmental policies. Closed-loop logistics is one of the most essential keys in relation to the cost incurred by companies.

Since every part of the mentioned issues is related to the overall logistics in a product life-cycle system, closed-loop logistics with its overall cost has become an urgent concern for companies in their supply-demand chain management. Closed-loop supply chain management was then studied to integrate conventional forward logistics with reversed logistics. Although there have been some results in the literature, the lack of realistic features has drawn our attention to this study.

Based on the properties of closed-loop logistics that include both forward and reverse logistics, in this study, we have proposed a mathematical programming model for general applications. To retain the integral properties, the model was formulated with Gaussian symbols that are not algebraically computable. Therefore, a transformation procedure was proposed to convert the model into an integer linear programming model with additional $2M$ decision variables and M constraints. Although the transformed model facilitates analytical solutions, due to its NP property, the size often becomes too large and complicate to solve. GA is a heuristic algorithm. By appropriate design, it is capable of solving this kind of problem even without transformation.

Therefore, based on a determinant encoding approach, an efficient algorithm was proposed to revise the existing spanning-tree based genetic algorithm. The algorithm has been evaluated and compared with LINGO and CPLEX for its accuracy, capability, and efficiency. The results showed that the proposed algorithm is able to find a good solution efficiently for a closed-loop logistics.

For the future research, the uncertainty embedded in demand and recovery rates should be examined in a more analytical way to facilitate practical applications.

Acknowledgment

The authors acknowledge the financial support from the National Science Council, Taiwan, ROC with project number NSC95-2221-E007-213.

Appendix A. The heuristic to ensure the feasibility after crossover and mutation

Assumption (b) limited flows must exist between different stages without generality of infeasibility and illegal situations. Similar, Crossover and Mutation must be hold by *restriction condition*. We use $I+J-1$ genes in a chromosome between I suppliers and J manufactories for an example.

(Two-point crossover)

Step 1. Random selected a “head”.
Step 2. Judge the position of the “head”. If the head is in the first $I-1$ genes, then go to step 3. Else if the head is in the last J genes, then go to step 4.
Step 3. Random select a “tail” in the first $I-1$ genes.
Step 4. Random select a “tail” in the last J genes.

(Exchange mutation)

Step 1. Random selected the first position.
Step 2. Judge the location of the first position. If the first location is in the first I-1 genes, then go to step 3. Else if the first position is in the last J genes, then go to step 4.
Step 3. Random select the second position in the first I-1 genes.
Step 4. Random select the second position in the last J genes.

References

[1] Abuali FN, Wainwright RL, Schoenefeld DA. Determinant factorization: a new encoding scheme for spanning trees applied to the probabilistic minimum spanning tree problem. In: Eshelman LJ, editor. Proceedings of the sixth international conference on genetic algorithms. San Mateo, CA: Morgan Kaufmann; 1995. p. 470–7.
[2] Baker Barrie M, Ayechew MA. A genetic algorithm for the vehicle routing problem. Computers and Operations Research 2003;30:787–800.
[3] Baumgarten H, Christian B, Annerous F, Thomas S-D. Supply chain management and reverse logistics—integration of reverse logistics processes into supply chain management approaches. In: International symposium on electronics and the environment, 2003. p. 79–84.
[4] Bowersox DJ, Closs DJ. Logistical management: the integrated supply chain process. New York: McGraw-Hill; 1996.
[5] Chou H, Premkumar G, Chu CH. Genetic algorithm for communications network design—an empirical study for the factors that influence performance. IEEE Transactions on Evolutionary Computation 2001;5(3) June.
[6] De Groene A, Hermans M. Economic and other implications of integrated chain management: a case study. Journal of Cleaner Production 1998;6:199–211.
[7] Dengiz B, Altiparmak F, Smith AE. Local search genetic algorithm for optimal design of reliable networks. IEEE Transactions on Evolutionary Computation 1997;1:179–88.
[8] Eckert C, Gottlieb J. Direct representation and variation operators for the fixed charge transportation problem. In: Guervós JJM, Adamidis P, Beyer H-G, Martín JLF-V, Schwefel H-P, editors. Parallel problem solving from nature—PPSN VII, Proceedings of seventh international conference, Granada, Spain; 2002, September: 7–11, Lecture notes in computer science, vol. 2439. Berlin: Springer, p. 77–87.

[9] Fleischmann M, Beullens P, Bloemhof-Ruwaard JM, Van Wassenhove LN. The impact of product recovery on logistics network design. Production and Operations Management 2001;10(2):156–73.
[10] Fleischmann M, Jacqueline MB-R, Rommert D, Erwin van der Laan JAEE, van Nunen, Van Wassenhove LN. Quantitative models for reverse logistics: a review. European Journal of Operational Research 1997;16:1–17.
[11] Gen M, Cheng R. Genetic algorithms and engineering design. New York: Wiley; 1997.
[12] Goldberg DE. Genetic algorithms in search optimization, and machine learning. Reading, MA: Addison-Wesley; 1989.
[13] Gottlieb J, Julstrom BA, Raidl GR, Rothlauf F. Prüfer numbers: a poor representation of spanning trees for evolutionary search. In: SAP Proceedings of the genetic and evolutionary computation conference; San Francisco, CA; 2001. p. 343–50.
[14] Gottlieb J, Paulmann L. Genetic algorithms for the fixed charge transportation problem. In: Council INN, editor. The 1998 IEEE international conference on evolutionary computation; 1998. p. 330–35.
[15] In: Guide Jr VDR, Van Wassenhove LN, editors. Business aspects of closed-loop supply chains. Pittsburgh, PA: Carnegie Mellon University Press; 2003.
[16] Prüfer H. Neuer beweis eines satzes uber permutation. Archives of Mathematical Physics 1918;27:742–4.
[17] Hsu H-W, Wang H-F. Modeling of green supply logistics. In: Wang HF, editor. Web-based green products life cycle management systems: reverse supply chain utilization. USA: IGI Global Publication; 2009. p. 268–82.
[18] Jo J-B, Yinzhen L, Gen M. Nonlinear fixed charge transportation problem by spanning tree-based genetic algorithm. Computer and Industrial Engineering 2007;53:290–8.
[19] Krikke H, Bloemhof-Ruwaard J, Van Wassenhove LN. Concurrent product and closed-loop supply chain design with an application to refrigerators. International Journal of Production Research 2003;41(16):3689–719.
[20] Krishnamoorthy M, Ernst AT, Sharaiha YM. Comparison of algorithms for the degree constrained minimum spanning tree. Technical Report, CSIRO Mathematical and Information Sciences, Clayton, Australia; 1999.
[21] van der Laan E, Salomon M, Dekker R, Van Wassenhove L. Inventory control in hybrid systems with remanufacturing. Management Science 1999;45(5): 733–47.
[22] Booker L. Improving search in genetic algorithm. In: Davis L, editor. Genetic algorithms and simulated annealing. San Mateo, CA: Morgan Kaufmann; 1987. p. 61–73.
[23] Lu Q, Vivi C, Julie AS, Taylor R. A practical framework for the reverse supply chain. International Symposium on Electronics and the Environment 2000; 266–71.
[24] Min H, Ko HJ, Ko CS. A genetic algorithm approach to developing the multi-echelon reverse logistics network for product returns. Omega 2006;34(1): 56–69.
[25] Pochampally KK, Surendra MG, Sagar VK. Identification of potential recovery facilities for designing a reverse supply chain network using physical programming. The International Society for Optical Engineering 5262, Environmentally Conscious Manufacturing III 2004; 139–46.
[26] The European Working Group on Reverse Logistics (REVLOG) 1999. ⟨http://www.fbk.eur.nl/OZ/REVLOG/PROJECTS/TERMINOLOGY/def_reverselogistics.html⟩.
[27] Doris S, Cortés CE, Núñez A. Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering. Computers and Operations Research 2008;35:3412–38.
[28] Salema MIG, Barbosa-Povoa AP, Novais AQ. An optimization model for the design of a capacitated multi-product reverse logistics network with uncertainty. European Journal of Operational Research 2007;179(3):1063–77.
[29] Schultmann F, Moritz Z, Otto R. Modeling reverse logistic tasks within closed-loop supply chains: an example from the automotive industry. European Journal of Operational Research 2006;171:1033–50.
[30] Syarif A, Yun YS, Gen M. Study on multi-stage logistic chain network: a spanning tree-based genetic algorithm approach. Computer and Industrial Engineering 2002;43:299–314.
[31] Üster H, Easwaran G, Çetinkaya EAS. Benders decomposition with alternative multiple cuts for a multi-product closed-loop supply chain network design model. Naval Research Logistics 2007;54(8):890–907.
[32] Wolsey LA. Integer programming. USA: Wiley-Interscience Publication; 1998.
[33] Yao M-J, Hsu H-W. A new spanning tree-based genetic algorithm for the design of multi-stage supply chain network with nonlinear transportation costs. Optimization and Engineering 2009;10(2):219–37.
[34] Yeh W-C. A hybrid heuristic algorithm for the multistage supply chain network problem. Internal Journal of Advanced Manufacturing Technology 2005;26: 675–85.
[35] Zhu Q, Raymond PC. Integrating green supply chain management into an embryonic eco-industrial development: a case study of the Guitang Group. Journal of Cleaner Production 2004;12:1025–35.