Int J Adv Manuf Technol (2006) 29: 803–813
DOI 10.1007/s00170-005-2556-6

ORIGINAL ARTICLE

Wei-Chang Yeh

An efficient memetic algorithm for the multi-stage supply chain network problem

Received: 23 August 2004 / Accepted: 8 January 2005 / Published online: 23 November 2005
© Springer-Verlag London Limited 2005

Abstract A supply chain is dynamic and involves the constant flow of information, production, services, and funds from suppliers to customers between different stages. In this paper, a memetic algorithm (MA, a hybrid genetic algorithm) is developed to find the strategy that can give the lowest cost of the physical distribution flow. The proposed MA is combined with the genetic algorithm (GA), a multi-greedy heuristic method (GH), three local search methods (LSMs): the pairwise exchange procedure (XP), the insert procedure (IP), and the remove procedure (RP), the Fibonacci number procedure, and the linear programming technique (LP) to improve the tradition genetic algorithm (GA). Preliminary computational experiments demonstrate the efficiency and performance of the proposed MA.

Keywords Fibonacci number · Genetic algorithm · Greedy heuristic algorithm · Linear programming technique · Local search methods · Memetic algorithm · Network · Supply chain

1 Introduction

In recent years, many developments in logistics were connected to the need for information in an efficient supply chain flow. In practical applications, a supply chain is often represented as a network (SCN), comprises nodes represent facilities (suppliers, plants, distribution centers and customers), and arcs connect nodes along with the production flow. A MSCN is a sequence of multiple SCN stages. The flow can be transferred only between two consecutive stages. The MSCN can achieve great success in fulfilling customer demands in the best possible way [1–5]. The appropriate MSCN design depends on both the customer's needs and the roles of the stages involved in filling those needs. The MSCN problem proposed here is to find a network strategy that involves the choice of facilities (plants and distribution centers) to be opened and the distribution network design must satisfy the demand with minimum cost. It is based on the three-sequence of stages proposed by Yu [6]. This problem is a NP-hard problem combining the multiple-choice knapsack problem with the capacitated location-allocation problem [7].

Other researches have presented approaches to solve the proposed MSCN problem [2, 8, 9]. Most of their algorithms were based on an incorrect mathematical model that violated the flow conservative law [2, 8, 9]. Otherwise, the published algorithms were very simple and could not find good quality approximation solutions [1]. The need for an efficient algorithm for the MSCN problem thus arises.

In the past decade, we have seen an increasing interest in biologically motivated approaches for finding optimal or good quality solutions to larger problems, including neural networks (NNs), genetic algorithms (GAs), Tabu Search (TS), and simulated annealing methods (SAs) [10–22]. Among these methods, GAs have recently been investigated with a high degree of success and shown to be effective at exploring a large and complex space in an adaptive way, guided by the equivalent biological evolution mechanisms of reproduction, crossover, and mutation [10–21].

GAs are a stochastic search method for optimization problems based on the mechanism of natural selection and genetics, which is the popular notion of the survival-of-the-fittest tenet of Darwinian evolution. Recently, GA has been applied to harder combinatorial optimization problems because it has better characteristics, e.g., there is less effect on calculation when the system becomes more complex or larger. However, the weakness of GA for local searches is well acknowledged [15–21].

Moscato and Morman [15] introduced the term "memetic algorithm" (MA) to describe the genetic algorithms in which a local search plays a significant part. In MAs, a local optimizer is applied to each offspring before it is inserted into the population in order to push it to climb the local optimum [15–21]. With the hybrid method [15–21], GAs are used to perform global exploration within a population, while local optimizers are used to perform local exploitation around chromosomes. Since the properties of GAs and conventional local optimizers are com-

W.-C. Yeh
Department of Industrial Engineering and Engineering Management,
National Tsing Hua University,
P.O. Box 24-60, Hsinchu, Taiwan 300, R.O.C.
E-mail: yeh@ieee.org

plementary, MAs are often better than either method operating alone [15–21].

The purpose of this paper is to present an efficient memetic algorithm (MA) that combines GA, hybrid local search method (LSM), multi-greedy heuristic method (GH), Fibonacci number, and linear programming technique (LP) to solve the MSCN problem proposed first by [2] and revised in [1]. To indicate the performance and efficiency of the proposed MA, the traditional GA, and a heuristic method were also developed and compared to the proposed MA. Our results compare favorably with the GA and GH proposed in [1].

The paper is organized as follows. Section 2 describes the acronyms, notations, and assumptions required. Section 3 contains the discussions about the mixed 0-1 integer programming model (MIP) for the MSCN problem and some important properties in MSCN. In Sect. 4, the GA part in the proposed MA is given. The remaining part of the proposed MA includes the multiple-greedy-heuristic algorithm, the hybrid local search methods, the Fibonacci number procedure, and the linear programming technique presented in Sect. 5. Computational experiments among the proposed MA, traditional GA, and 5 GA based techniques are provided and compared in Sect. 6. Concluding remarks and further research are given in Sect. 7.

2 Acronym, nomenclature, notation and assumptions

Acronym:

MSCN: Multi-stage supply chain network
GA: The genetic algorithm
MA: The memetic algorithm
GH: The multi-greedy heuristic method
LP: The linear programming technique
XP: The pairwise exchange procedure
RP: The remove procedure
IP: The insert procedure
LSM: The hybrid local search method combined with XP, RP, and IP
L_8: The LSM procedure is employed to improve all eight chromosomes obtained from GH in the initial MA population procedure
L_b: The LSM procedure is employed to improve only the best chromosome obtained from GH in the initial MA population procedure
L_r: The LSM procedure is employed to improve the chromosome with the smallest fitness function value among all other chromosomes in the initial MA population procedure
FN: The LSM procedure is employed to improve the best chromosome from the current generation while the generation number is a Fibonacci number in the MA selection procedure
No.: Number
Avg.: Average
Gen.: Generation
Con.: Convergent
Div.: Divergent
Max.: Maximal
Rel.: Relative
Run.: Running
Prob: Problem

Notation:

$|\bullet|$: the number of elements in $\bullet$
S: $S = \{s_1, s_2, \ldots, s_{|S|}\}$ is the supplier set of a MSCN
P: $P = \{p_1, p_2, \ldots, p_{|P|}\}$ is the plant set of a MSCN
D: $D = \{d_1, d_2, \ldots, d_{|D|}\}$ is the DC set of a MSCN
C: $C = \{c_1, c_2, \ldots, c_{|C|}\}$ is the customer set of a MSCN
$\bar{\bullet}$: The average value of $\bullet$
$W(\bullet)$: The capacity (demand) of node $\bullet$
x_{ij}: The number of flows transfer from nodes i to j
$u_{ij} = u(i, j)$: The unit cost of transportation from nodes i to j
f_i: The fixed cost to operate facility i
g_i: The ith gene in a chromosome. If $i = 1, 2, \ldots, \pi$, it is represented the binary value of p_i. Otherwise, it is denoted the binary value of $d_{i-\pi}$
X_i: $X_i = [g_1, g_2, \ldots, g_\pi, g_{\pi+1}, g_{\pi+2}, \ldots, g_{\pi+\delta}]$ is the ith chromosome in the current population
$\bullet_{MAX}$, $\bullet_{MIN}$: The maximal and minimal number of open facility (node) $\bullet$, separately
$I(\bullet)$: $\begin{cases} 1, & \text{if production takes place at } \bullet \in P \cup D \\ 0, & \text{otherwise} \end{cases}$

Nomenclature

Stages: the set of all facility with the same characteristic, e.g., all plants
MSCN: A multiple sequence of stages network s.t. the flow can only transfer between two consecutive stages
MSCN problem: A problem to choose the facilities (plants and DCs) to be opened and designed the network to satisfy the customer demand with minimal cost including the transportation and fixed cost [1, 2]
Flow: The information, production, service, or funds, etc.
Opened (unopened) facility: A supplier, plant or DC has (not) chosen to send production flows, or a customer has (not) chosen to receive production flows.

3 A mixed 0-1 integer programming formulation and preliminaries

The MSCN problem is a NP-hard problem combined of multiple-choice knapsack problems together with capacitated location-allocation problems [7]. The mathematical programming formulation is a natural way to attack a NP-hard problem, although it is not an efficient solution procedure. The mathematical programming for the MSCN problem is first proposed in [2] and revised in [1] by emerging the flow conservative law. The number and capacities (or demands) of suppliers, plants, DCs and customers, the unit transportation cost between suppliers and plants, plants

and DCs, and DCs and customers, as well as the fixed cost for operating plants and DCs are all known in advance. The MSCN problem is formulated in [1] using the flowing mixed 0-1 integer programming model as follows:

Model MSCN

$$\textbf{\textit{Minimize}} \sum_{\substack{i \in S \\ j \in P}}^{x_{ij}} \cdot u_{ij} + \sum_{\substack{i \in P \\ j \in D}}^{x_{ij}} \cdot u_{ij} + \sum_{\substack{i \in D \\ j \in C}}^{x_{ij}} \cdot u_{ij} + \sum_{i \in P \cup D}^{I(i)} \cdot f(i) \quad (1)$$

$$\textbf{\textit{Subject to}} \sum_{j \in P}^{x_{ij}} = W(i), \text{ for all } i \in S \quad (2)$$

$$\sum_{j \in D}^{x_{ij}} \leq I(i) \cdot W(i), \text{ for all } i \in P \quad (3)$$

$$\sum_{j \in C}^{x_{ij}} \leq I(i) \cdot W(i), \text{ for all } i \in D \quad (4)$$

$$\sum_{i \in D}^{x_{ij}} \leq W(i), \text{ for all } j \in C \quad (5)$$

$$\sum_{i \in S}^{x_{ij}} = \sum_{k \in D}^{x_{jk}}, \text{ for all } j \in P \quad (6)$$

$$\sum_{i \in P}^{x_{ij}} = \sum_{k \in C}^{x_{jk}}, \text{ for all } j \in D \quad (7)$$

$$\sum_{p \in P}^{I(p)} \leq P_{\text{MAX}} \quad (8)$$

$$\sum_{d \in D}^{I(d)} \leq D_{\text{MAX}} \quad (9)$$

$$\textbf{\textit{where}}\ x_{ij} \geq 0, \text{ for all } i \in S \text{ and } j \in P \quad (10)$$

$$x_{ij} \geq 0, \text{ for all } i \in P \text{ and } j \in D \quad (11)$$

$$x_{ij} \geq 0, \text{ for all } i \in D \text{ and } j \in C \quad (12)$$

$$I(i) = \begin{cases} 1, \text{ if production takes place at } i \in P \cup D \\ 0, \text{ otherwise} \end{cases}. \quad (13)$$

The objective function minimizes the total cost for setting up and operating the network. In the above model, Eqs. 2–4 are the capacity (or demands) constraints for the suppliers, plants, DCs and customer, separately. The constraints in Eq. 5 state that the total amount shipped to customers must cover the demand. Eqs. 6–7 enforce the total input flows for any plant/DC equal to its own output flow. Eqs. 8–9 are the maximal number constraints for plants and DCs, respectively. Both P_{MAX} and D_{MAX} (the maximum number of opened plants and DCs, respectively) can be given by decision makers [1, 2]. However, the ideal of both P_{MAX} and D_{MAX} used in [1] are adapted according the following two properties in this study.

Property 1: If $W(p_1) \leq W(p_2) \leq \ldots \leq W(p_{|P|})$,

$$\text{then } \sum_{i=1}^{P_{\text{MAX}}-1} W(p_i) \leq \sum_{\forall s \in S} W(s) \leq \sum_{i=1}^{P_{\text{MAX}}} W(p_i). \quad (14)$$

Property 2: If $W(d_1) \leq W(d_2) \leq \ldots \leq W(d_{|D|})$,

$$\text{then } \sum_{i=1}^{D_{\text{MAX}}-1} W(d_i) \leq \sum_{\forall s \in S} W(s) \leq \sum_{i=1}^{D_{\text{MAX}}} W(d_i). \quad (15)$$

To assure that the demands of all customers (called the weight-bound here) are satisfied, the following important property is needed in discussing any MSCN problem.

Property 3: The sum of the total capacities of suppliers, opened plants, and opened DCs in a MSCN problem must all be greater than or equal to the demand of customers, i.e.,

$$\sum_{k=1}^{|S|} W(s_k) \leq Min \left\{ \sum_{k=1}^{|C|} W(c_k), \sum_{k=1}^{|P|} I(p_k) \cdot W(p_k), \sum_{k=1}^{|D|} I(d_k) \cdot W(d_k) \right\}. \quad (16)$$

The following theorem is trivial and forms the basis of the proposed MA. Form this theorem, the proposed MA focus only on how to code the chromosomes in binary integer vectors, then find the corresponding fitness function values using any LP technique.

Theorem 1. *If the values of all binary integer variables $I(i)$ in model MSCN are known, then model MSCN is a linear programming model.*

4 The proposed GA

To overcome the weakness of GA for local searches, an efficient MA is proposed here by combining GA, GH, LSM, and the Fibonacci number to solve the MSCN problem. To show the power of the proposed MA, most of the GA part, e.g., the way of using the binary vector to express chromosomes, the selection operator, the mutation operator, and the crossover operator, are all implemented using the simplest ways as shown in the traditional GAs. The significant differences between the proposed MA and GA are only as follows:

1. The initial populations (see Sect. 4.1).
2. The stopping criterion (see Sect. 4.7).
3. LSM can increase the solution quality very well under the cost of extra execution time. Hence, only the best chromosome created in the generation in which the generation number is a Fibonacci-number or the best offspring is better than the best parents. The best offspring is then improved using

the proposed LSM to save time and produce a better solution (see Sect. 5.3).

The details are discussed in the following subsections.

4.1 The initial populations

The initial population can be created in either a random way or a well-adapted method. Liepins et al. [10] suggested that the use of a well-adapted population provides little advantage despite fast convergence. Therefore, the initial populations are generated in the following three different ways:

1. Randomly.
2. Eight solutions obtained from GH (see Sect. 5.1).
3. Eight solutions obtained by implementing LSM (see Sect. 5.3) into each solution obtained from GH.

4.2 Chromosome representation

As the initialization process of GA, a binary variable vector is encoded as a finite-length string called a chromosome. Chromosome representation is critical to the success of the GA. Here, the permutation encoding is used because the order of items can be most naturally modeled in this way. In permutation encoding, every chromosome is described by a feasible binary variable vector:

$$\begin{aligned} I &= [I(P)|I(D)] \\ &= [I(p_1), I(p_2), \ldots, I(p_{|P|})|I(d_1), I(d_2), \ldots, I(d_{|D|})] \end{aligned} \tag{17}$$

where each gene $I(\bullet)$ is a binary number defined as follows:

$$I(\bullet) = \begin{cases} 1, & \text{if production takes place at } \bullet \in P \cup D \\ 0, & \text{otherwise} \end{cases} . \tag{18}$$

4.3 Fitness function

The fitness function plays an important role in deciding the offspring in the next generation [10]. Chromosomes which are selected to form new solutions (offspring) are according to their fitness function value – the more suitable they are the more chances they have to reproduce. In this study, the fitness function value for each chromosome equals to the objective function list in Eq. 1. Once each gene $I(\bullet)$ is determined, we can calculate their fitness function using the MIP model list in Sect. 3 by any LPs.

4.4 Selection

The selection process discussed here is a mixed method of the elite and the random selection. S eve nty% of the population is selected from the better chromosomes of the parents and offspring by the elite method to prevent losing the best found solution and to rapidly increase the performance of the GA. The random method is based on all of the chromosomes having a chance to be selected. It is used here to select the other 30% population from the unselected chromosomes.

4.5 Crossover

Crossover and mutation are the most important parts of GA. The crossover plays an important role in exchanging information among chromosomes. It leads to an effective combination of partial solutions in other chromosomes and speeds up the search procedure early in the generation. The single-point crossover is developed here to produce two offspring for each pair of parents. To increase the offspring variety and reduce the number of chromosome duplications, a slight revision is made to the original single-point using the following procedure:

Algorithm: The proposed crossover operator.
Input: Two randomly chosen parent chromosomes.
Output: Two offspring.
STEP 0. Randomly choose a single point (an integer number,) say r, among $[1, |P| + |D|)$.
STEP 1. If $r \leq |P|$, then let $\alpha = 1$. Otherwise, let $\alpha = |P| + 1$.
STEP 2. Generate the offspring 1 by coping $g_\alpha, g_{\alpha+1}, \ldots, g_r$ from parent 1, and the remaining genes from parent 2.
STEP 3. Generate the offspring 1 by coping $g_\alpha, g_{\alpha+1}, \ldots, g_r$ from parent 2, and the remaining genes from parent 1.

The crossover is applied with a probability of 0.7 per chromosome in this study. Table 1 shows examples of the proposed crossover operator.

4.6 Mutation

To prevent all solutions in the population from falling into a local optimum of solved problems; mutation takes place after a crossover is performed. The swapping mutation is applied here by choosing one gene within the selected chromosome randomly, and changing its value from 0 to 1 or 1 to 0.

Algorithm: The proposed mutation operator.
Input: One randomly chosen parent chromosome.
Output: One offspring.
STEP 0. Generate one offspring by copying all genes from the chosen parent chromosome.
STEP 1. Randomly choose a single point (an integer number,) say r, among $[1, |P| + |D|$p.
STEP 2. Change the value of g_r in the offspring from 0 to 1 or 1 to 0.

The mutation is applied with the relatively high probability of 0.3 per chromosome. Table 2 shows the swapping mutation with one 3-plant and 4-DC chromosome.

4.7 Stopping criterion

The most frequently used stopping criterion for genetic algorithms is to specify a maximum number of generations. From the experiments, the proposed MA is converged very rapidly. Hence, the stopping criterion is that all of the fitness function values for the selected chromosomes are equal or the best fitness function

Table 1. An example for the proposed crossover

Parent 1	[1, 0, 1 | 1, 1, 0, 0]	
Parent 2	[1, 1, 0 | 0, 1, 1, 0]	
Assume the cut-point is 5.		
Offspring 1	[1, 1, 0 | 1, 1, 0, 0]	Copy $g_{(\|P\|+1)}, g_{(\|P\|+2)}, \ldots, g_6$ from parent 1, and the rest genes from parent 2.
Offspring 2	[1, 0, 1 | 0, 1, 1, 0]	Copy $g_{(\|P\|+1)}, g_{(\|P\|+2}, \ldots, g_6$ from parent 2, and the rest genes from parent 1.

Table 2. An example for the proposed mutation operator

Parent	[1, 0, 1 | 1, 1, 0, 1]	Randomly choose one gene, say g_4 (see the number inside "□").
Offspring	[1, 1, 1 | 0, 1, 0, 1]	Change g_4 from 1 to 0 (see the number inside "□").

value has no further improvements after 100 consecutive generations, then stop.

5 The proposed memetic algorithm

The remaining part of MA, i.e., GH, FN, and LSM are discussed in this section.

5.1 The proposed GH

To speed up the convergent of the proposed MA, the first eight chromosomes are created using a simple multi-greedy heuristic method (GH) in the initial population procedure. All of these eight chromosomes are then improved using a HLSM that is discussed in Sect. 5.3. GH is simply based on the eight rules. These rules can find the exact solution in some special situations.

Rule 1: The path with the smallest unit transportation cost is originally defined in [1] as follows:

$$\upsilon_t = Min\left\{u(s_h, p_i) + u(p_i, d_j) + u(d_j, c_k) + \frac{f_{t-1}(p_j) + f_{t-1}(d_k)}{Min\{w_{t-1}(s_h), w_{t-1}(p_i), w_{t-1}(d_j), w_{t-1}(c_k)\}}\right\},$$

where $Min\{w_{t-1}(s_h), w_{t-1}(p_i), w_{t-1}(d_j), w_{t-1}(c_k)\} \neq 0$. (19)

Rule 2: This rule is similar to *Rule* 1 except customers are not considered, i.e.,

$$\upsilon_t = Min\left\{u(s_h, p_i) + u(p_i, d_j) + \frac{f_{t-1}(p_j) + f_{t-1}(d_k)}{Min\{w_{t-1}(s_h), w_{t-1}(p_i), w_{t-1}(d_j)\}}\right\},$$

where $Min\{w_{t-1}(s_h), w_{t-1}(p_i), w_{t-1}(d_j)\} \neq 0$. (20)

Rule 3: The rule is similar to *Rule* 1 except fixed costs are not considered, i.e.,

$$\upsilon_t = Min\left\{u(s_h, p_i) + u(p_i, d_j) + u(d_j, c_k)\right\}. \quad (21)$$

Rule 4: This rule is similar to *Rule* 3 except customers are not considered, i.e.,

$$\upsilon_t = Min\left\{u(s_h, p_i) + u(p_i, d_j)\right\}. \quad (22)$$

Rule 5: This rule is preceded as follows:

1. Find the smallest unit transportation cost from suppliers to plants.
2. Find the smallest unit transportation cost from chosen plants to DCs.
3. Find the smallest unit transportation cost from chosen DCs to customers, i.e.,

$$\upsilon_t(s_h, p_i) = Min\left\{u(s_h, p_i) + \frac{f_{t-1}(p_j)}{Min\{w_{t-1}(s_h), w_{t-1}(p_i)\}}\right\},$$

where $Min\{w_{t-1}(s_h), w_{t-1}(p_i)\} \neq 0$. (23)

$$\upsilon_t(p_i, d_j) = Min\left\{u(p_i, d_j) + \frac{f_{t-1}(d_k)}{Min\{w_{t-1}(p_i), w_{t-1}(d_j)\}}\right\},$$

where p_i is belong to $\upsilon_t(s_h, p_i)$ with $Min\left\{w_{t-1}(p_i), w_{t-1}(d_j)\right\} \neq 0$. (24)

$$\upsilon_t = Min\left\{u(d_j, c_k)\right\}, \text{ where } d_j \text{ is belong to } \upsilon_t(p_i, d_j). \quad (25)$$

Rule 6: This rule is similar to *Rule* 5 except the flow between DCs to customers are not considered:

$$\upsilon_t(s_h, p_i) = Min\left\{u(s_h, p_i) + \frac{f_{t-1}(p_j)}{Min\{w_{t-1}(s_h), w_{t-1}(p_i)\}}\right\},$$

where $Min\{w_{t-1}(s_h), w_{t-1}(p_i)\} \neq 0$. (26)

$$\upsilon_t = Min\left\{u(p_i, d_j) + \frac{f_{t-1}(d_k)}{Min\{w_{t-1}(p_i), w_{t-1}(d_j)\}}\right\},$$

where p_i is belong to $\upsilon_t(s_h, p_i)$ with $Min\{w_{t-1}(p_i), w_{t-1}(d_j)\} \neq 0$. (27)

Rule 7: Choose the path with the smallest fixed costs, i.e.,

$$\upsilon_t = Min\left\{f_{t-1}(p_i) + f_{t-1}(d_j)\right\}. \quad (28)$$

Rule 8: The solution with the minimal number of facilities may be the optimum.

$$\upsilon_t = Min\left\{Max\{w_{t-1}(s_h)\}, Max\{w_{t-1}(p_i)\}, Max\left\{w_{t-1}(d_j)\right\}, Max\{w_{t-1}(c_k)\}\right\}. \quad (29)$$

After finding υ_t in the tth iteration for each rule, the fixed costs for the plant and DC in the chosen path form according to υ_t are all set to zero. The residual costs are also decreased by the maximal flow, which is the minimal residual capacity among the facilities in the path. The above three steps, i.e., calculate υ_t, set the corresponding fixed costs to zero, and decrease the corresponding residual capacities, are also performed repeatedly until all customers are satisfied, i.e., all residual capacities for customers are zeros. The remaining model MSCN is then solved using LP under the obtained information that $I(\bullet) = 1$ if $\bullet$ is opened otherwise $I(\bullet) = 0$ in the above steps.

5.2 The feasibility check

Each new chromosome is created randomly in the initial population procedure (i.e., except for those created by GH, L_8 and/or L_b). The mutation procedure and/or crossover procedure, must satisfy properties 1–3. Otherwise, the chromosome is modified using the following procedure to assure its feasibility due to the number of open plants violated.

1. If total capacity of opened plants (DCs) is less than the total demand of customers, i.e., $\sum_{p\in P}^{I(p)} \cdot W(p) \le \sum_{s\in S}^{W(s)} \left(\sum_{d\in D}^{I(d)} \cdot W(d) \le \sum_{s\in S}^{W(s)} \right)$, then we randomly open unopened plants (DCs), i.e., let $I(i) = 1$ if $I(i) = 0$ randomly where $i \in P$ (D), until Eq. 14 is satisfied.
2. If the number of opened plants (DCs) is greater than P_{MAX} (D_{MAX}), i.e., $P_{\mathrm{MAX}} \le \sum_{p\in P}^{I(p)} \left(D_{\mathrm{MAX}} \le \sum_{d\in D}^{I(d)} \right)$, then we randomly close opened plants (DCs), i.e., let $I(i) = 0$ if $I(i) = 1$ randomly where $i \in P$ (D), until $\sum_{p\in P}^{I(p)} \le P_{\mathrm{MAX}} \left(\sum_{d\in D}^{I(d)} \le D_{\mathrm{MAX}} \right)$.

5.3 Local search methods (LSMs)

One of the major drawbacks of the GA is that it is convergent too slowly [10–21]. LSMs are implemented to prevent this drawback and guide the search toward unexplored regions in the solution space. LSM can increase the quality of obtained solutions very well under the cost of extra execute time [21]. Hence, LSM only implements to the following three conditions to save time and produce a better solution:

1. Each chromosome is created using GH.
2. The best offspring is created in the Fibonacci number generation.
3. The best offspring is not created in the Fibonacci number generation but with better fitness function value than the best parent.

To easily determine under which condition LSM is employed, if LSM is implemented in the 1st, 2nd, and 3rd condition, it is called L_8, L_b, and L_r, respectively. The proposed LSM has combined three famous local improvement methods: the pairwise exchange procedure (XP), remove procedure (RP), and insert procedure (IP). These three local improvement methods are used very often in the scheduling problem, and they are revised here to improve the initial population obtained in the GA for the MSCN problem.

In XP, the binary values of every pair of genes are exchanged for each plant and DC chromosome. In IP, the status of one selected unopened facility is changed to open, i.e., replaced with 1 if its value in the current gene is 0. RP is the reverse procedure of IP, i.e., replace with 0 if the value of the currently gene is 1.

LSM is performed in XP-RP-IP series to the specific chromosomes satisfying any one of the above three conditions. Whole procedures are performed in respect to the genes regarding plants first. The genes regarding DCs are then discussed. If there is no improvement in the objective function value after the same procedure is executed a second time, LSM is stopped. For example, assume that the last procedure for updating the objective function value is XP. If the following RP, IP, and the next XP cannot improve the objective function value, then the proposed LSM must stop.

After executing any of the above procedure to the current (feasible) chromosome, the new obtained chromosome needs to have its feasibility checked. If it is unfeasible, then the two procedures listed in Sect. 5.2 are implemented to transfer this new unfeasible chromosome into a feasible chromosome. Next, the corresponding objection function value of the new feasible chromosome is found using LP. Replace the current fitness function value and the current chromosome with the new fitness function value and new chromosome if the solution is improved.

As shown in Table 3, the total none-zero numbers in each new chromosome obtained from XP, IP, and RP are no change, increasing by 1, and decreasing by 1, respectively. Hence, Eq. 14 must verify for XP and RP, and Eqs. 15 and 16 must verify for IP and RP.

5.4 The linear programming technique

Linear programming is the core model of constrained optimization. It also plays an important role in the proposed MA. If each

Table 3. Examples for the proposed XP, IP, and RP

XP	Parent	[1, 0, 1 | 1, 1, 0, 1]	Randomly choose two genes in which their values are different, say g_4 and g_6 (see the number inside "□").
	Offspring	[1, 1, 1 | 0, 1, 0, 1]	Exchange their value (see the number inside "□").
IP	Parent	[1, 0, 1 | 1, 1, 0, 1]	Randomly choose one unopened gene, say g_2 (see the number inside "□").
	Offspring	[1, 1, 1 | 0, 1, 0, 1]	Change the value of g_2 from 0 to 1 (see the number inside "□").
RP	Parent	[1, 0, 1 | 1, 1, 0, 1]	Randomly choose one opened gene, say g_4 (see the number inside "□").
	Offspring	[1, 1, 1 | 0, 1, 0, 1]	Change the value of the g_4 from 0 to 1 (see the number inside "□").

value of $I(\bullet)$ of the MIP model for MSCN is known, the residual model is just a simple linear programming model. Therefore, each fitness function of every new obtained chromosome is calculated by the LP, e.g., simplex method or interior method. The standard method for solving LP is currently still Dantzig's simplex method [22, 23]. Two recent polynomial algorithms have been widely publicized: the ellipsoid algorithm and the algorithm of Karmarkar [23]. The ellipsoid algorithm first established that linear programs are solvable in polynomial time but has given poor performance in practice. The claims for Karmarkar's algorithm for large problems are encouraging.

6 Computational result

The numerical tests will be described in this section. To show the efficiency (running time) and the quality of the proposed MA, seven methods (see Table 4) including the traditional GA approach were employed and tested in randomly generated test problems.

Table 4. The seven methods

Method no.	remarks
1	The proposed MA, i.e., GA combining the GH, L_8, L_r, FN, and LP;
2	The proposed MA but replacing L_8 with L_b which is implementing only once to the best solution obtained from GH to reduced the running time;
3	The proposed MA but without implementing L_8;
4	The proposed MA but without implementing GH and L_8;
5	The proposed MA but without implementing GH, L_8 and L_r;
6	The proposed MA but without implementing GH, L_8 and FN;
7	The traditional GA together with LP, i.e., without implementing GH, L_8, L_b, L_r, and FN;

Table 5. A summary of the difference of seven methods

Method no.	GH	L_8	L_b	L_r	FN	LP	GA
1	√	√		√	√	√	√
2	√		√	√	√	√	√
3	√			√	√	√	√
4				√	√	√	√
5					√	√	√
6				√		√	√
7						√	√

A summary of the differences among the above methods are listed in Table 5.

All of the seven methods were implemented in C^{++} and run on a Pentium 4-2.6G notebook personal computer with he same cross-rate 0.7 and mutation-rate 0.3. Each method was tested using four different sizes adapted from [1] as given in Table 6. Each test group contained 100 different problems. Therefore, there were 400 test problems in this experiment. The values of D_{MIN}, D_{MAX}, P_{MIN}, and D_{MAX} for each problem were given under the condition that properties 1–3 were satisfied.

The unit costs of transportation, the fixed costs on plants and DCs and the facility capacities (demands) in every data set were randomly generated in a uniform distribution s.t. Property 3 is held. Otherwise, the corresponding data are regenerated. The details of the range of created data are as shown in Table 7.

Each different method was executed two times on each test problem. One with a population size equal only to 16, and the other four times of the number of DCs and plants, e.g., the populations for test group 4553 was 40. The experimental results for a population size equal to 16 are listed in Table 8. The results for the other population sizes are list in Table 9. From Tables 8 and 9, we have the following observations:

1. Comparing methods 1 to 2, L_8 plays a more important role than L_b in obtaining exact solutions.
2. Comparing methods 1 to 3, L_8 plays an important role in obtaining exact solutions.
3. Comparing methods 2 to 3, L_b plays an important role in obtaining exact solutions.
4. Comparing methods 3 to 4, L_r plays a more important role than $GH + L_r$ in obtaining exact solutions.
5. Comparing methods 4, 5 and 6, 7, L_r can find more exact solutions than the traditional GA in the initial population procedure.
6. Comparing methods 4, 6 and 5, 7, FP can find more exact solutions than the traditional GA after the initial population procedure.
7. The effectiveness of GH decreases dramatically when $(|P| + |D|)$ is increasing. However, L_8, L_b, L_r and FN are not sensible to the problem size. Moreover, GH mixed with L_8 and/or L_b obtained more exact solutions.
8. The more exact solutions obtained in the initial population procedure, the less effective L_r and FN were. Also, FN helped to obtain more exact solutions if no other improvement operations were implemented in the initial population procedure of the traditional GA.

Table 6. The facility number, $\sum_{i=1}^{|C|} Cap(c_i)$, $\bar{P}_{\mathrm{MIN}}$, $\bar{P}_{\mathrm{MAX}}$, $\bar{D}_{\mathrm{MIN}}$ and $\bar{D}_{\mathrm{MAX}}$ for the test groups

Group name	$\lvert C\rvert$	$\lvert D\rvert$	$\lvert P\rvert$	$\lvert S\rvert$	$\sum_{i=1}^{\lvert C\rvert} Cap(c_i)/\lvert C\rvert$	$\bar{D}_{\mathrm{MIN}}$	$\bar{D}_{\mathrm{MAX}}$	$\bar{P}_{\mathrm{MIN}}$	$\bar{P}_{\mathrm{MAX}}$
4553	4	5	5	3	799.43	1.56	2.95	1.51	3.09
21101010	21	10	10	10	4205.26	1.80	4.71	1.78	4.79
50121520	50	12	15	20	9980.58	1.93	5.51	1.88	6.04
1008610	100	8	6	10	19918.74	2.00	4.78	2.10	4.23
Average	43.75	8.75	9	10.75	8726.003	1.8225	4.4875	1.8175	4.5375

Table 7. The range of created data

Item	Range
The capacity of each customer	U[100, 300)
The capacity of each supplier, plant, and DC	100+U[0, $\sum_{c \in C} Cap(c)$]
the unit cost of transportation	U[3, 10)
The fixed cost	U[900,2000)

Overall, the proposed MA (i.e., method 1) implemented together with GH, L_8, L_r, FN, LP and GA is the best approach among all other GA based algorithms. MA obtained more exact solutions during the initial population procedure (see Table 10). It also has fewer absolute errors and generations on average for the test problems solved from convergence. Its total running time was greater than that for the other methods but was less than 1 minute for each test problem on average.

The major improvement over the traditional GA by the proposed MA is the initial population procedure. The initial population procedure therefore receives more discussion. Table 10 demonstrates the overall average percentage for obtaining exact solutions. Table 11 shows that which operation, i.e. GH, L_8, L_b and/or L_r, is more significant in obtaining exact solutions in the initial population procedure in detail. In the initial population procedure, GH obtained nearly 36 optimums among 100 problems when the problem size was smaller, 14.25% exact solutions on average. L_8 also worked very well and its average percentage for obtaining exact solutions was more than 57% of the remaining problems after executing GH. When only implementing L_b to the best solution obtained from GH, 28.43% exact solutions were obtained on average. From Table 11, only GH combined with L_8 and L_r was the best one in average for obtaining exact solutions in the initial population procedure. Another special factor is that the average percentages for obtaining exact solutions decreased when the total number of DCs and plants was increased. The reason is that only the variables corresponding to DCs and plants were integers requiring solutions.

After determining $GH + L_8 + L_r$ as the best on average in obtaining the exact solutions in the initial population procedure, eight rules forming GH are discussed next to see which rule was really useful. From Table 12, rule 1 was the best one, rule 2 was the worst one in GH but the best one in L_8. Moreover, rules 1–6 played almost equally important roles after executing GH and L_8. Table 13 shows the same result and that L_8 significantly improved the solution quality.

Not every test problem could be solved to obtain the exact solution. It is necessary to discuss the quality of the obtained approximation solutions. Table 14 shows that among seven methods, the proposed MA (method 1) produced the minimal average relative error between the approximation solution and the optimum. The average relative error is defined as the sum

Table 8. The results for a population size equal to 16

Group name	Method no.	GH	L_8	L_b	L_r	rnd	FN	GA	Con. prob. no.	Div. prob. no.	Con. to opt. Avg. gen. no.	Con. to opt. Max. gen. no.	Con. to non-opt. Avg. gen. no.	Con. to non-opt. Max. gen. no.	Avg. rel. error	Avg. run. time
4453	1	36	49		1		4		10	0	4.250	8	2.300	4	2.345%	2.154
	2	36		30	2		16		18	0	2.471	5	7.408	20	1.085%	3.877
	3	36			4		37		23	0	2.162	5	5.913	9	2.795%	0.973
	4				50		36		14	0	2.778	8	8.000	16	2.966%	0.608
	5					3	76		21	0	2.711	8	8.238	25	2.745%	0.758
	6				51			28	21	0	5.286	13	8.905	23	2.735%	0.464
	7					1		59	40	0	6.898	22	9.800	17	3.306%	1.127
21101010	1	4	52		0		10		34	0	2.200	3	7.941	29	0.730%	25.142
	2	4		25	0		16		55	0	3.000	13	7.909	36	1.819%	3.516
	3	4			1		41		55	0	2.902	13	9.800	24	1.509%	11.353
	4				17		35		48	0	3.200	13	9.458	23	1.668%	4.840
	5					0	53		47	0	3.906	34	11.170	26	2.019%	6.612
	6				15			7	78	0	11.857	33	10.692	32	2.900%	1.701
	7					0		13	87	0	12.385	18	16.816	51	3.957%	3.266
50121520	1	1	47		0		10		42	0	2.600	5	11.333	53	0.542%	85.347
	2	1		18	0		15		66	0	2.867	5	9.091	42	1.127%	12.036
	3	1			2		36		61	0	3.778	13	10.279	45	1.087%	41.051
	4				17		26		57	0	3.923	13	10.316	37	1.178%	17.147
	5					0	37		63	0	3.703	13	10.651	30	1.180%	22.473
	6				9			4	87	0	10.250	17	11.402	38	2.260%	3.293
	7					0		4	96	0	20.000	36	20.521	35	2.924%	6.056
1008610	1	16	42		0		7		34	0	2.571	6	4.618	14	0.513%	21.671
	2	16		19	0		17		49	0	2.471	5	7.408	20	1.085%	3.877
	3	16			2		36		46	0	3.167	21	9.109	27	0.998%	13.517
	4				22		33		45	0	3.030	8	8.311	24	0.695%	5.567
	5					0	55		45	0	3.473	18	9.222	21	1.360%	6.992
	6				22			12	66	0	9.583	17	10.742	30	1.921%	2.954
	7					1		28	71	0	10.786	44	14.070	36	2.437%	4.539

Table 9. The results for a population size was $4(|P|+|D|)$

Group name	Method no.	GH	L_8	L_b	L_r	rnd	FN	GA	Con. prob. no.	Div. prob. no.	Con. to opt. Avg. gen. no.	Con. to opt. Max. gen. no.	Con. to non-opt. Avg. gen. no.	Con. to non-opt. Max. gen. no.	Avg. rel. error	Avg. run. time
4453	1	36	49		1		6		8	0	3.333	6	4.750	6	1.990%	399.371
	2	36		30	2		23		9	0	3.304	8	6.444	11	1.005%	50.645
	3	36			8		45		11	0	3.022	15	8.727	18	1.936%	122.388
	4				59		26		15	0	2.769	7	10.067	40	2.953%	84.628
	5						82		11	0	2.549	10	10.000	12	1.790%	101.571
	6				61	31		31	8	0	4.645	12	9.250	16	1.676%	71.615
	7					84		84	11	0	4.845	12	9.909	14	1.320%	108.869
21101010	1	4	51		0		14		31	0	3.071	7	15.871	84	0.450%	4271.838
	2	4		25	0		31		40	0	4.129	21	15.575	46	0.017%	890.331
	3	4			1		65		30	0	5.016	21	19.065	89	0.686%	1031.294
	4				17		51		31	1	3.588	15	19.774	60	0.892%	962.867
	5						65		35	0	4.862	43	18.314	64	0.627%	1196.838
	6				21	39		39	40	0	10.051	32	22.375	82	0.976%	978.887
	7					65		65	35	0	11.338	31	19.486	54	0.878%	1103.017
50121520	1	1	47		0		15		37	0	6.800	46	20.703	72	0.310%	12931.818
	2	1		18	0		26		53	2	6.769	34	16.698	42	0.539%	3498.423
	3	1			0		57		42	0	8.456	49	22.476	84	0.398%	4107.004
	4				16		37		46	1	5.730	34	24.543	87	0.515%	3831.789
	5						55		44	1	6.727	25	21.705	69	0.506%	4780.627
	6				12	38		38	48	2	14.947	49	23.417	65	0.580%	2990.399
	7					55		55	45	0	16.509	35	26.800	68	0.355%	3785.146
1008610	1	16	42		0		10		32	0	3.800	8	9.469	52	0.220%	3203.879
	2	16		19	0		27		38	0	5.111	28	11.079	41	0.558%	930.076
	3	16			2		46		36	0	4.646	63	12.143	29	0.472%	1006.299
	4				24		40		36	0	3.750	12	11.806	29	0.446%	1042.896
	5						65		35	0	3.446	13	12.229	21	0.286%	1220.268
	6				28	38		38	34	0	7.079	21	15.088	43	0.434%	917.259
	7					56		56	44	0	7.571	14	14.614	33	0.551%	1184.360

Table 10. The average percentage for obtaining exact solutions

Population size	Groups name	1	2	3	4	5	6	7
16	4553	90.00%	84.00%	77.00%	86.00%	79.00%	79.00%	60.00%
16	21101010	66.00%	45.00%	46.00%	52.00%	53.00%	22.00%	13.00%
16	50121520	58.00%	34.00%	39.00%	43.00%	37.00%	13.00%	4.00%
16	1008610	65.00%	52.00%	54.00%	55.00%	55.00%	34.00%	29.00%
	Sub Average	69.75%	53.75%	54.00%	59.00%	56.00%	37.00%	26.50%
40	4553	92.00%	91.00%	89.00%	85.00%	89.00%	92.00%	89.00%
80	21101010	69.00%	60.00%	70.00%	68.00%	65.00%	60.00%	65.00%
108	50121520	63.00%	45.00%	58.00%	53.00%	55.00%	50.00%	55.00%
56	1008610	68.00%	62.00%	64.00%	64.00%	65.00%	66.00%	56.00%
	Sub Average	73.00%	64.50%	70.25%	67.50%	68.50%	67.00%	66.25%
	Average	71.38%	59.13%	62.13%	63.25%	62.25%	52.00%	46.38%

Table 11. The average percentage for obtaining exact solutions from GH, L_8, L_b, and/or $L_r^{\#}$

Groups name	GH	L_8^*	GH + L_8	GH + L_8 + L_r	L_b^*	GH + L_b	GH + L_b + L_r	GH + L_r	L_r
4553	36.00%	76.56%	85.00%	86.00%	46.88%	66.00%	68.00%	42.00%	55.25%
21101010	4.00%	53.13%	55.00%	55.50%	26.04%	29.00%	29.00%	5.00%	17.50%
50121520	1.00%	47.47%	48.00%	48.00%	18.18%	19.00%	19.00%	2.00%	13.50%
1008610	16.00%	50.00%	58.00%	58.00%	22.62%	35.00%	35.00%	18.00%	24.00%
Average	14.25%	56.79%	61.50%	61.88%	28.43%	37.25%	37.75%	16.75%	27.56%

$^{\#}$ with or without executing GH, L_8 and/or L_b.

* the average percentage for obtaining the exact solutions after executing GH.

Table 12. The average percentage for obtaining exact solutions from GH under eight different rules, and/or from L_8

Procedure	Groups name	Rule no. 1	2	3	4	5	6	7	8
GH	4553	26.00%	0.00%	23.00%	2.00%	23.00%	1.00%	0.00%	0.00%
	21101010	3.00%	0.00%	3.00%	0.00%	2.00%	0.00%	0.00%	0.00%
	50121520	0.00%	0.00%	0.00%	0.00%	1.00%	0.00%	0.00%	0.00%
	1008610	9.00%	0.00%	9.00%	0.00%	11.00%	0.00%	0.00%	0.00%
	Average	9.50%	0.00%	8.75%	0.50%	9.25%	0.25%	0.00%	0.00%
L_8^*	4553	45.31%	46.88%	42.19%	45.31%	42.19%	50.00%	37.50%	51.56%
	21101010	18.75%	28.13%	23.96%	29.17%	23.96%	27.08%	7.29%	13.54%
	50121520	18.18%	25.25%	14.14%	25.25%	14.14%	26.26%	4.04%	10.10%
	1008610	10.71%	30.95%	20.24%	28.57%	19.05%	27.38%	16.67%	10.71%
	Average	23.24%	32.80%	25.13%	32.08%	24.83%	32.68%	16.37%	21.48%
$GH + L_8$	4553	55.00%	30.00%	50.00%	31.00%	50.00%	33.00%	24.00%	33.00%
	21101010	21.00%	27.00%	26.00%	28.00%	25.00%	26.00%	7.00%	13.00%
	50121520	18.00%	25.00%	14.00%	25.00%	15.00%	26.00%	4.00%	10.00%
	1008610	18.00%	26.00%	26.00%	24.00%	27.00%	23.00%	14.00%	9.00%
	Average	28.00%	27.00%	29.00%	27.00%	29.25%	27.00%	12.25%	16.25%

* without considering these test problems in which exact solutions were found using GH

Table 13. The average error between exact solutions and obtained solutions from GH under eight different rules, and/or from L_8

Procedure	Groups name	Rule no. 1	2	3	4	5	6	7	8
GH	4553	5.864%	24.459%	6.650%	22.873%	7.782%	24.652%	28.280%	21.214%
	21101010	6.465%	28.606%	6.535%	28.236%	6.465%	29.313%	25.855%	24.942%
	50121520	5.644%	25.099%	5.644%	25.099%	6.324%	25.800%	22.210%	25.165%
	1008610	6.288%	20.819%	6.288%	20.819%	5.451%	21.652%	21.396%	20.529%
	Average	6.065%	24.746%	6.279%	24.257%	6.505%	25.355%	24.435%	22.962%
L_8^*	4553	2.411%	2.182%	2.180%	2.673%	2.426%	2.081%	3.720%	2.424%
	21101010	2.553%	1.712%	2.603%	1.626%	2.446%	1.915%	3.961%	3.861%
	50121520	1.883%	1.088%	1.889%	1.169%	1.903%	1.129%	3.069%	3.858%
	1008610	2.635%	1.337%	1.827%	1.569%	1.817%	1.653%	2.982%	3.227%
	Average	2.371%	1.580%	2.125%	1.759%	2.148%	1.694%	3.433%	3.342%

* without considering these test problems in which exact solutions were found using GH

Table 14. The average relative error between obtained solutions and exact solutions

Population size	Groups name	1	2	3	4	5	6	7
16	4553	2.345%	1.085%	2.795%	2.966%	2.745%	2.735%	3.306%
16	21101010	0.730%	1.819%	1.509%	1.668%	2.019%	2.900%	3.957%
16	50121520	0.542%	1.127%	1.087%	1.178%	1.180%	2.260%	2.924%
16	1008610	0.513%	1.085%	0.998%	0.695%	1.360%	1.921%	2.437%
	Sub Average	1.033%	1.279%	1.597%	1.627%	1.826%	2.454%	3.156%
40	4553	1.990%	1.005%	1.936%	2.953%	1.790%	1.676%	1.320%
80	21101010	0.450%	0.017%	0.686%	0.892%	0.627%	0.976%	0.878%
108	50121520	0.310%	0.539%	0.398%	0.515%	0.506%	0.580%	0.355%
56	1008610	0.220%	0.558%	0.472%	0.446%	0.286%	0.434%	0.551%
	Sub Average	0.743%	0.530%	0.873%	1.202%	0.802%	0.917%	0.776%
	Average	0.888%	0.904%	1.235%	1.414%	1.314%	1.685%	1.966%

of the absolute value for the relative error divided by the total approximation solution number. If the approximation solution number is smaller, then the corresponding average relative error is bigger. From Tables 8, 9, and 10, the proposed MA always obtained a larger number of exact solutions than the others. Hence, the relative errors for groups 4553 obtained from the proposed MA were not the best.

7 Conclusions and further research

The main purpose of this article was to present a more efficient and effective algorithm to find an approximated MSCN solution. The proposed algorithm employs a simple memetic algorithm that combines the multi-greedy methods, local search methods

including XP, RP and IP, Fibonacci number, and LP to solve the MSCN. From the above computational experiments, the proposed MA is very efficient and effective in solving the MSCN problem with high quality solutions, more exact solutions and less relative error in fewer numbers of generations.

MA is a parameter-sensitive technique similar to GAs, SAs, TSs, etc. An interesting research aspect of an extensive parametric study on seeking the relationship among the convergence rate, variable strategy (e.g., a different crossover operation, and different mutation operation, selection and mixed operation), and variable parameter rate, such as the variable crossover rate, variable mutation rate, variable selection rate, and variable population size, is the subject of ongoing research, and results will be reported elsewhere. For future works, we also shall test our algorithm in larger scale problems. Moreover, for each iteration, processing of the population itself can be paralleled by having several processors working on subpopulations of solutions simultaneously. This aspect still merits intensive investigation.

References

1. Yeh W-C (in press) A hybrid heuristic algorithm for the multi-stage supply chain network problem. Int J Adv Manuf Technol 26:675–685
2. Syarif A, Yun YS, Gen M (2002) Study on multi-stage logistic chain network: a spanning tree-based genetic algorithm approach. Comput Ind Eng 43:299–314
3. Chopra S, Meindl P (2003) Supply chain management: strategy, planning and operation, 2nd edn, Prentice-Hall, New York
4. Stadtler H, Kilger C (eds) (2002) Supply chain management and advanced planning. Springer, Berlin Heidelberg New York
5. Poirier CC (1999) Advanced supply chain management: how to build a sustained competitive advantage. Berrett-Kochler, San Francisco, CA
6. Yu H (1997) ILOG in the supply chain. ILOG Technical Report
7. Gen M, Cheng R (1997) Genetic algorithm and engineering design. Wiley, New York
8. Kaufman L, Eede MV, Hansen P (2000) A plant and warehouse location problem. Oper Res Q 2:547–554
9. Lee CY (1993) A cross decomposition algorithm for a multi-product multi-type facility location problem. Comput Oper Res 20:527–540
10. Leipins G, Hilliard M (1989) Genetic algorithm: foundation and applications. Ann Oper Res 21:31–58
11. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA
12. Nachtigall K, Voget S (1996) A genetic algorithm approach to periodic railway synchronization. Comput Oper Res 23:453–463
13. Hoi DY, Dutta D (1996) A genetic algorithm application for sequencing operations in process planning for parallel machining. IIE Trans 28: 55–68
14. Al-Sultan KS, Hussain MF, Nizami JS (1996) A genetic algorithm for the set covering problem. J Oper Res Soc 47:702-709
15. Moscato P, Norman M (1992) A memetic approach for the traveling salesman problem: implementation of a computational ecology for combinatorial optimization on message-passing system. Proceeding of 20th International Conference on Parallel Computing and Transportation Applications
16. Takashi K, Hiroaki K, Masakazu N (1993) A hybrid search for genetic algorithms: combining genetic algorithms, tabu search, and simulated annealing. Proceeding of 5th International Conference on Genetic Algorithms, pp 641
17. Ghoshray S, Yeh KK, Andrian J (1995) Modified genetic algorithms by efficient unification with simulated annealing. In: Pearson DW, Steele NC, Albrecht RF (eds) Artificial Neural Nets Genetic Algorithm. Springer-Verlag, pp 487–490
18. Cotta A, Aldana JF, Nebro AJ, Troya JM (1995) Hybridizing genetic algorithms with branch-and-bound techniques for the resolution of the TSP. In: Pearson DW, Steele NC, Albrecht RF (eds) Artificial Neural Nets and Genetic Algorithm. Springer-Verlag, pp 277–280
19. Hattori M, Naruse M, Shirataki J, Tomikawa T (1996) A study of parameter optimization in mega-genetic algorithm. Proceeding of 20th International Conference on Computers & Industrial Engineering, pp 445–448
20. Yeh W-C (2000) A memetic algorithm for the min k-cut problem. Control Intell Syst 28:47–55
21. Yeh W-C (2000) A memetic algorithm for the n/2/flowshop/$\alpha F + \beta C_{max}$ scheduling problem. Int J Adv Manuf Technol 20(9):464–473
22. Corley HW, Rosenberger J, Yeh W-C, Sung TK (2005) The cosine simplex algorithm. Int J Adv Manuf Technol. Published online 6 April 2005
23. Bazaraa MS, Jarvis JJ (1990) Linear programming and network flows 2nd edn. Wiley, New York