Çankaya University Journal of Science and Engineering
Volume 10 (2013), No 2, 185-201.

CUJ

Using Segment-based Genetic Algorithm with Local Search to Find Approximate Solution for Multi-Stage Supply Chain Network Design Problem

Marjan Kuchaki Rafsanjani* and Sadegh Eskandari*

*Department of Computer Science,
Shahid Bahonar University of Kerman, Kerman, Iran
e-mail: kuchaki@uk.ac.ir, sadegh_esk@yahoo.com.

Abstract: Designing an optimal supply chain network (SCN) is an NP-hard and highly nonlinear problem; therefore, this problem may not be solved efficiently using conventional optimization methods. In this article, we propose a genetic algorithm (GA) approach with segment-based operators combined with a local search technique (SHGA) to solve the multistage-based SCN design problems. To evaluate the performance of the proposed algorithm, we applied SHGA and other competing algorithms to SCNs with different features and different parameters. The results obtained show that the proposed algorithm outperforms the other competing algorithms.

Keywords:. *Supply Chain Network; Genetic Algorithm; Segment-Based Operators; Local Search*.

1. Introduction

Supply chain (SC) management is an active subject in manufacturing research today, thus supply chain network (SCN) design became one of the most important topics of numerical analysis and optimization methods. A supply chain (SC) is a network of facilities and distribution options that performs the functions of procurement of materials, transformation of these materials into intermediate and finished products, and the distribution of these finished products to customers [1]. A multi-stage supply chain network (MSCN) can be considered as a sequence of stages, each consisting of a set of potential facilities located in several regions, where each stage supplies demands of the next stage. Thus we can model such structure by representing facilities and flows of materials respectively using nodes and weighted arcs.The weight in an arc represents the cost caused by connecting two nodes corresponding to two ends of the arc. For an arc, we define the outgoing node as the source and the incoming node as a depot. The multi-stage supply chain network (MSCN) design problem is to find an allocation of sources to depots, minimizing the overall costs and satisfying the demands of depots, according to capacities of sources.

ISSN 1309 – 6788

MSCN design is an NP-hard problem [11], and its complexity increases extremely with increasing the size of the problem (number of stages and number of facilities in each stage). The conventional methods such as local search and hill climbing are inappropriate to solve MSCN design problem, because these methods solve problems in a serial manner and they need a lot of time to find appropriate solutions. To overcome these shortcomings, many genetic algorithm-based heuristic approaches have been developed by the researchers in the last decade [1, 2, 5, 13,14, 15, 16, 17]. The genetic algorithm (GA) is a population based searching algorithm that has attracted the attention of many of the researchers. To solve MSCN design problems using GAs, Syarif, Yun and Gen have developed a spanning tree-based GA based on Prufer numbers [13], Altiparmak have developed a steady-state GA with a segment based encoding procedure [1].

However, there are two weaknesses in applying conventional GA to multistage-based SCs. 1) Conventional GA evaluates each supply chain network using a single evaluation function, but the MSCN design problem consists of several features (such as number of stages, number of facilities at each stage and number of materials produced at each stage) and parameters (such as demand matrixes and transportation cost matrixes) therefore evaluating individuals using a single fitness function will ignore most of the information about these details. 2) When the GA is converging to local optimal solution the performance of GA definitely deteriorates [17], especially for highly nonlinear problem such as the MSCN design problem. In this paper to overcome these two weaknesses we proposed a segment based hybrid genetic algorithm with local search (SHGA) to solve the MSCN design problem.

In section 2, a detailed structure of the MSCN model is suggested. While second section includes instruction of the proposed segment based GA and its operators, the brief description of local search and similarity control mechanism is discussed in section 4. The structure of the proposed SHGA is given in section 5. Moreover, the fifth section gives computational results followed by conclusions in section 6.

2. Multi-Stage Supply Chain Network (MSCN) Design Problem

At this stage, we provide a formulation of the MSCN design problem. As mentioned in the previous section, we can model a supply chain network (SCN) using a directed tree which connects sources to depots using weighted arcs. The overall structure of such a tree is outlined in Fig. 1. This structure is divided into stages and each stage consists of a set of nodes. Each node represents a facility that performs a given task so the nodes in each

stage perform the same class of tasks. A node in the *ith* stage (except first and last stages) can be both a source for the nodes of the $(i+1)th$ stage and a depot for nodes of the $(i-1)th$ stage. The nodes of the first stage can only be sourced and we call them suppliers and the nodes of the last stage can only be depots and we call them customers.This problem is to determine the subsets of sources to be opened and to design the distribution network that will satisfy all capacities and demand requirements for each product imposed by customers at minimum cost.

Some assumptions for implementing MSCN design problems are defined as follows:

i The number of potential facilities of each stage and their tasks is known.
ii The capacities of sources and the demands of customers are known.
iii Each depot can be supplied by multiple sources.

There are three types of costs: 1) processing costs, 2) annual fixed costs and 3) transportation costs.

Under these assumptions, we can design a mathematical formulation for the/a MSCN model as follows:

$$Min\ C = \sum_i \sum_s \sum_d \sum_m q_{m,s,d,i}\, utc_{m,s,d,i} + \sum_i \sum_s o_{s,i}\, afc_{s,i} + \sum_i \sum_s \sum_m qmp_{m,s,i} upc_{m,s,i} \quad (1)$$

$$\sum_m qmp_{m,s,i} cur_{m,i} \le cap_{s,i} o_{s,i} \quad \forall s \text{ and } i \quad (2)$$

$$\sum_s qmt_{m,s,d,i-1} = d_{m,d,i} \quad (3)$$

Where

N	Set of stages($N = \{1,2,\dots,n\}$)
n	The index of the last layer
F_i	Set of facilities of stage i $(i \in N)$
M_i	Set of materials that are outcomes of facilities of stage i $(i \in N - \{n\})$
$cap_{s,i}$	Capacity of source s of stage i $(i \in N - \{n\}, s \in F_i)$
$d_{m,d,i}$	Demand of depot d of stage i for material m produced at stage $i-1$ $(i \in N - \{1\}, d \in F_i, m \in M_{i-1})$
$afc_{s,i}$	Annual fixed cost for operating a source s of stage i $(i \in N - \{n\}, s \in F_i)$
$upc_{m,s,i}$	Unit processing cost of material m at source s of stage i $(i \in N - \{n\}, s \in F_i, m \in M_i)$
$utc_{m,s,d,i}$	Unit transportation cost of material m from source s of stage i to depot d of stage $i+1$ $(i \in N - \{n\}, d \in F_{i+1}, s \in F_i, m \in M_i)$
$cur_{m,i}$	Capacity utilization rate of material m at facilities of stage i $(i \in N - \{n\}, m \in F_i)$
$o_{s,i}$	1 if source s at stage i is open, 0 otherwise $(i \in N - \{n\}, s \in F_i)$
$qmt_{m,s,d,i}$	Quantity of material m transported from source s of stage i to depot d of stage $i+1$ $(i \in N - \{n\}, s \in F_i, d \in F_{i+1}, m \in M_i)$
$qmp_{m,s,i}$	Quantity of material m produced at source s of stage i $(i \in N - \{n\}, s \in F_i, m \in M_i)$

Eq. (1) shows the objective function of the MSCN design problem. Eq.(2) means that the sum of capacities, occupied by materials produced in a source, cannot be greater than its capacity. And constraint (3) means that the sum of the quantities of a material received by a depot must be equal to the demand of that depot.

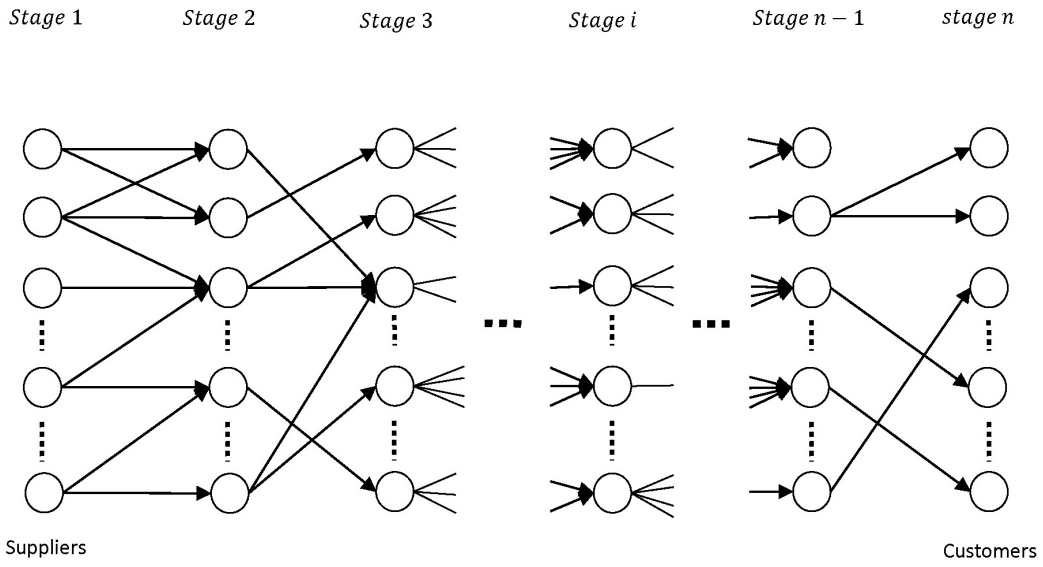

FIGURE 1. Structure of supply chain network.

3. Genetic Algorithm

The genetic algorithm (GA), introduced by Holland [6], is an evolutionary search technique that adopts Darwin'ssurvival of the fittest concepts. The most important feature of GAs is parallel searching and thus generating all possible solutions. This algorithm maintains a collection of solutions or chromosomes called a population. It initializes apopulation with potential solutions to the problem and seeks to produce better solutions by combining the best of the existing ones through the use of genetic operators [5].

In this section, the methodology for constructing the GA for the MSCN design problem is mentioned. In order to effectively represent this methodology, we illustrated genetic operators implemented in our proposed GA.

3.1. Representation

In general the performances of GAs and other evolutionary algorithms can be strongly affected by the problem representation; thus, the problem of encoding is the first step

aimed at coding each GA chromosome individual [2]. Different problems have different data structures or genetic representations. In this study, we used the priority-based encoding, which is a type of tree-based encoding [3, 4], for the MSCN design problem.

3.1.1. Priority Based Encoding

In the priority-based encoding for MSCN, each chromosome divides into segments where each segment represents a stage of SCN. Let N be set of stages for an MSCN, the chromosome based on priority-based encoding consist of $|N|$ segments, where each segment represents transportations of materials between a/the corresponding stage and the following stage. Let S_i, D_j and M_k, respectively be a set of sources at stage $i(S_i=F_i)$, set of depots at stage $J(D_j=F_j)$, and set of materials produced at stage k, *ith* segment of the chromosome consists of $|M_i|$ parts, and the length of each part is $|S_i|+|D_{i+1}|$. The value of each gene of the *ith* segment can be between 1 and $|M_i|(|S_i|+|D_{i+1}|)$[1]. Therefore the length of each chromosome is: $\sum_{i=1}^{|N|-1} |M_i|(|S_i|+|D_{i+1}|)$. Fig. 2 represents a transportation tree between stage $i-1$ and stage i and between stage i and stage $i+1$ of a MSCN with given transportation cost matrixes ($utc_{i-1,i}$ and $utc_{i,i+1}$) and its priority-based encoding.

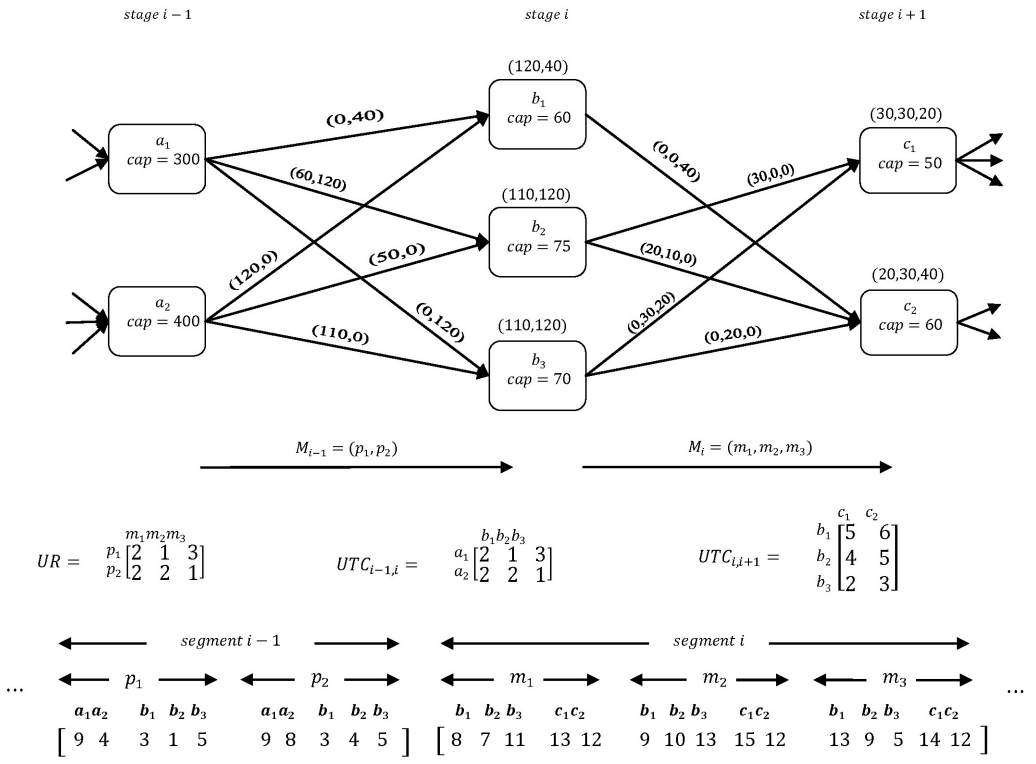

FIGURE 2. A sample of transportation tree for multi-product and its encoding.

To find out more about encoding and decoding procedures of segment-based encoding refer to [1, 13].

3.2. Generating the Initial Population

Initial population and population size are two important parameters of GAs. The initial population should have a gene pool as large as possible in order for the whole search space to be able to be explored, thus the initial population is, in most of cases, generated randomly. The size of the population is an important parameter because the larger a/the population is, the easier it is to explore the search space inspite of (any) increases in the time required by a GA to converge [12]. In this study, the population size is selected in proportion to the length of chromosomes and the diversities of gene values:

$$p = l \text{ x } d \tag{4}$$

Where
p is the population size, and
l is the length of the chromosomes:

$$l = \sum_{i=1}^{|N|-1} |M_i|(|S_i| + |D_{i+1}|) \tag{5}$$

d is the average diversity of the gene values:

$$d = \frac{\sum_{i=1}^{|N|-1} |M_i|(|S_i| + |D_{i+1}|)}{|N| - 1} \tag{6}$$

3.3. Evaluation

The evaluation is a process of assigning a value to each individual according to a fitness function such that better individuals have higher fitness values. The choice of fitness function is also very critical because it accurately has to measure the desirability of the features described by the chromosome [1]. Since the objective is the minimization of the total cost of the MSCN design, better solutions are those results in lower objective function. A higher fitness value means the better chromosome, so we define the following function to calculate each fitness value:

$$Fitness\ value\ = \frac{1}{objective\ value} \tag{7}$$

$$= \frac{1}{\sum_i \sum_s \sum_d \sum_m q_{m,s,d,i}\, utc_{m,s,d,i} + \sum_i \sum_s o_{s,i}\, afc_{s,i} + \sum_i \sum_s \sum_m qmp_{m,s,i} upc_{m,s,i}}$$

Although a single and exhaustive fitness function such as the one given in Eq. (7) can describe the goodness of an individual, it is not suitable for complex problems such as the MSCN design problem for two reasons:

First, the MSCN design problem consists of several features (such as number of stages, number of facilities at each stage and number of materials produced at each stage) and parameters (such as demand matrixes and transportation cost matrixes), that the complexity of the problem strongly depends on them. So evaluating individuals using a single fitness function will ignore most of the details just mentioned. Second, the length of chromosome varies with different values of the network features and we will have very long chromosomes for real world examples (according to Eq. (6)); therefore, the values of each gene will not play an appropriate role in evaluating all of such chromosomes using single evaluation functions. In this section, we define an alternative evaluation method to overcome the problems just mentioned.

3.3.1. Segment-based Evaluation

For the MSCN design problem, the chromosome based on priority-based encoding consists of several segments, so evaluating each segment independently gives more detailed information about that segment.In order to evaluate segments of an MSCN chromosome, we used a segment-based fitness function, defined as Eq.(8) for each segment of the chromosome:

$$f_i = \frac{1}{\sum_s \sum_d \sum_m q_{m,s,d,i} utc_{m,s,d,i} + \sum_s o_{s,i} afc_{s,i} + \sum_s \sum_m qmp_{m,s,i} upc_{m,s,i}} \tag{8}$$

where, f_i is the fitness function for *ith* segment of the chromosomes. The denominator of equation obtains the total costs (transformation costs, processing costs and annual fixed costs) of each segment.

3.4. Selection

Selection is about how to choose individuals from the population for crossing and how many offspring each will create. The purpose of selection is to emphasize fitter individuals in the population in hopes that their offspring will have higher fitness [12]. According to mentioned evaluation methods, we can consider two kinds of selections:

1) Chromosome-based selection using exhaustive fitness function given in Eq. (7),

2) Segment-based selection using segment-based fitness functions given in Eq. (8).

Fig. 3 demonstrates the difference between segment-based selection and chromosome-based selection.

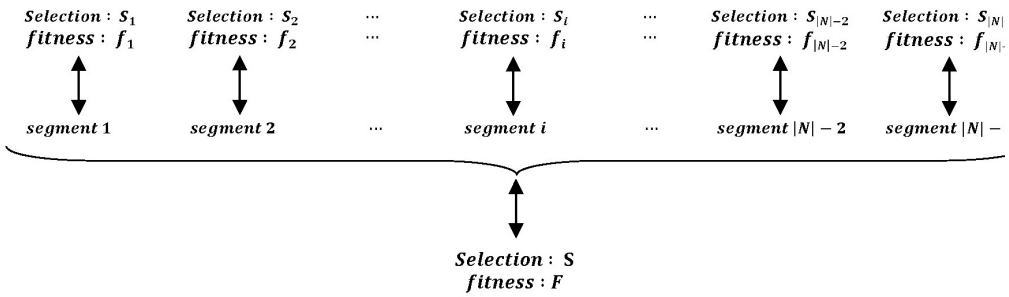

FIGURE 3. Segment-based selection using segment-based fitness functions against chromosome-based selection using exhaustive fitness function.

Different selection operators, such as a roulette wheel, tournament [10], back controlled [7], sequential and hybrid [7], can be chosen for different segments in segment-based selection methods. Hence in the selection step of our proposed genetic algorithm, two individuals will be selected for each segment according to the evaluation function defined for that segment and therefore we will select 2(N–1) parents to apply the crossover operator and generate one offspring.

3.5. Crossover

In this study, a segment-based crossover operator, which is based on uniform crossover [1], is employed for the MSCN design problem. In the segment-base crossover, each segment of the offspring is randomly selected with equal chance among the corresponding segments of parents selected for that segment. Fig.4 demonstrates the overall structure of the segment-base crossover for the MSCN design problem. For each segment *i,* two parents a_i and b_i, selected using segment-based selection s_i, participate in constructing of *ith* segment of offspring. This crossover operator utilizes a binary mask. Its length is equal to the number of stages in the MSCN.While "0" means that the offspring will inherit its genetic materials from the first parent, "1" means that the second parent will transfer its genetic materials to the off spring for the corresponding segment.

3.6. Mutation

The mutation operator is critical to the success of genetic algorithms since it determines the search directions and avoids convergence to local optima [8]. In this paper, we utilize

segment-based mutation as a mutation operator. In the segment-base mutation, firstly a decision about which segments will be mutated is given by the probability of 0.5, and then selected segments are mutated [1].

a_1	***segment 1***	*segment 2*	⋯	*segment i*	⋯	*segment* $\|N\|-2$	*segment* $\|N\|-1$
b_1	***segment 1***	*segment 2*	⋯	*segment i*	⋯	*segment* $\|N\|-2$	*segment* $\|N\|-1$
a_2	*segment 1*	***segment 2***	⋯	*segment i*	⋯	*segment* $\|N\|-2$	*segment* $\|N\|-1$
b_2	*segment 1*	***segment 2***	⋯	*segment i*	⋯	*segment* $\|N\|-2$	*segment* $\|N\|-1$
⋮				⋮			
a_i	*segment 1*	*segment 2*	⋯	***segment i***	⋯	*segment* $\|N\|-2$	*segment* $\|N\|-1$
b_i	*segment 1*	*segment 2*	⋯	***segment i***	⋯	*segment* $\|N\|-2$	*segment* $\|N\|-1$
⋮				⋮			
$a_{\|N\|-2}$	*segment 1*	*segment 2*	⋯	*segment i*	⋯	***segment* $\|N\|-2$**	*segment* $\|N\|-1$
$b_{\|N\|-2}$	*segment 1*	*segment 2*	⋯	*segment i*	⋯	***segment* $\|N\|-2$**	*segment* $\|N\|-1$
$a_{\|N\|-1}$	*segment 1*	*segment 2*	⋯	*segment i*	⋯	*segment* $\|N\|-2$	***segment* $\|N\|-1$**
$b_{\|N\|-1}$	*segment 1*	*segment 2*	⋯	*segment i*	⋯	*segment* $\|N\|-2$	***segment* $\|N\|-1$**
binary mask	**0**	**1**	⋯	**1**	⋯	**0**	**1**
c	***segment 1***	***segment 2***	⋯	***segment i***	⋯	***segment* $\|N\|-2$**	***segment* $\|N\|-1$**

FIGURE 4. The overall structure of segment-base crossover.

4. Design of Local Search

When applying the GA to a problem, two situations may occur: First, the GA is converging to the global optimal solution. When this situation occurs, the GA solution is continuously improved. Second, the GA is converging to a local optimal solution. In this situation the performance of the GA definitely deteriorates and the technique that helps improving this situation is to insert new individuals with certain high fitness values into a current GA loop [2]. To find such individuals, we need to search around the convergence area of the GA loop; so we can use the iterative hill climbing method suggested by Michalewicz [9]. The overall structure of this method is demonstrated in Fig.5. In order to apply a local search to GA loops, we must determine the convergence of the GA to the local optima. In this paper, we used the local search control mechanism proposed by Yun [17]. The logic behind this mechanism is that when the GA is continually converging, the similarity among the individuals of the GA population becomes higher, and therefore we can use a local search whenever the similarity among individuals becomes higher than a predefined threshold. This mechanism uses a similarity coefficient to measure the similarity between two individuals.

Suppose that $M=[v_{1m}, v_{2m}, \cdots, v_{Nm}]$ and $L=[u_{1l}, u_{2l}, \cdots, u_{Nl}]$ are two individuals of current population, we can calculate the similarity coefficient (SC_{im}) between L and M as follows:

$$SC_{lm} = \frac{\sum_{k=1}^{N} \partial(v_{km}, u_{kl})}{N_1} \tag{9}$$

Where

k	Index of gene
l, m	Indices of individuals
v_{km}	Value of kth gene from individual m
u_{kl}	Value of kth gene from individual l
N_1	Number of genes.

$$\partial(v_{km}, u_{kl}) = \begin{cases} 1, & if\ |v_{km} - u_{kl}| \leq \alpha \\ 0, & otherwise \end{cases}$$

α	A predefined coefficient that measures the similarity between v_{km} and u_{kl}

Since the priority-based encoding of the MSCN design problem produces individuals with multiple segments, we can define two types of similarity coefficient between two individuals: 1) similarity coefficient between corresponding segments of the two individuals, and 2) similarity coefficient between two individuals themselves. We define the first kind as follows:

$$SC_{lm}^{i} = \frac{\sum_{j=1}^{|M_i|} \sum_{k=1}^{|S_i|+|D_{i+1}|} \partial\left(a_{l,i,(j-1)(|S_i|+|D_{i+1}|+k}, a_{m,i,(j-1)(|S_i|+|D_{i+1}|+k}\right)}{|M_i|(|S_i| + |D_{i+1}|)} \tag{11}$$

Where

i	Index of segment of chromosome
j	Index of material at segment i
k	Index of facility at segment i
$\|M_i\|$	Number of materials that are outcomes of facilities of stage i ($i \in N - \{n\}$)
$\|S_i\|$	Number of sources of stage i ($i \in N - \{n\}$)
$\|D_i\|$	Number of depots of stage $i + 1$ ($i \in N - \{n\}$)
$a_{h,i,j}$	Value of jth gene of segment i from individual h

$$\partial\left(a_{l,i,j}, a_{m,i,j}\right) = \begin{cases} 1, & if\ \left|a_{l,i,j} - a_{m,i,j}\right| \leq \alpha \\ 0 & otherwise \end{cases}$$

Now we define similarity coefficient between two individuals as follows:

$$SC_{lm} = \frac{\sum_{n=1}^{N} SC_{lm}^{n}}{N} \tag{12}$$

Where

n	Index of segment
N	Number of segments in an individual

Therefore, the average $\overline{SC_{tm}}$ of the SC_{tm} for all individuals in current population can be expressed as follows:

$$\overline{SC_{lm}} = \frac{\sum_{l=1}^{N_1-1}\sum_{m=l+1}^{N_1} SC_{lm}}{N_2} \tag{10}$$

Where

N_1 Population size

N_2 Population sizeNumber of SC_{tm}

If the $SC_{tm} \geq \beta$, where β is a predefined threshold, then the similarity among the individuals is very high and the search by the GA alone may not guarantee an improvement of the solution. In this situation, therefore, it needs to use a local search technique, around the solutions converged by a GA loop to improve the GA performance. Therefore, we can regulate the use of the local search method in a GA loop under the following conditions:

apply local search method to GA loop; if $\overline{SC_{tm}} \geq \beta$

apply GA alone; otherwise

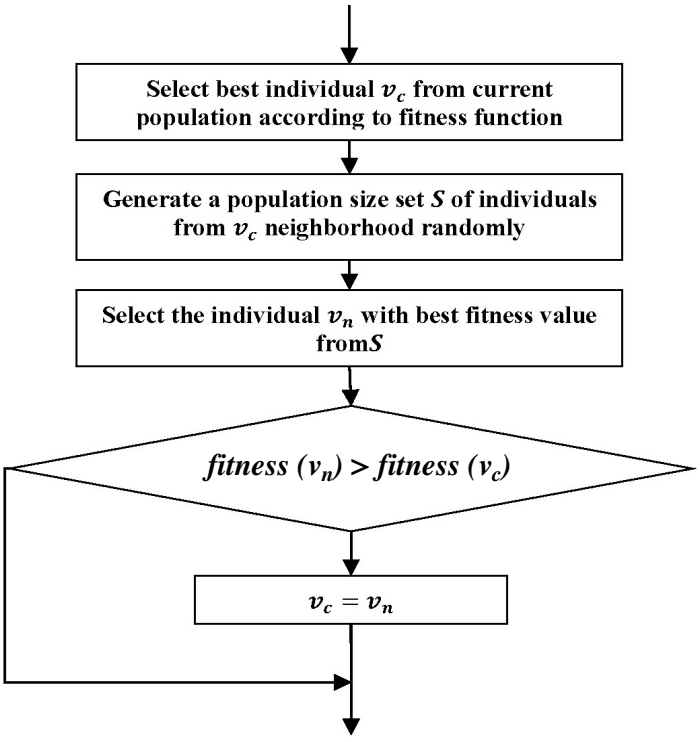

FIGURE 5. The overall structure of iterative hill climbing method suggested by Michalewicz [9]

5. Implementation of Proposed Hybrid Method

The proposed hybrid genetic algorithm (HGA) combines the GA suggested in section 3 and the iterative hill climbing method with a local search control mechanism mentioned in section 4. The algorithm which describes the process is:

Algorithm: HGA for multi-stage based supply chain network design problem.
Input: supply chain network (SCN), GA parameters
Output: best solution
Begin
Step1: Genetic Algorithm
Step1.1: Representation
Encode the problem using priority-based encoding discussed in section 3.1.1
Step1.2: Initial population generation
Calculate the population size using equation (4) and generate initial population randomly
Step1.3: Genetic Operators
Step1.3.1: Selection
Select two parents for each segment according the segment-based selection method discussed in section 3.4
Step1.3.4: Crossover
Crossover *2N* chromosomes selected in Step3 using segment-based crossover discussed in section 3.4
Step1.3.5: Mutation
Mutate the offspring generated in Step4 using segment-based mutation operator discussed in section 3.5 with pre- defied mutation probability
Step1.4: Evaluation
Evaluating current population using segment-based evaluation discussed in section 3.3.1
Step1.5: Stop Criterion Check
Check the pre-defined stopping criterions satisfaction.
If this criterions are satisfied output the best individual
and stop
Step2: Local search
Apply iterative hill climbing method proposed in [15] using local search controlling mechanism proposed in section 4 and Go to Step 1.3
End

---

6. Computational Results

To evaluate the performance of the proposed segment-based hybrid genetic algorithm (SHGA), the results of the SHGA are compared to the solutions obtained by 1) Simulated Annealing (SA), 2) simple genetic algorithm without segment-based operators (GA), 3) hybrid genetic algorithm with local search and without segment-based operators (HGA) [17] and 4) proposed segment-based GA without local search (SGA) on different MSCN design problems. To do this, we used several tests on different supply chain networks with different features (different number of stages and different number of facilities on different stages).

In this study, we used segment-based selection which applies a tournament selection (TS) operator for all segments. TS chooses each parent by choosing a tournament size subset of individuals using roulette selection and then choosing the best individual out of that set to be a parent [7]. In this article, the tournament size of two, was selected. The probabilities of crossover and mutation operators are selected as 0.85 and 0.05 respectively for all GAs used in tests. In the local search scheme of the proposed SHGA, the pre-defined coefficient (α) and the predefined threshold value (β) are set to 0.05 and 0.85, respectively.

Twenty different test problems (numbered as 1, …, 20) are generated for comparison. These test problems are divided into four groups (five test problem per group) based on their number of stages (N). Test problems in a same group are different in the number of facilities at each stage i($|F|_i$). All the parameters such as the number of materials for each facility, the capacity of each source facility, demand matrix of customers, annual fixed cost for operation of the source facilities, etc., are generated randomly for each test problem.

We applied the proposed SHGA and other four algorithms to each test problem; Each test was performed 20 times using MATLAB software. The minimum and mean costs for MSCNs are obtained when each algorithm reaches a given number of iterations varies with problem size, as shown in Table 2.

TABLE 2. The results of runs are obtained with different algorithms for MSCN design problem.

$N = 2$

P-No	$\|F_1\|$	$\|F_2\|$	*iteration number*	*SA*		*GA*		*HGA*		*SGA*		*SHGA*	
				best	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*
1	2	2	30	840	894.32	764	803.04	786	816.16	724	753.06	680	701.18
2	3	4	50	1311	1363.11	1008	1098	939	983.86	918	945.5	865	890.02
3	4	4	75	1918	2068.05	1881	1926.58	1714	1803.09	1582	1643.01	1404	1467.07
4	5	5	120	2580	2681.40	2472	2593.02	2448	2556.36	2214	2301.83	2008	2125.80
5	6	6	150	3217	3341.76	3203	3298.76	3164	3232.49	2968	3097.80	2850	2903.93

$N = 3$

	$\|F_1\|$	$\|F_2\|$	$\|F_3\|$	*iteration number*	*SA*		*GA*		*HGA*		*SGA*		*SHGA*	
					best	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*
6	2	2	2	100	2442	2510.18	2138	2214.40	2017	2093.16	1868	1954.03	1704	1760.16
7	2	3	5	200	3016	3227.30	2980	3171.01	2822	2889.43	2690	2800.11	2601	2683.44
8	3	3	6	300	4563	4675.92	4569	4632.78	4397	4469.02	4309	4374.44	4124	4254.78
9	4	4	7	400	6012	6103.65	5993	6098.23	5940	5992.92	5847	5976.10	5716	5794.86
10	5	6	7	500	7827	7984.87	7786	7829.83	7742	7803.02	7582	7635.60	7396	7478.27

$N = 4$

	$\|F_1\|$	$\|F_2\|$	$\|F_3\|$	$\|F_4\|$	*iteration number*	*SA*		*GA*		*HGA*		*SGA*		*SHGA*	
						best	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*
11	2	2	2	2	200	4820	4980.03	4831	4984.81	4751	4818.50	4701	4808.48	4588	4623.86
12	2	3	4	5	400	7104	7353.37	6911	7213.42	6786	6891.13	6718	6820.20	6613	6709.21
13	3	4	4	6	500	8453	8610.07	8390	8538.33	8412	8518.47	8149	8324.69	7920	8111.75
14	4	4	5	7	600	10761	11030.27	10219	10871.01	10186	10607.76	9706	9954.63	9228	9461.91
15	5	5	5	8	800	13281	13986.51	12711	13012.27	12693	13126.66	11878	12116.75	10963	11233.66

$N = 5$

	$\|F_1\|$	$\|F_1\|$	$\|F_1\|$	$\|F_1\|$	$\|F_1\|$	*iteration number*	*SA*		*GA*		*HGA*		*SGA*		*SHGA*	
							best	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*
16	2	2	2	2	2	400	7892	8125.65	7276	7548.25	7192	7502.38	6894	7213.44	6173	6287.7
17	2	3	3	4	5	700	12391	12674.67	11620	12016.09	11422	11735.35	10381	10271.33	8910	9271.9
18	3	3	3	5	5	800	13166	13419.54	12893	13201.26	12539	12863.82	10899	11080.80	9237	9702.6
19	4	4	4	5	6	1000	16403	16636.84	15447	15920.66	15193	15389.47	13866	14298.74	12168	12635.8
20	5	5	6	7	9	1300	21773	22739.92	19367	19993.64	17466	18528.74	16938	18005.28	14279	14839.5

Table 2 shows, SHGA yields the MSCNs with minimum costs compared with the other four (SA, GA, HGA and SGA) algorithms. Fig.5 depicts more comparative demonstration of the best solutions found by the five algorithms.

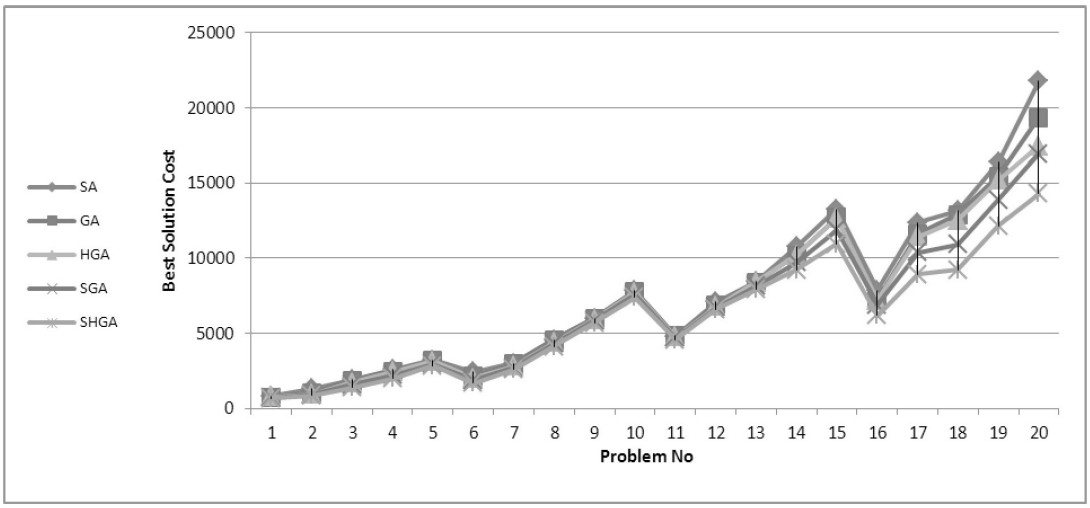

FIGURE 6. The best solutions cost found by five algorithms for test problems.

In Fig. 6, the SA generated the worst results in all tests and SGA generated the best results among the other four existing algorithms. Another important point is the effect of the number of stages in the SHGA performance. Because of the segment-based evaluation mechanism, as the number of stages becomes larger, the SHGA becomes more efficient than the other four existing algorithms. For example, suppose the test problems 15 and 18, which have 4 and 5 stages, respectively. The problem 15 has more facilities in its stages, comparing with problem 18. In spite of the fact that the best solutions found by five algorithms are in the same range of the two problems, they have more variance in problem 18.

Conclusions

In this study, a multistage-based SC model has been designed. This model occurs very often in today's manufacturing systems. However, MSCN design is an NP-hard problem and therefore representing and checking all feasible routes in the model may be impractical when the model is implemented in most conventional algorithms.

As a good alternative for improving this weakness, a new SHGA approach both with a segment-based operators and a local search technique has been proposed in this study. We employed the local search control mechanism proposed by Yun [17] in order to determine convergence of the GA to the local optima. To prove the efficiency of the proposed SHGA, the results of the SHGA are compared to the solutions obtained by several algorithms such as simulated annealing (SA), a simple genetic algorithm without segment-based operators (GA), a hybrid genetic algorithm with local search and without segment-based operators (HGA) [17] and aproposed segment-based GA without the local search (SGA) on different MSCN design problems. The computational results have shown that the proposed SHGA outperforms the other competing algorithms.

Although the proposed SHGA finds lower-cost MSCNs, it exhibits two major deficiencies: 1) the run time for each iteration of the proposed algorithm is higher than the other existing algorithms. This occurs for two reasons: first, the use of segment-based GA operators, instead of exhaustive operators, needs more computations in each iteration. Second, the calculation of the average similarity coefficient for a population ($\overline{SC_{lm}}$) is an expensive process since it needs to evaluate the pairwise similarities between all individuals in the population. 2) The threshold-based similarity coefficient mechanism, introduced in this paper, defines two crisp conditions for a population, the converged population

($\overline{SC_{lm}} \geq \beta$) and a non-converged population ($\overline{SC_{lm}} \geq \beta$). For example, if we select $\beta = 0.8$, then two populations P_1 with $\overline{SC_{lm}} = 0.79$ and P_2 with $\overline{SC_{lm}} = 0.81$ are non-converged and converged populations respectively, while the two cases may not show any significant difference.

Our future works are:

1) Introducing some more efficient and exhaustive evaluation functions which are able to represent detailed information about an MSCN individual.

2) Introducing an efficient GA convergence control mechanism, that uses some non-crisp concepts such as fuzzy logic to remove the crisp behavior of the threshold-based similarity control mechanism.

Acknowledgment

The authors are extremely grateful to the anonymous referees for having provided the many valuable comments and helpful suggestions which helped to improve the presentation of this paper.

References

[1] F. Altiparmak, M. Gen, L. Lin, I. Karaoglan, A steady-state genetic algorithm for multi-product supply chain network design, *Computers & Industrial Engineering,* **56**, (2009), 521–537.

[2] A. Costa, G. Celano, S. Fichera, E.Trovato, A new efficient encoding/decoding procedure for the design of a supply chain network with genetic algorithms, *Computers & Industrial Engineering,* **59**(4), (2010), 986–999.

[3] M. Gen, R. Cheng, Genetic algorithms and engineering optimization. *New York: John Wiley and Sons,* (2000).

[4] M .Gen, F. Altiparmak,L. Lin, A genetic algorithm for two-stage transportation problem using priority-based encoding. *OR Spectrum,* **28,** (2006), 337–354.

[5] M. Hajiaghaei-Keshteli, The allocation of customers to potential distribution centers in supply chain networks: *GA and AIA approaches, Applied Soft Computing,* **11**(2), (2010), 2069–2078.

[6] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, (1975).

[7] M. Kaya, The effects of a new selection operator on the performance of a genetic algorithm, *Applied Mathematics and Computation,* **217**(19), (2011), 7669–6778.

[8] M. Kaya, The effects of two new crossover operators on genetic algorithm performance, *Applied Soft Computing* **11**(1), (2011), 881–890.

[9] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Program. Spring-Verlag, (1994).

[10] B. L. Miller, D. E. Goldberg, Genetic Algorithms, Tournament Selection, and the Effects of Noise, *Complex Systems,* **9,** (1995), 193–212.

[11] M. S. Pishvaee, M .Rabbani, A graph theoretic-based heuristic algorithm for responsive supply chain network design with direct and indirect shipment. *Advances in Engineering Software,* **42**(3), (2010), 57–63.

[12] S. N. Sivanandam, S. N. Deepa, Introduction to Genetic Algorithms, New York: Springer Berlin Heidelberg, (2008).

[13] A. Syarif, Y. Yun, M. Gen. Study on multi-stage logistics chain network: A spanning tree-based genetic algorithm approach. *Computers & Industrial Engineering,* **43**(1-2), (2002), 299–314.

[14] L. C. Wang, T. L. Chen, Y.Y. Chen, H. Y. Miao, S. C. Lin, S. T. Chen, Genetic algorithm approach for multi–objective optimization of closed–loop supply chain network, *Proceedings of the Institute of Industrial Engineers Asian Conference 2013,* (2013), 149–156.

[15] M. J. Yao, H. W. Hsu, A new spanning tree–based genetic algorithm for the design of multi–stage supply chain networks with nonlinear transportation costs, *Optimization and Engineering,* **10**(2), (2009), 219–237.

[16] W. C. Yeh, A hybrid heuristic algorithm for the multistage supply chain network problem, *Int J AdvManufTechnol,* **26**(5–6), (2005), 675–685.

[17] Y. Yun, C. Moon, D. Kim, Hybrid genetic algorithm with adaptive local search scheme for solving multistage-based supply chain problems, *Computers & Industrial Engineering,* **56**(3), (2009), 821–838.