Contents lists available at ScienceDirect

Expert Systems with Applications

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

An integrated approach to the design and operation for spare parts logistic systems

Muh-Cherng Wu [*], Yang-Kang Hsu, Liang-Chuan Huang

Department of Industrial Engineering and Management, National Chiao Tung University, Hsin-Chu, Taiwan, ROC

ARTICLE INFO

Keywords:
Logistic network design
Bill of material
Spare parts
Genetic algorithm
Neural network
Tabu search

ABSTRACT

This paper attempts to solve a comprehensive design problem for a spare part logistic system. The design factors encompass *logistic network design, part vendor selection,* and *transportation modes selection.* Two approaches to solve the problem were proposed. In Approach 1, we simultaneously considered all the design factors and proposed two algorithms (SGA-1 and TGA-1). In Approach 2, the design problem was solved in two stages. Firstly, we aimed to find a near-optimal logistic network. Secondly, with the obtained *logistic network,* we proposed three algorithms (SGA-2, TGA-2, and NN-GA-Tabu) to find optimal combinations for part vendor and transportation modes selection. Numerical experiments indicate that Approach 2 outperforms Approach 1, and the NN-GA-Tabu outperforms all the other four algorithms. The proposed NN-GA-Tabu might also be a good solution architecture for solving other comprehensive space search problems.

1. Introduction

Spare part management is a very important issue for capitally-intensive industries (e.g., semiconductor manufacturing, aerospace, defense, and high-speed train). Building a leading-edge semiconductor wafer fab may cost up to 2 billion dollars; and the associated spare parts inventory may need 10–15% of the total expenditure. Other capitally-intensive industries also reveal the same characteristics. Thus, the design and operation of a spare part logistic system is very important for these industries.

A spare part logistic system (also called a *logistic network*) typically involves a group of stations that are hierarchically structured as shown in Fig. 1. In the hierarchy, terminal stations, essentially designed to repair machines in the service field, are equipped with machine-repairing staffs and spare parts inventory. Other higher-layer stations are designed to store and repair spare parts in order to supply spare parts to terminal stations. Parts delivery between any two stations needs a transportation time. In literature, such a logistic network is characterized as a *multi-echelon system* (Sherbrooke, 1968)

As shown in Fig. 2, a machine typically comprises a hierarchical assembly of parts – called *bill of materials* (BOM). In literature, a spare part logistic system that considers only one kind of part is called a *single-indenture* system. In contrast, a *multi-indenture* system is a spare part logistic system that considers a BOM hierarchy involving many kinds of parts. This research is concerned with a *multi-indenture, multi-echelon* (simply called MIME) spare part supply chain system.

Several survey papers on spare part logistics in a MIME system have been published (Guide & Srivastava, 1997; Kennedy, Patterson, & Fredendall, 2002). Prior studies could be essentially grouped in two categories.

One category aimed to find optimal *operation policies* for a given spare part logistic system; that is, how to determine optimal inventory level and repair-staff level for each station in order to reduce the total operational cost. Some assumed that each station is equipped with an *infinite staffing capacity for repairing parts*; and paid attention to the decision of *stocking levels.* The pioneer one is the METRIC model developed by Sherbrooke (1968); many of its extensions have been developed (e.g., Graves, 1985; Muckstadt, 1973; Sherbrooke, 1986). Given a *finite staffing capacity for repairing parts*, some others investigated the decision for optimum *stocking levels* (e.g., Diaz & Fu, 1997; Kim, Shin, & Park, 2000; Perlman, Mehrez, & Kaspi, 2001). Extending the frontier, Sleptchenko, van der Heijden, and van Harten (2003) aimed to solve a more complex problem – finding an optimum combination for both repair-staff capacities and stocking levels.

The other category attempted to find an optimal design for a spare part logistic system. Some aimed to design an optimal logistic network (Candas & Kutanoglu, 2007; Jeet, Kutanoglu, & Partani, 2009; Rappold & van Roo, 2009); some focused on optimal selection of part vendors (Wu & Hsu, 2008); and some others examined optimal selection of transportation modes (Kutanoglu & Lohiya, 2008). Such design factors were only *partially* addressed in prior studies. Their obtained solutions might leave a space for further improvement if more design factors are simultaneously addressed.

* Corresponding author. Tel.: +886 35 731 913.
E-mail address: mcwu@mail.nctu.edu.tw (M.-C. Wu).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.08.088

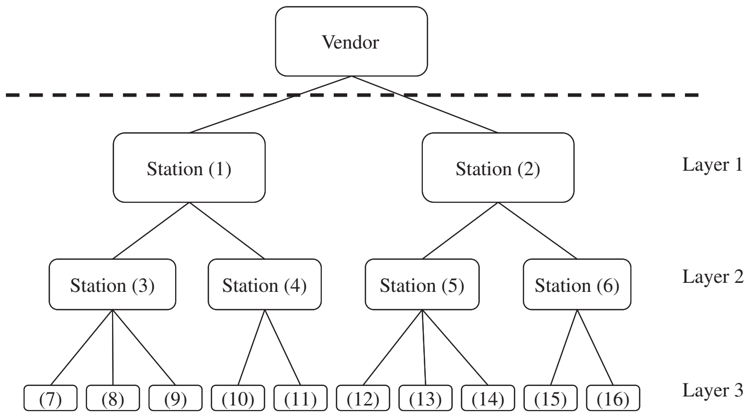


Fig. 1. The hierarchical structure of a logistic network.

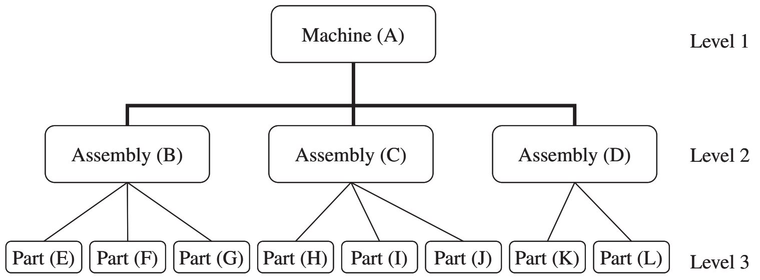


Fig. 2. The BOM hierarchy of each machine.

Yet, such a comprehensive inclusion of design factors may require formidable computational efforts.

In this paper, we attempt to solve a comprehensive design problem for a spare part logistic system. The design factors encompass *logistic network design*, *part vendor selection*, and *transportation modes selection*. Two approaches to solve the problem were proposed.

In Approach 1, all the design factors are *simultaneously* considered. That is, a new solution could be generated by varying the selection for any of the design factors. Based on such a solution representation, two meta-heuristic algorithms were proposed to solve the design problem. The two algorithms, adapted from literature (Goldberg, 1989; Tsai, Liu, & Chou, 2004), are respectively called SGA-1 (simple genetic algorithm in Approach 1) and TGA-1 (Taguchi genetic algorithm in Approach 1).

Approach 2 decomposes the design problems into two subproblems. That is, we solve the design problem in two stages. In stage 1, we focus on finding a near-optimal logistic network, by the application of a technically sound heuristic rule. In stage 2, with the obtained *logistic network*, we proposed three meta-heuristic algorithms to find optimal combinations for part vendor and transportation modes selection. The three algorithms are called SGA-2 (simple genetic algorithm in Approach 2), TGA-2 (Taguchi genetic algorithm in Approach 2), and NN-GA-Tabu (neural network-genetic algorithm-tabu-search).

Numerical experiments indicate that Approach 2 outperforms Approach 1. This advocates the use of a problem-decomposition approach in solving a large-scale problem, if a technically sound heuristic rule can be found. Of the three algorithms in Approach 2, the NN-GA-Tabu outperforms the other two both in solution quality and computation time. We developed the NN-GA-Tabu based on two ideas. First, we develop an *efficient* yet *rough* performance evaluator to quickly justify a solution. Second, we use GA to find a quality solution and then use a tabu-search (a local tuning process) to obtain an improved one.

The remainder of this paper is organized as follows: Section 2 describes the problem in more detail. Section 3 formulates the comprehensive design problem and analyzes possible ways to solve the problem. Section 4 describes the two algorithms in Approach 1. Section 5 describes the solution architecture of Approach 2 and the proposed NN-GA-Tabu algorithm. Experiment results of all the five algorithms are compared in Section 6. Concluding remarks are in the last section.

2. Problem statement

In this research, machines are capitally-intensive and their *availabilities* are very important. Machine availabilities are determined by the installing levels of two resources: (1) spare part inventory and (2) repair-staffs. Having a higher installing level for any of the two resources would lead to higher machine availabilities, yet at a price of incurring higher costs. How to make such a trade-off decision is critical to capitally-intensive industries.

As shown in Fig. 2, the BOM of a machine is a hierarchy comprising many assembly/parts. An assembly/part hereafter is called an *item*. The failure of each item follows a Poisson process. With long-lead times for acquisition, all items if failure need to be repaired. Repair time is an exponential distribution and first-come-first-serve policy is adopted.

The failure of any item in the BOM would lead to machine-down and reduce its availability. Quick replacement of the failure item can alleviate the effect of machine unavailability. This is achievable by installing a high stocking level, yet would incur higher inventory costs. By installing a higher level of repair-staffs, we would shorten the failure duration of items and consequently require a lower stocking level, yet at a price of increasing staffing costs.

A spare part logistic network is as shown in Fig. 1. Each node in the network is called a logistic station (simply called *station*). Each station is equipped with two kinds of resources: (1) spare part inventory and (2) repair-staffs. The purpose of a logistic network is to maintain a target average availability for machines in field, which are directly supported by *terminal* stations. Each station has one and only one *parent* station. Each station has a probability of successfully repairing an item. A failure item that cannot be successfully repaired by a station should be sent to its parent for repair.

Items-repairing may require various techniques. Therefore, different items may need different types of repair-staffs. An inventory replacement policy $(s, s-1)$ is adopted in each station (Feeney & Sherbrooke, 1966). Consider a case in which an inventory level s_{ij} is installed for item i at station j. Two features of this inventory replacement policy is explained below. First, for item i at station j, its total number of stocks (including failure ones) should always be kept at s_{ij}. Second, a failure stock at station j, if sent to its parent, should get a good unit back for exchange. Likewise, receiving a failure stock from its son station should give the son a good stock in exchange. A failure stock that cannot be repaired in the logistic network will ultimately be sent to its external vendor, who can always successfully repair the stock but requiring much longer lead time.

The logistic network for supporting a particular group of machines can be in various configurations. That is, given a generic network (Fig. 1), we can close some stations and reassign the parent–son relationships to create a network *instance* (Fig. 3). Consider a generic network that has E layers and each layer has $l_1, l_2, \ldots, l_E$ stations to open/close. Each station at layer e should be assigned to one parent in layer $e-1$, therefore, it has l_{e-1} possible assignments. As a result, all stations at layer e as a whole have $l_{e-1}^{l_e}$ possible assignments. This implies that the possible number of network instances is $\prod_{e=2}^{E} l_{e-1}^{l_e}$.

To reduce the total costs of a logistic network, we have two other design alternatives: (1) changing item vendors, (2) changing transportation mode for the path connecting each pair of parent and son stations. That is, consider a network instance that has s items and each item has k vendors to select. We have k^s possible choices in vendor selection. Such a vendor selection is a trade-off decision because the price of an item charged by a vendor with a lower failure rate is more expensive. Likewise, consider a network instance that has p paths connecting all pairs of parent–son-stations, and each path has m types of transportation modes. We then have m^p possible transportation configurations.

In summary, we have three decisions in the design of a logistic system: (1) logistic network instance selection, (2) item vendor selection and (3) transportation mode selection. Therefore, the possible number of design configurations is $\prod_{e=2}^{E} l_{e-1}^{l_e} \times k^s \times m^p$. Noticeably, in justifying the effectiveness of each design configuration, we need to determine its optimal operating conditions; that is, its optimal item stocking levels and repair-staffing levels.

3. Formulation and analysis

This section firstly formulates the research problem and proceeds to analyze possible ways to solve it.

3.1. Formulation

Sets and indices.

I = set of items, with index $i \in I$;
K = set of staff-types for repairing items, with index $c \in K$;
I_c = set of items that could be repaired by staff-type $c \in K$, with index $i \in I_c$;
Π = set of possible part vendors, with index $l \in \Pi$;
Γ = set of possible transportation modes, with index $t \in \Gamma$;
Λ = set of possible stations, with index j, $m \in \Lambda$;
Λ_τ = set of terminal stations, with index j, $m \in \Lambda_\tau$;
Λ_j^{up} = set of possible parent-stations for station $j \in \Lambda$, with index $m \in \Lambda_j^{up}$;
Λ_j^{down} = set of possible son-stations for station $j \in \Lambda$, with index $m \in \Lambda_j^{down}$;

3.2. Decision variables

$\overline{D} = \{[x_j], [y_{jm}], [z_{jt}], [v_{il}]\}$ = set of decision variables for logistic system design;
$\overline{O} = \{[k_{cj}], [s_{ij}]\}$ = set of decision variables for logistic system operation;

$$x_j = \begin{cases} 1, & \text{if station } j \in \Lambda \text{ is opened;} \\ 0, & \text{otherwise;} \end{cases}$$

$$y_{jm} = \begin{cases} 1, & \text{if station } m \in \Lambda \text{ is assigned as the parent of station } j \in \Lambda; \\ 0, & \text{otherwise;} \end{cases}$$

$$z_{jt} = \begin{cases} 1, & \text{if the path between station } j \in \Lambda \text{ and its parent is through transportation mode } t \in \Gamma; \\ 0, & \text{otherwise;} \end{cases}$$

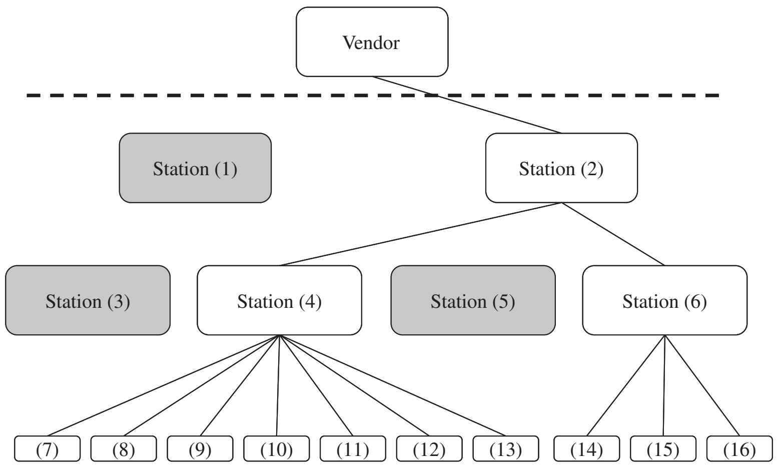


Fig. 3. An example design of logistic networks.

$$v_{il} = \begin{cases} 1, & \text{if item } i \in I \text{ is supplied by part vendor } l \in \Pi; \\ 0, & \text{otherwise;} \end{cases}$$

$\bar{k} = [k_{cj}]$, where k_{cj} = installed capacity of staff-type $c \in K$ at station $j \in \Lambda$;
$\bar{s} = [s_{ij}]$, where s_{ij} = the base stock level of item $i \in I$ at station $j \in \Lambda$;

3.3. Derived variables

$A^{avg}(\overline{D}, \overline{O})$: average machine availability for a design $\overline{D}$ operated at $\overline{O}$;
$\lambda_{ij}(\overline{D})$: mean arrival rate of item $i \in I$ at station $j \in \Lambda$ for a deign $\overline{D}$;

3.4. Parameters

A^{obj} = target average availability of machines;
p_{ij}^{rep} = the probability that item $i \in I$ could be repaired at station $j \in \Lambda$;
$p_{il}^{failure}$ = failure rate of item $i \in I$ while supplied by vendor $l \in \Pi$;
n_i^{item} = total number of item $i \in I$ per machine;

$$n_j^{machine} = \begin{cases} \text{total number of machines at station } j \in \Lambda_\tau; \\ 0, \qquad \text{if } j \notin \Lambda_\tau; \end{cases}$$

μ_{ij} = the mean repair rate of item $i \in I$ at station $j \in \Lambda$;
c_j^{loc} = the fixed cost of opening station $j \in \Lambda$;
$c_{j,m,t}^{trans}$ = the transportation cost from station j to station m by transportation mode t;
$t_{j,m,t}^{trans}$ = the transportation time from station j to station m by transportation mode t;
c_c^{cap} = cost of adding one more staff for staff-type $c \in K$;
c_{il}^{stock} unit inventory holding cost of item $i \in I$ supplied by vendor $l \in \Pi$;

3.5. Formulation

$$\textbf{Minimize} \sum_{j\in\Lambda} x_j \cdot c_j^{loc} + \sum_{j\in\Lambda}\sum_{m\in\Lambda}\sum_{t\in\Gamma} y_{jm} \cdot z_{jt} \cdot c_{j,m,t}^{trans} + \sum_{j\in\Lambda}\sum_{c\in K} k_{cj} \cdot c_c^{cap} + \sum_{i\in I}\sum_{j\in\Lambda}\sum_{l\in\Pi} s_{ij} \cdot v_{il} \cdot c_{il}^{stock}$$

s.t.

$$\sum_{m\in\Lambda} y_{jm} = 1, \quad \forall j \in \Lambda, \tag{1}$$
$$\sum_{m\in\Lambda_j^{up}} y_{jm} = 1, \quad \forall j \in \Lambda, \tag{2}$$
$$y_{jm} \leqslant x_m, \quad \forall j,\ \forall m \in \Lambda, \tag{3}$$
$$x_j = 1, \quad \forall j \in \Lambda_\tau, \tag{4}$$
$$\sum_{t\in\Gamma} z_{jt} = 1, \quad \forall j \in \Lambda, \tag{5}$$
$$\sum_{l\in\Pi} v_{il} = 1, \quad \forall i \in I, \tag{6}$$
$$\lambda_{ij}(\overline{D}) = \sum_{m\in\Lambda_j^{down}} (1 - p_{im}^{rep}) \cdot \lambda_{im}(\overline{D}) \cdot y_{mj} + \sum_{l\in\Pi} n_i^{item} \cdot n_j^{machine} \cdot p_{il}^{failure} \cdot v_{il}, \tag{7}$$
$$\sum_{i\in I_c} \lambda_{ij}(\overline{D}) \leqslant \mu_{ij} \cdot k_{cj}, \quad \forall j \in \Lambda,\ \forall c \in K, \tag{8}$$
$$A^{avg}(\overline{D}, \overline{O}) = Queueing\_Network(\overline{D}, \overline{O}), \tag{9}$$
$$A^{avg}(\overline{D}, \overline{O}) \geqslant A^{obj}, \tag{10}$$
$$k_{cj},\ s_{ij} \in Z, \quad \forall i \in I,\ \forall j \in \Lambda,\ \forall c \in K, \tag{11}$$
$$x_j,\ y_{jm},\ z_{jt},\ v_{il} \text{ are binary variables.} \tag{12}$$

In the above formulation, the objective function is to minimize total logistic costs, which involve: opening costs of logistic stations, transportation costs, costs of equipping repair-staffs, and inventory holding costs of items. Constraints (1) and (2) denote that each station has only one parent. Constraint (3) ensures that a station that has been closed cannot be a parent. Constraint (4) denotes that each terminal station should be opened. Constraint (5) denotes that only one transportation mode can be selected for each path. Constraint (6) denotes that only one vendor can be selected for each item. Constraint (7) describes a *recursive* formula for computing mean arrival rate for each item at each station. Constraint (8) defines the minimum capacity for each type of repair-staff. Constraint (9) denotes that average machine availability for a particular design/operation option can be obtained by a queuing network model. Constraint (10) defines the target availability. Constraints (11) and (12) ensure decision variables are in valid ranges.

3.6. Analysis of solution approaches

The formulation is a nonlinear program, in which constraint (7) is a recursive formula and constraint (9) is a complicated procedure which cannot be expressed by an explicit function. Therefore, we cannot solve the problem by analytical methods.

The problem by nature is a huge space search problem which involves two groups of decision variables – one group $\overline{D} = \{[x_j], [y_{jm}], [z_{jt}], [v_{il}]\}$ is for design optimization and the other group $\overline{O} = \{[k_{cj}], [s_{ij}]\}$ is for operational optimization. To effectively justify a given $\overline{D}$, we have to know its optimal $\overline{O}$. Sleptchenko et al. (2003) has developed a technique to determine an optimal $\overline{O}$ for a given $\overline{D}$. Therefore, we consider this problem as a space search problem that involves only $\overline{D}$; and the minimum operational costs for each $\overline{D}$ is obtainable by computing its optimal $\overline{O}$. To solve such a space search problem, we naturally consider the use of meta-heuristic algorithms.

Two approaches to solve the problem were proposed. In Approach 1, all the design factors $\overline{D} = \{[x_j], [y_{jm}], [z_{jt}], [v_{il}]\}$ are *simultaneously* considered, and two meta-heuristic algorithms (SGA-1 and TGA-1) were developed. In Approach 2, we decompose the design problem into two sub-problems, which are proceeded in two stages. In stage 1, we focus on finding a near-optimal logistic network, $L^* = \{[x_j]^*, [y_{jm}]^*\}$. In stage 2, with the obtained *logistic network* L^*, we develop three meta-heuristic algorithms (SGA-2, TGA-2, NN-GA-Tabu) to find $S^* = \{[z_{jt}]^*, [v_{il}]^*\}$. Herein, S^* denotes an optimal combination for the part vendor and transportation modes selection decisions.

Of the meta-heuristic algorithms, SGA-1 and SGA-2 are adapted from traditional GAs (Gen & Cheng, 2000; Goldberg, 1989). TGA-1 and TGA-2 are adapted from an enhanced GA, which additionally embeds the Taguchi experimental design technique in a traditional GA. The NN-GA-Tabu, partly adapted from Wu and Hsu (2008) and partly adapted from Chiou and Wu (2009), embeds neural network and tabu-search techniques in a traditional GA.

The reasons why we proposed the development of TGA-1, TGA-2 and NN-GA-Tabu are described below. By a pilot study, determining an optimal $\overline{O}$ for a given $\overline{D}$ requires about 6 s in computation time. This implies that applying a typical meta-heuristic algorithm such as genetic algorithm (GA) would be computationally extensive. For example, evaluating 20,000 solutions – a relatively small number in a typical GA application even requires about 1.5 days.

To alleviate the computational issues, two techniques are considered. One technique is by embedding an experiment design paradigm to *efficiently* generate *quality* solutions. The TGA-1 and TGA-2 are applications of this technique. The other technique is by the application of neural network (NN) algorithm to develop an *efficient* yet *rough* performance evaluator for $\overline{D}$. With such a rough performance estimator, we can quickly justify much greater number of solutions to obtain a candidate list, and then use the accurate performance evaluator to select the best one from the list. Then, the obtained solution is further refined by a tabu-search process to get an improved one. The NN-GA-Tabu is an application of this technique.

4. Approach 1

This section describes the two algorithms (SGA-1 and TGA-1) in Approach 1. We first describe the chromosome (or solution) representation in the two algorithms; then present how to obtain the fitness (or quality) of a chromosome; and finally explain the basic ideas of the two algorithms.

4.1. Chromosome representation

In Approach 1, we simultaneously consider three types of design factors: *logistic network design*, *part vendor selection*, and *transportation mode selection*. A design solution (called a *chromosome*) is a string that comprises three *segments*. Each segment (a smaller string) represents a design alternative for a particular type of design factors. That is, a chromosome is represented by $X = [X^{net}|X^{vendor}|X^{trans}]$ where X^{net}, X^{vendor}, and X^{trans} are its three segments, each of which respectively models a type of design factor.

Segment $X^{net} = [g_1, \ldots, g_n]$ is intended to represent an alternative for *logistic network design*, where $g_j \in \Lambda_j^{up}$ is the *assigned parent* of station j and n is the total number of possible stations. Each station always has a parent but not vice versa. That is, a station may not have a son. A non-terminal station (see stations 1, 3, and 5 in Fig. 3) that does not have a son should be closed. Therefore, X^{net} denotes a particular logistic network – it describes which stations are closed and how stations are supported by their possible parents.

In addition, segment $X^{vendor} = [v_1, \ldots, v_k]$ is intended to represent an alternative for *part vendor selection*, where $v_i \in \Pi$ is a vendor selection for item i. Segment $X^{trans} = [h_1, \ldots, h_n]$ is intended to represent an alternative for *transportation mode selection*, where $h_j \in \Gamma$ is a transportation mode selection for station j. Notice that each element (g_j, v_i, or h_j) in a chromosome X is called a *gene*.

4.2. Fitness evaluation

Noticeably, a chromosome X just denotes a design solution. To evaluate the fitness of X (the solution quality), we have to obtain its optimal operating policies which include inventory stocking levels as well as repair-staffing levels at each station j. According to Sleptchenko et al. (2003), such optimal operating policies can be obtained. We therefore define the total logistic cost, under optimal operating policies, as the fitness of X.

4.3. Algorithms: SGA-1 and TGA-1

The two meta-heuristic algorithms (SGA-1 and TGA-1) are both based on a typical GA architecture, which involves four stages: *initial population generation, population evolution, population updating*, and *evolution termination*. Yet, in the stage of population evolution, the two algorithms are different in their ways of creating new chromosomes. Each of the four stages is explained below.

In stage 1 – *initial population generation*, we randomly create N valid chromosomes to form an initial population P_0. These created chromosomes should be *valid*, in the sense that each gene value must be in its designated set (i.e., $g_j \in \Lambda_j^{up}$, $v_i \in \Pi$, $h_j \in \Gamma$). The initial population will be iteratively updated, and the updated population at iteration t is called P_t.

In stage 2 – *population evolution*, various kinds of genetic operators are developed to create new chromosomes from P_t. In SGA-1, only two genetic operators: crossover and mutation are included. Other than the two, TGA-1 includes one more genetic operator (called the Taguchi operator). The mechanism for creating new chromosomes by each genetic operator is explained below.

In the crossover operator, we randomly choose two chromosomes (e.g., X_1 and X_2), by randomly sectioning each segment into two parts and swap their gene values, to create two new ones (e.g., X_3 and X_4). Define $X_i = \left[X_i^{net}\middle|X_i^{vendor}\middle|X_i^{trans}\right]$ for i = 1, 2, 3, 4. For each segment (X_i^{seg}) in X_i, define its two sectioning parts as $X_{i,a}^{seg}$ and $X_{i,b}^{seg}$. As shown in Fig. 4, new chromosomes X_3 and X_4 can be created by applying sectioning and swap operations on X_3 and X_4.

In the mutation operator, we randomly choose one chromosome (e.g., X_1), by randomly changing one of its gene values for each segment (i.e., X_1^{net}, X_1^{vendor}, X_1^{trans}), to create one new chromosome.

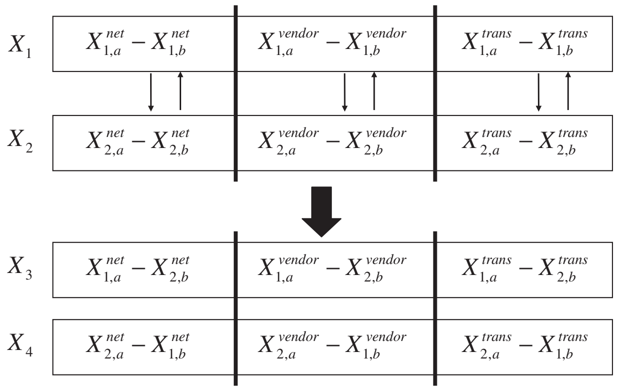
In the Taguchi operator, we randomly choose two chromosomes (e.g., X_1 and X_2), by applying the Taguchi experimental design method, to create one new chromosome (e.g., X_5). Suppose a chromosome has n genes in total. Each gene, of the two chromosomes X_1 and X_2, has at most two levels. Then, finding a good chromosome from all possible combinations of X_1 and X_2 can be seen as an experimental design problem, which has n factors and each factor has two levels. Using the orthogonal array experimental design, the Taguchi operator can efficiently obtain a new and good chromosome X_5.

In each iteration, the total number of newly created chromosomes is $N(P_{cr} + P_{mu} + P_{ta})$, where $0 \leqslant P_{cr}, P_{mu}, P_{ta} \leqslant 1$ represent parameters of various genetic operators. This implies that, at the end of stage 2, we have $N(1 + P_{cr} + P_{mu} + P_{ta})$ chromosomes in total.

In stage 3 – population updating, we use roulette wheel selection method (Michalewicz, 1996) to screen N chromosomes from the $N(1 + P_{cr} + P_{mu} + P_{ta})$ ones to form a new population. In stage 4 – *evolution termination*, the population evolution/updating procedures will be terminated if the iteration number has achieved an upper bound (i.e., $t = T_{max}$) or a best solution has sustained over a certain number of iterations (T_{best}).

5. Approach 2

Approach 2 is intended to solve the comprehensive design problems in two stages. In stage 1, we focus on finding a near-optimal logistic network. In stage 2, with the obtained *logistic network*, we use three meta-heuristic algorithms (SGA-2, TGA-2, and NN-GA-Tabu) to find optimal combinations for part vendor and transportation modes selection.



X_1
$X_{1,a}^{net} - X_{1,b}^{net}$
$X_{1,a}^{vendor} - X_{1,b}^{vendor}$
$X_{1,a}^{trans} - X_{1,b}^{trans}$
X_2
$X_{2,a}^{net} - X_{2,b}^{net}$
$X_{2,a}^{vendor} - X_{2,b}^{vendor}$
$X_{2,a}^{trans} - X_{2,b}^{trans}$
X_3
$X_{1,a}^{net} - X_{2,b}^{net}$
$X_{1,a}^{vendor} - X_{2,b}^{vendor}$
$X_{1,a}^{trans} - X_{2,b}^{trans}$
X_4
$X_{2,a}^{net} - X_{1,b}^{net}$
$X_{2,a}^{vendor} - X_{1,b}^{vendor}$
$X_{2,a}^{trans} - X_{1,b}^{trans}$

Fig. 4. The crossover operator and its corresponding chromosomes.

5.1. Stage 1: determining logistic network

In stage 1, the solution space of possible logistic networks can be quite large. Consider a logistic network that has E layers and $l_1, l_2, \ldots, l_E$ stations to open/close. As stated, the possible number of network instances is $Q = \prod_{e=2}^{E} l_{e-1}^{l_e}$; that is. $Q = 2^{24} = 1.68 \times 10^7$ for $E = 3$, $l_1 = 2$, $l_2 = 4$, and $l_3 = 10$.

To select an optimal one from the solution space, we need to evaluate each candidate solution. Yet, evaluating a candidate logistic network may be computationally extensive because it has a huge number of possible versions, due to the variations caused by the selection of item vendors and transportation modes. Consider a scenario with s items and p connecting paths, each item has k vendors and each path has m transportation modes to select. Then a logistic network has $V = k^s \cdot m^p$ possible versions; that is $V = 3^{22} = 3.14 \times 10^{10}$ for a typical application case with $m = k = 3$, $p = 10$, $s = 12$. Moreover, evaluating such a version may be also time-consuming, because for each possible version we need to compute its optimal operating policies.

To reduce the computational extensive issues, this research proposes one heuristic rule in finding a near-optimal logistic network. The heuristic rule, which appears to be *technically sound*, request that the parent of each station j is the one in Λ_j^{up} that is open and is the nearest one in transportation distance. With this heuristic rule, we only have to determine whether a station should be open or close. As a result, the solution space of possible logistic networks is greatly reduced. For a logistic network with E layers and $l_1, l_2, \ldots, l_E$ stations, its possible number of network instances now becomes $Q' = \prod_{e=1}^{E-1}\left(2^{l_e} - 1\right)$; that is. $Q' = 45$ if $E = 3$, $l_1 = 2$, $l_2 = 4$, and $l_3 = 10$.

For a logistic network, the number of its possible versions is quite huge ($V = k^s \cdot m^p$). We randomly sample 30 versions and compute the total logistic cost for each version while it is under optimal operating policies. That is, for a logistic network with E layers and $l_1, l_2, \ldots, l_E$ stations, we only have to evaluate $N' = 30Q' = 30\prod_{e=1}^{E-1}\left(2^{l_e} - 1\right)$ logistic network versions. For a case with $E = 3$, $l_1 = 2$, $l_2 = 4$, and $l_3 = 1$, we have $N' = 1350$ while the complexity of its original solution space is $N = QV = 2^{24} \cdot 3^{22} = 5.2 \times 10^{17}$.

5.2. Stage 2: selecting vendor and transportation modes

The output of stage 1 is a near-optimal logistic network (say, L^*). With the *logistic network* L^*, in stage 2, we aim to find an optimal combination for the decisions of part vendor and transportation modes selection. For this purpose, three meta-heuristic algorithms (SGA-2, TGA-2, and NN-GA-Tabu) were developed, in which a chromosome is now represented by a smaller string, say $X = [X^{vendor}|X^{trans}]$. The procedures of SGA-2 and TGA-2 are similar to that of SGA-1 and TGA-1 as described in Section 4. Thus we only explained the procedure of NN-GA-Tabu herein.

The procedure of NN-GA-Tabu comprises four major steps. Firstly, we apply the neural network technique to develop an *efficient* yet *rough* performance evaluator of a chromosome. Secondly, with such an efficient evaluator, we use a traditional GA to obtain a list of candidate solutions. Thirdly, we use the original performance evaluator, developed by Sleptchenko et al. (2003), to select the best one from the candidate list. Finally, the obtained solution is further refined by a tabu-search process, which is adapted from Glover and Laguna (1997). Each step is further explained below.

The development of the rough performance evaluator is based on the back-propagation (BP) neural network algorithm (Fausett, 1994), which has been widely used to emulate a system's behavior by a sampled data set. That is, for a system with vectors X_i as input and Z_i as its corresponding output, we can randomly sample n_1 pairs of $\{X_i, Z_i\}$ to train (or establish) a BP neural network, which can compute an estimated output $\widehat{Z}_i$ for each sampled X_i. Then n_2 pairs of $\{X_i, Z_i\}$ are randomly sampled to test the BP neural network. If the predicted error $e = \sqrt{\sum_{i=1}^{n_2} \frac{\left(Z_i - \widehat{Z}_i\right)^2}{n_2}}$ is acceptable, then the BP network can be used to emulate the system's behavior; that is, computing for $\widehat{Z}_i$ for other X_i. A detailed algorithm for developing such a BP neural network can be referred to Fausett (1994).

Based on the BP neural network algorithm, a procedure *Rough_Evaluator* is developed to quickly and roughly estimate the fitness of a chromosome, as stated below.

5.2.1. Procedure Rough_Evaluator

- Sample N_t chromosomes, say $X_i = \left[X_i^{vendor}|X_i^{trans}\right]$, $i = 1, \ldots, N_t$.
- Use the original evaluator to compute the fitness for each sampled chromosomes, $Z_i = Original\_Evaluate(X_i)$; $i = 1, \ldots, N_t$.
- Use the data set $\{X_i, Z_i | i = 1, \ldots, N_t\}$ to develop a BP neural network, where X_i is input and Z_i is output.
- Represent the developed BP neural network by $\widehat{Z}_i = Rough\_Evaluate(X_i)$.

Notice that, in the above procedure, the rough evaluator (or the BP neural network) is denoted by $\widehat{Z}_i = Rough\_Evaluate(X_i)$ and the original evaluator is denoted by $Z_i = Original\_Evaluate(X_i)$. The procedure NN-GA-Tabu can thus be stated below.

5.2.2. Procedure NN-GA-Tabu

Step 1: Establish a rough performance evaluator, $\widehat{Z}_i = Rough\_Evaluate(X_i)$.

Step 2: Develop a traditional GA that evaluates a chromosome by $\widehat{Z}_i = Rough\_Evaluate(X_i)$. By the GA, find a candidate set of chromosomes S, where

$$S = \left\{X_i(t_e), \widehat{Z}_i(t_e)\right\}, \quad i = 1, \ldots, N,$$

t_e is the iteration while the GA terminates.

Step 3: Find the best chromosome X_{i^*} in S, by the original evaluator

$$Z_i(t_e) = Original\_Evaluate(X_i(t_e)), \quad \text{for } i = 1, \ldots, N,$$
$$i^* = Arg \underset{1 \leqslant i \leqslant N_L}{Min} Z_i(t_e).$$

Step 4: Use a tabu-search process to refine X_{i^*}; that is

$$X_{i^\oplus} = Tabu\_Search(X_{i^*}), \quad \text{where}$$
$X_{i^\oplus}$ is the solution obtained by the tabu-search process.

Table 1
Station opening cost and the number of machines at each terminal station (cost: $K).

Station ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Layer ID	1	1	2	2	2	2	3	3	3	3	3	3	3	3	3	3
Open cost	6800	5000	2000	1800	1900	1200	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
# of machines	–	–	–	–	–	–	1	1	1	2	1	1	1	2	2	2

Table 2
Transportation times/costs between layers 1 and 2, operated under transportation mode 1 (time: hour, cost: $K).

	Vendor		Station 3		Station 4		Station 5		Station 6	
	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost
Station 1	240	2500	32	5400	48	15,000	64	29,400	34	15,000
Station 2	240	2500	80	48,600	48	15,000	32	5400	34	15,000

Table 3
Transportation times/costs between layers 2 and 3, operated under transportation mode 1 (time: hour, cost: $K).

	Station 3		Station 4		Station 5		Station 6	
	Time	Cost	Time	Cost	Time	Cost	Time	Cost
Station 7	24	2400	56	21,600	72	38,400	248	60,000
Station 8	32	5400	48	15,000	64	29,400	240	48,600
Station 9	40	9600	40	9600	56	21,600	232	38,400
Station 10	56	21,600	24	2400	40	9600	216	21,600
Station 11	64	29,400	32	5400	32	5400	224	29,400
Station 12	80	48,600	48	15,000	32	5400	240	48,600
Station 13	88	60,000	56	21,600	40	9600	248	60,000
Station 14	232	38,400	232	38,400	248	60,000	40	9600
Station 15	248	60,000	216	21,600	232	38,400	24	2400
Station 16	264	86,400	232	38,400	216	21,600	40	9600

Table 4
Unit costs for various types of repair-staffs.

Staff type	Repairing item type	Cost ($K)
Type 1	Level 1	800
Type 2	Level 2	620
Type 3	Level 3	480

6. Numerical experiments

Numerical experiments are carried out to justify the five algorithms (SGA-1, TGA-1, SGA-2, TGA-2, and NN-GA-Tabu) to solve the comprehensive design problem. We first describe tested scenario; then present the parameters of the five algorithms and the computing hardware; and finally report and analyze the experiment results.

The tested scenario, involving 16 stations in a three-layer hierarchy (Fig. 1), has a target machine availability $A^{obj} = 0.95$. Table 1 shows the opening cost for each station and the number of machines at each terminal station. The transportation times/costs, operated under transportation mode 1, are shown in Tables 2 and 3. We assume that the transportation times/costs, operated under different modes, are proportional; that is, $t^{trans}_{j,m,2}/t^{trans}_{j,m,1} = 0.75$, $t^{trans}_{j,m,3}/t^{trans}_{j,m,1} = 0.50$, $c^{trans}_{j,m,2}/c^{trans}_{j,m,1} = 1.20$, and $c^{trans}_{j,m,3}/c^{trans}_{j,m,1} = 1.50$. This implies the faster is a transportation mode the more expensive is its transportation cost.

The BOM hierarchy for each machine involves three levels (Fig. 2). Items in level i require type i repair-staffs and their unit staffing costs are shown in Table 4. Table 5 shows the quantity of each item as well as its failure rates and units costs while supplied by different vendors. The higher the failure rates, the lower is the unit cost. Each station is given a *repairable probability* – a probability of successfully repairing an item as shown in Table 6, which also gives repairing times.

Parameters of the five algorithms are set as follows: $N = 50$, $T_{best} = 100$, $T_{max} = 100$, $P_{cr} = 0.8$, $P_{mu} = 0.2$, and $P_{ta} = 0.04$. In NN-GA-Tabu, a BP neural network, with $e = 0.00795$, is established by using $n_1 = 1300$ and $n_2 = 200$. For each experiment, we run 10 replicates, by using personal computers equipped with 3.4G CPU and 504 MB RAM.

Table 7 compares the experiment results of the five algorithms, in terms of solution quality (total logistic cost) and computation time. The table indicates that Approach 2 significantly outperforms Approach 1 both in solution quality and computation time. Moreover, the qualities of solutions obtained in Approach 1 have a higher variation. That is, Approach 2 is not only more effective but also more robust.

The reason why Approach 2 outperforms may be due to the adoption of the heuristic rule – the parent of each station is just the nearest one in its upper layer. This heuristic rule appears to be technically sound. Therefore, a chromosome violating the heuristic rule tends to be inferior. In Approach 1, many such violating chromosomes are likely to be generated in the population evolu-

Table 6
Repairable probability and repairing time for stations at each layer.

	Layer 1	Layer 2	Layer 3
Repairable probability	0.8	0.7	0.6
Repairing time (h)	8	10	12

Table 5
Item failure rates and unit costs of different vendors.

Item	Quantity of each item	Failure rates provided by each vendor (number of failures per 10^6 h)			Unit cost charged by each vendor ($K)		
		FRT_1	FRT_2	FRT_3	C_1	C_2	C_3
A	1	2835	2948.4	3175.2	1300	1200	1000
B	1	2205	2293.2	2469.6	220	200	190
C	1	2520	2620.8	2822.4	310	280	250
D	1	2835	2948.4	3175.2	180	170	160
E	2	1575	1638	1764	40	38	35
F	1	1890	1965.6	2116.8	50	40	30
G	1	2205	2293.2	2469.6	30	25	20
H	1	1575	1638	1764	50	42	36
I	3	1890	1965.6	2116.8	35	30	25
J	1	2205	2293.2	2469.6	55	48	45
K	2	2835	2948.4	3175.2	40	30	20
L	2	2835	2948.4	3175.2	30	28	25

Table 7
Experiment results of Approach 1 and Approach 2 (time: hour, cost: $K).

Replicate	Approach 1				Approach 2					
	SGA-1		TGA-1		SGA-2		TGA-2		NN-GA-Tabu	
	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time
1	74,747	49.5	75,406	47.8	74,225	39.7	74,199	35.8	74,216	7.3
2	76,251	72.3	74,625	73.2	74,081	51.2	74,081	74.0	74,081	7.1
3	78,907	53.7	74,511	106.7	74,113	38.6	74,205	57.3	74,081	7.6
4	75,659	53.8	74,244	85.8	74,255	27.5	74,101	54.3	74,081	7.1
5	74,560	48.6	78,512	148.4	74,205	30.6	74,241	42.9	74,081	7.1
6	76,774	44.7	74,229	72.0	74,159	25.1	74,205	83.5	74,081	7.2
7	74,517	42.3	78,722	137.2	74,108	51.9	74,081	83.1	74,081	7.5
8	79,530	29.5	74,342	96.0	74,081	22.0	74,081	79.5	74,205	6.9
9	76,202	39.7	74,284	139.4	74,139	19.9	74,081	62.2	74,081	7.5
10	74,468	48.3	78,459	362.5	74,133	26.0	74,085	62.4	74,206	6.9
AVG	76,162	48.2	75,733	126.9	74,150	33.2	74,136	63.5	74,119	7.2
DEV	1815	11	1984	89	60	12	67	17	62	0.2

tion process. Then, the best solution in the chromosome population is less likely to be improved. As a result, the ultimately obtained solution tends to be inferior. This advocates the use of a problem-decomposition approach in solving a large-scale space search problem, if a technically sound heuristic rule can be found.

Of the three algorithms in Approach 2 (Table 7), the NN-GA-Tabu significantly outperforms the other two in terms of computation time, and slightly better than the other two in terms of solution quality. This indicates that the NN-GA-tabu is an *effective* and *efficient* method to solve the comprehensive logistic network design problem.

7. Concluding remarks

This paper examined a comprehensive design problem for a spare part logistic system, which involves the following decisions: *logistic network design*, *part vendor selection*, and *transportation modes selection*. In prior studies, these decisions were only *partially* addressed. A comprehensive inclusion of such design factors may require formidable computational efforts.

Two approaches were proposed to solve the problem. In Approach 1, we simultaneously considered all design factors and proposed two meta-heuristic algorithms (SGA-1 and TGA-1). In Approach 2, we decomposed the design problems into two sub-problems. The first sub-problem is to find a near-optimal logistic network. With the obtained *logistic network*, we proposed three meta-heuristic algorithms (SGA-2, TGA-2, and NN-GA-Tabu) to solve the second sub-problem – selecting vendors and transportation modes.

Numerical experiments indicate that the NN-GA-Tabu outperforms the other four algorithms. The merits of the NN-GA-Tabu lead to three implications. First, it advocates the use of a *problem-decomposition paradigm* in solving a comprehensive space search problem, if a technically sound heuristic rule can be found. Second, it advocates the development of an *efficient* (could be *rough*) performance evaluator to speed up the justification of a solution. Third, it advocates the use of *GA-Tabu search mechanism* to find a near-optimum solution. Thus, the proposed NN-GA-Tabu may also be a good solution architecture for solving a comprehensive space search problem. Other applications of the NN-GA-Tabu solution architecture are being considered in future research.

Acknowledgements

This research was partially support by a research grant, under contract NSC94-2623-7-009-004, Taiwan.

References

Candas, M. F., & Kutanoglu, E. (2007). Benefits of considering inventory in service parts logistics network design problems with time-based service constraints. *IIE Transactions, 39*, 159–176.

Chiou, C. W., & Wu, M. C. (2009). A GA-Tabu algorithm for scheduling in-line steppers in low-yield scenarios. Expert Systems with Applications.

Diaz, A., & Fu, M. C. (1997). Models for multi-echelon repairable item inventory systems with limited repair capacity. *European Journal of Operational Research, 97*, 480–492.

Fausett, L. (1994). *Fundamental of neural networks: Architectures, algorithms, and applications*. Prentice-Hall.

Feeney, G. J., & Sherbrooke, C. C. (1966). The $(s-1, s)$ inventory policy under compound Poisson demand. *Management Science, 12*(5), 391–411.

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

Glover, F., & Laguna, M. (1997). *Tabu search*. Boston: Kluwer.

Goldberg, D. E. (1989). *Genetic algorithms in search. Optimization and machine learning*. MA: Addison-Wesley.

Graves, S. C. (1985). A multi-echelon inventory model for a repairable item with one-for-one replenishment. *Management Science, 31*(10), 1247–1256.

Guide, V. D. R., Jr., & Srivastava, R. (1997). Reparable inventory theory: Models and applications. *European Journal of Operational Research, 102*, 1–20.

Jeet, V., Kutanoglu, E., & Partani, A. (2009). Logistics network design with inventory stocking for low-demand parts: Modeling and optimization. *IIE Transactions, 41*(5), 389–407.

Kennedy, W. J., Patterson, J. W., & Fredendall, L. D. (2002). An overview of recent literature on spare parts inventories. *International Journal of Production Economics, 76*, 201–215.

Kim, J. S., Shin, K. C., & Park, S. K. (2000). An optimal algorithm for repairable-item inventory system with depot spares. *Journal of the Operational Research Society, 51*, 350–357.

Kutanoglu, E., & Lohiya, D. (2008). Integrated inventory and transportation mode selection: A service parts logistics system. *Transportation Research Part E, 44*, 665–683.

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs* (3rd ed.). Berlin, Heidelberg, New York: Springer.

Muckstadt, J. A. (1973). A model for a multi-item, multi-echelon, multi-indenture inventory system. *Management Science, 20*(4), 472–481.

Perlman, Y., Mehrez, A., & Kaspi, M. (2001). Setting expediting repair policy in a multi-echelon repairable-item inventory system with limited repair capacity. *Journal of the Operational Research Society, 52*, 198–209.

Rappold, J. A., & van Roo, B. D. (2009). Designing multi-echelon service parts networks with finite repair capacity. *European Journal of Operational Research, 199*(3), 781–792.

Sherbrooke, C. C. (1968). METRIC: A multi-echelon technique for recoverable item control. *Operational Research, 16*, 122–141.

Sherbrooke, C. C. (1986). VARI-METRIC: Improved approximation for multi-indenture, multi-echelon availability models. *Operations Research, 34*(2), 311–319.

Sleptchenko, A., van der Heijden, M. C., & van Harten, A. (2003). Trade-off between inventory and repair capacity in spare part networks. *Journal of the Operational Research Society, 54*, 263–272.

Tsai, J. T., Liu, T. K., & Chou, J. H. (2004). Hybrid Taguchi-genetic algorithm for global numerical optimization. *IEEE Transactions on Evolutionary Computation, 8*(4), 365–377.

Wu, M. C., & Hsu, Y. K. (2008). Design of BOM configuration for reducing spare parts logistic costs. *Expert Systems with Applications, 34*(4), 2417–2423.