PERGAMON

Computers & Industrial Engineering 43 (2002) 407–421

computers & industrial engineering

www.elsevier.com/locate/dsw

An evolutionary algorithm for optimizing material flow in supply chains

F. Elizabeth Vergara[a,*], Moutaz Khouja[a], Zbigniew Michalewicz[b]

[a]*Business Information Systems and Operations Management Department, The Belk College of Business Administration, The University of North Carolina at Charlotte, Charlotte, NC 28223, USA*
[b]*Department of Computer Science, The University of North Carolina at Charlotte, Charlotte, NC 28223, USA*

Abstract

Supply chain management literature calls for coordination between the different members of the chain. Materials should be moved from one supplier to the next according to a just-in-time schedule. In this paper we develop an evolutionary algorithm (EA) for optimal synchronization of supply chains. In developing our algorithm, we use the economic delivery and scheduling model and analyze supply chains dealing with multiple-components. We test the performance of the proposed EA and show that it provides optimal, or near optimal, solutions for a wide range of problems. The EA is shown to be much faster at solving large problems than an enumeration procedure and exhibits robust behavior when tested on a variety of different problem parameters. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Evolutionary algorithms; Supply chain management; Synchronized production

1. Introduction

In today's increasing global and competitive marketplace, it is imperative that members of a supply chain work together in an effort to minimize overall transportation, holding, and setup cost. Efficient and effective management of material flow across a supply chain is critical to its success (Handfield & Nichols, 1999). A supply chain involves suppliers (one or more tiers), assemblers/manufacturers, distribution centers, retailers and customers as shown in Fig. 1. The figure shows two types of supply chain configurations. A simple supply chain is one in which each supplier is captive and supplies one or more components to only one upper tier supplier or assembly facility (AF). A complex supply chain is one in

* Corresponding author. Tel.: +1-704-687-3242; fax: +1-704-687-6330.
E-mail addresses: fvergara@email.uncc.edu (F.E. Vergara), mjkhouja@email.uncc.edu (M. Khouja), zbyszek@uncc.edu (Z. Michalewicz).

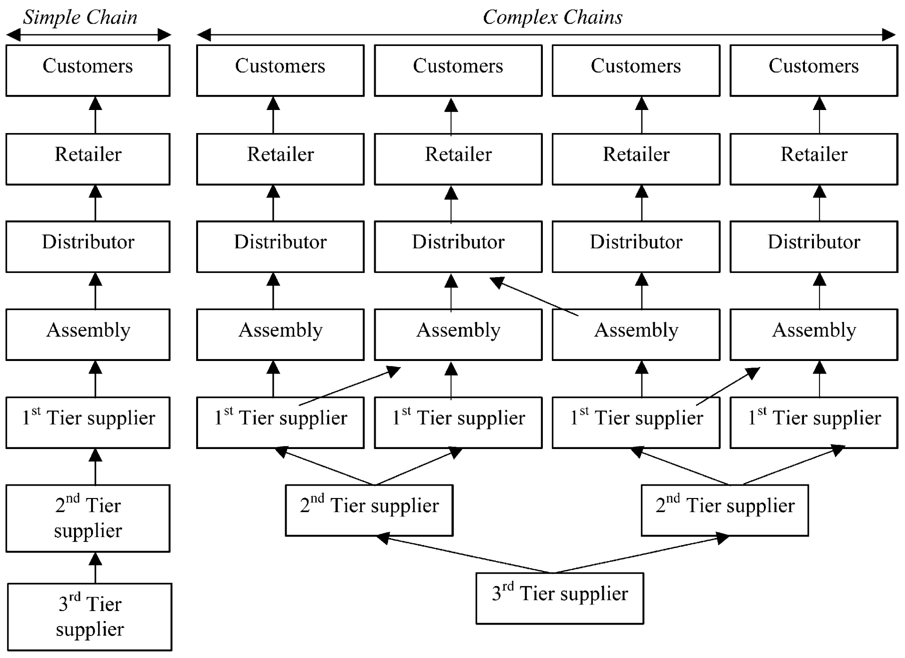


Fig. 1. Two types of supply chains.

which at least one supplier supplies one or more components to two or more upper tier suppliers or assembly facilities.

The purpose of this work is to develop an evolutionary algorithm (EA) that will find the production sequence at each supplier for multiple-components and a synchronized delivery cycle time that would minimize transportation, setup, and holding costs across a simple multi-stage supply chain. Synchronization of the supply chain in terms of a just-in-time cycle time is advocated as a way of achieving continuous improvement. In well-managed supply chains inventory flows between members of the chain with little delay (Handfield & Nichols, 1999). The goal of supply chain management is to optimize the whole system (Simchi-Levi, Kaminsky, & Simchi-Levi, 2000).

The proposed EA identifies a near optimal, or ideally, an optimal, solution for supply chain synchronization problems and then calculates the cost to each supplier if a synchronized delivery cycle time were to be used, in addition, it calculates the cost to each supplier if their independent optimal delivery cycle time and production sequence were to be used. Therefore, the cost associated with implementing the synchronized cycle time and sequence for each supplier can be computed. The EA is intended to find satisfactory solutions in a short period of time.

The proposed EA is designed for multi-supplier/multi-component supply chain configurations. The Economic Lot and Delivery Scheduling Problem (ELDSP) introduced by Hahm and Yano (1995a) is the foundation for this work. However, traditionally the ELDSP encompasses a single supplier and an AF. This work expands the ELDSP by applying its concepts to cover the entire supply chain. Khouja (2000) developed an algorithm for solving the multi-supplier ELDSP. The author applied the ideas of the RAND algorithm of Kaspi and Rosenblatt (1991) to solve the supply chain synchronization problem.

The main shortcoming of this approach is that can be used for constrained problems. On the other hand, constraints do not represent a problem for EAs (Michalewicz, 1992).

In Section 2, we describe the problem. In Section 3, we provide a summary of relevant aspects of EAs, followed in Section 4 by our proposed implementation of the EA for the simple supply chain problem. In Section 5 we discuss the results of testing the proposed EA. Section 6 wraps up the paper with a brief discussion of managerial implications, suggestions for future research, and some concluding comments.

2. Supply chain synchronization problem

The goal of the proposed EA is to solve a multi-stage synchronized simple supply chain problem, which is an extension of the traditional ELDSP. The original ELDSP is characterized by a supply chain consisting of a supplier that produces components on a single production line or machine, accumulates the components, and then delivers them to an AF (Hahm & Yano, 1992, 1995a,b). The AF uses the components at a constant rate. Hahm and Yano (1995a) considered a situation where the supplier is captive and supplies many components to only one assembly plant. The assumptions of the ELDSP are:

a. the supplier produces the components on a single machine one at a time,
b. production and usage rates are deterministic and constant,
c. the supplier incurs sequence independent setup cost and setup time,
d. there is a single delivery from the supplier to the AF per cycle, during which one batch of each component is produced,
e. both the supplier and the assembler incur linear holding cost on component inventory, and
f. the delivery charge per shipment is fixed.

Let:

g	supplier index, $g = 1, \ldots, G$
j	component index, $j = 1, \ldots, J$
p_{jg}	production time per unit of component j at supplier g
u_{jg}	the value added by supplier g to component j
U_{jh}	the value of component j after the transformation by supplier h, $U_{jh} = \sum_{g=1}^{h} u_{jg}$
I	fraction holding cost
S_{jg}	setup cost for component j at supplier g
s_{jg}	setup time for component j at supplier g
D_{jg}	demand for component j at supplier g
A_g	transportation cost per delivery at supplier g and
T_g	setup interval (time between setups); also equal to the delivery interval (time between deliveries) for supplier g, a decision variable

Because holding cost is dependent on accumulated inventory, the production sequence of each component is an important consideration. The number of possible production sequences at any given supplier is equal to the factorial of the number of components involved. For example, a three-component problem has six possible supplier production sequences. The inventory levels for two cycles of the {2,1,3} sequence are shown in Fig. 2 for a given value of T, where T is the delivery cycle time.

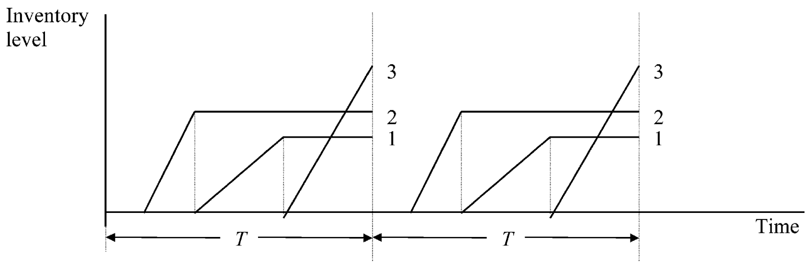


Fig. 2. Inventory levels at the supplier for the {2,1,3} sequence.

We first introduce the independent solution (IS) for each supplier. Under the IS each supplier optimizes its own component production sequence and cycle time. The IS solution is usually infeasible because it does not allow for synchronization. We then formulate the supply chain synchronization problem.

2.1. The independent solution

Hahm and Yano (1995a) developed an efficient heuristic algorithm for solving the ELDSP where only a single supplier is included, and tested the performance of the algorithm. The experimental results showed that the algorithm identified the optimal solution for a broad range of problems. The algorithm builds on some results of scheduling theory (Baker, 1974). The idea is to produce components with long processing times and low holding costs early in the sequence and components with short processing time and high holding cost late in the sequence. This policy ensures that components with high holding cost will not wait for long times to be shipped. Let [*i*] be the index of the component produced in the *i*th position. The total cost for supplier *g* per unit time is:

$$\mathrm{TC}_g = \frac{A_g + \sum_{j=1}^{J} S_{jg}}{T_g} + \frac{1}{2} T_g I \sum_{j=1}^{J} D_j^2 p_{jg} u_{jg} + I \sum_{i=1}^{J} D_{[i]} u_{[i]g} \sum_{j=i+1}^{J} (T_g D_{[j]} p_{[j]g} + s_{[j]g}) + T_g I \sum_{j=1}^{J} D_j U_{j,g-1}. \tag{1}$$

Define $\nu_g \in C$, the set of all possible production sequences at supplier g, and

$$S_g = \sum_{j=1}^{J} S_{jg},$$

$$s_g = \sum_{j=1}^{J} s_{jg},$$

$$\omega_{jg} = I D_{jg} u_{jg},$$

$$\omega_g = I \sum_{j=1}^{J} D_j u_{jg},$$

$$\gamma_g = \sum_{j=1}^{J} D_j \omega_{jg} p_{jg},$$

$$C_g = I \sum_{j=1}^{J} D_j U_{j,g-1},$$

$$Z_{1g}(\nu) = \sum_{i=1}^{J} \omega_{[i]g} \sum_{j=i+1}^{J} s_{[j]g},$$

and

$$Z_{2g}(\nu) = \sum_{i=1}^{J} \omega_{[i]g} \sum_{j=i+1}^{J} D_{[j]} p_{[j]g}.$$

The total cost becomes

$$\mathrm{TC}_g = \frac{A_g + S_g}{T_g} + T_g\left[\frac{1}{2}\gamma_g + Z_{2g}(\nu_g) + C_g\right] + Z_{1g}(\nu_g), \tag{2}$$

and the ELDSP for supplier g is to minimize TC_g subject to:

$$\sum_{j=1}^{J} (s_{jg} + p_{jg} D_j T_g) \le T_g, \qquad g = 1, 2, \ldots, G \tag{3}$$

Constraint (3) ensures that there is sufficient time within the delivery interval T_g to set up and produce the components needed at the AF during T_g. Since the second derivative $\mathrm{d}^2\mathrm{TC}_g/\mathrm{d}T_g^2 > 0$ for $T_g > 0$, setting the first derivative $\mathrm{dTC}_g/\mathrm{d}T_g = 0$ gives the optimal delivery cycle for a fixed production sequence for supplier g as

$$T_{gU}^{*} = \sqrt{\frac{A_g + S_g}{\gamma_g/2 + Z_{2g}(\nu_g) + C_g}}. \tag{4}$$

Since Eq. (4) may not always yield a feasible solution, the optimal delivery cycle time for a given production sequence is given by

$$T_g^{*} = \max\left\{T_{Ug}^{*}, \tau_g\right\}, \tag{5}$$

where

$$\tau_g = \frac{s_g}{1 - \sum_{j=1}^{J} p_{jg} D_j}. \tag{6}$$

2.2. The synchronized solution

If the supply chain is synchronized at a delivery cycle time T, the total inventory holding cost at the AF per unit time is

$$\mathrm{TC_a} = \frac{1}{2} TI \sum_{j=1}^{J} D_j U_{jG} + \frac{S_{G+1}}{T}, \tag{7}$$

where S_{G+1} is the ordering cost at the AF. The total cost for the whole chain is

$$\mathrm{TC} = T\left[\frac{1}{2} I \sum_{j=1}^{J} D_j U_{jG} + \sum_{g=1}^{G}\left[\frac{1}{2}\gamma_g + C_g + Z_{2g}(\nu_g)\right]\right] + \sum_{g=1}^{G} Z_{1g}(\nu) + \frac{\sum_{g=1}^{G+1} (A_g + S_g)}{T}. \tag{8}$$

For a fixed sequence

$$\frac{\mathrm{dTC}}{\mathrm{d}T} = I\left[\frac{1}{2} \sum_{j=1}^{J} D_j U_{jG} + \sum_{g=1}^{G}\left[\frac{1}{2}\gamma_g + C_g + Z_{2g}(\nu_g)\right]\right] - \frac{\sum_{g=1}^{G+1} (A_g + S_g)}{T^2}, \tag{9}$$

and

$$\frac{\mathrm{d}^2\mathrm{TC}}{\mathrm{d}T^2} = \frac{2\sum_{g=1}^{G+1} (A_g + S_g)}{T^3}. \tag{10}$$

For any $T > 0$, $\mathrm{d}^2\mathrm{TC}/\mathrm{d}T^2 > 0$. Therefore, for a fixed set of sequences at the different suppliers, TC is convex in T and the optimal value of T is:

$$T_1^* = \sqrt{\frac{\sum_{g=1}^{G+1} (A_g + S_g)}{\left[\frac{1}{2} I \sum_{j=1}^{J} D_j U_{jG} + \sum_{g=1}^{G}\left[\frac{1}{2}\gamma_g + C_g + Z_{2g}(\nu_g)\right]\right]}}. \tag{11}$$

Since constraint (3) may be binding for one or more suppliers, the optimal cycle time is given by

$$T_g^* = \max\left\{T_1^*, \tau_g\right\}. \tag{12}$$

Table 1
Number of local minimums

Components	Suppliers				
	2	3	4	5	6
2	**4**	**8**	**16**	**32**	**64**
3	**36**	**216**	**1296**	**7776**	**46,656**
4	**576**	**13,824**	**331,776**	**7,962,624**	191,102,976
5	**14,400**	**1,728,000**	207,360,000	24,883,200,000	2,985,984,000,000
6	**518,400**	373,248,000	268,738,560,000	193,491,763,200,000	139,314,069,504,000,000

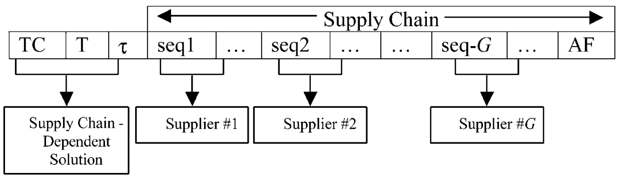
Because there are a finite number of candidate solutions, an enumeration procedure for minimizing Eq. (8) can be developed. For each set of sequences $(\nu_1^{(i)}, \nu_2^{(i)}, \dots, \nu_G^{(i)})$, the optimal value of T can be obtained using Eq. (12). The enumeration procedure can therefore identify all local minima and subsequently the global minima. Thus, the enumeration procedure provides means for checking the EA solution against the global minima. While the enumeration procedure will guarantee a global optimal solution it is computationally prohibitive for large problems. The problem size increases very rapidly as illustrated in Table 1. A problem with G suppliers and J components will have $(J!)^G$ local minima. For example, a four-supplier five-component problem has $(5!)^4 = 207{,}360{,}000$ local minima, each corresponds to a unique combination of production sequences for each supplier.

3. Evolutionary algorithms

An EA is a problem solving technique that uses the concepts of evolution and hereditary to produce quality solutions to complex problems that typically have enormous search spaces and are therefore difficult to solve. A well designed EA allows for the efficient and effective exploration and exploitation of the problem's search space of feasible solutions in an effort to identify the global optimal, or near optimal, solution to difficult problems.

EAs create and manipulate a group of possible solutions referred to as a population. Each possible solution within the population is called an individual. The population undergoes change throughout the run of the EA thereby evolving the individuals toward a best solution. Within the EA, the population loops through a series of processes a number of times; each executed loop is known as a generation. These processes include an evaluation process, an alteration process, and a selection process. These processes may occur in various orders, however, each is required at each generation (Michalewicz, 1992).

The evaluation process uses an evaluation function that assesses the relative fitness of each individual of the population at each generation. In addition, at each generation a number of individuals are subjected to some form of change. These alterations are manifested through the use of genetic operators. Genetic operators can be either mutation operators, which introduce small changes within a single individual, or crossover operators, which cut and paste different parts from two or more individuals together in order to create new individuals called offspring. The probability of an individual experiencing some form of transformation within any given generation is subject to the predefined parameters of the probability of mutation, and/or the probability of crossover. Through this process, some, or all, of the



*Any given seq location above contains a randomly generated combination of all integers from 1 to J. For example, if $J = 3$ the possible sequence combinations that could be recorded in any seq location are {1,2,3}, {1,3,2}, {2,1,3}, {2,3,1}, {3,1,2}, {3,2,1}.

Fig. 3. Representation of an individual with n suppliers.

individuals are altered and used to create a new population for the next generation. Finally, the EA uses the evaluated fitness of each individual to promote the survival of the best individuals to the next generation. This use of selective pressure encourages the population to converge to a quality solution. The EA will run for a predetermined maximum number of generations or until some specified terminating condition is meet.

Each EA is unique in its design with regard to several important elements. Some of these elements include data structure, genetic operators, method for creating the initial population, constraint handling techniques, evaluation function, selection method, generational policy, parameters, and terminating conditions. Parameters include population size, maximum number of generation, probability of mutation, and/or the probability of crossover. However, regardless of the differences, all EAs attempt to evolve the individuals within the population through the use of genetic operators and selective pressures to converge at a suitable solution to complex problems.

4. An evolutionary algorithm for supply chain synchronization

The proposed EA is designed for the simple supply chain configuration of Fig. 1 and provides a starting point for future work with complex supply chains. The proposed EA begins with a randomly generated initial population that is composed of individuals with only feasible solutions. Randomly generated integer sequences that represent permutations of possible production sequences for all suppliers within the supply chain are added to each individual. The generation of integers is accomplished using a random generator and then scaling the generated numbers as needed and rounding them to the closest integer. In addition, modifications and changes produced by the two operators used in the EA on the population of individuals are restricted so that only feasible solutions are produced. Finally, the population size is maintained constant throughout the run of the EA.

The problem has one constraint, the minimal cycle time τ, given by Eq. (6), necessary to produce the required demand for all the components at each supplier. The value of T for the supply chain can never be less than the largest τ of any of the suppliers.

The main elements of an EA include an evaluation function, representation, genetic operators, and selection method. The evaluation function used here is the total cost for the supply chain, given by Eq. (8). Individuals within the population with the lowest total cost are deemed to be of a better quality than those with a higher total cost.

The proposed EA uses a unique data structure. While synchronized supply chain problems are

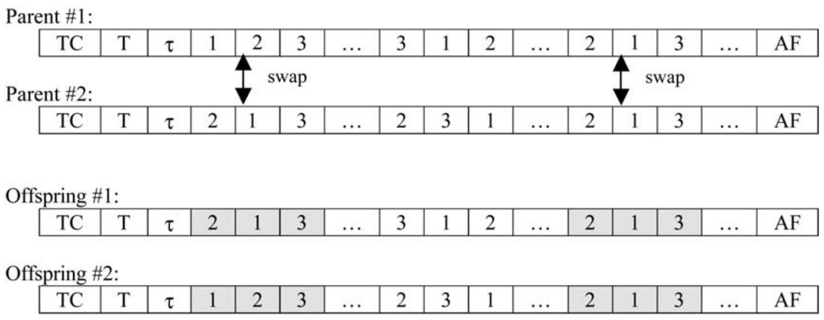

Parent #1:
TC T τ 1 2 3 … 3 1 2 … 2 1 3 … AF
swap
swap
Parent #2:
TC T τ 2 1 3 … 2 3 1 … 2 1 3 … AF
Offspring #1:
TC T τ 2 1 3 … 3 1 2 … 2 1 3 … AF
Offspring #2:
TC T τ 1 2 3 … 2 3 1 … 2 1 3 … AF

Fig. 4. High level cross-over operator process.

essentially discrete permutation problems that are best represented using integer values that correspond to real world representation of possible solutions, the actual data structure used for this work is a floating point array. This floating point array data structure allows for the recording of detailed cost and cycle time values as well as sequence information for each supplier within each individual in the population. The data structure is designed to be dynamic, allowing the EA to be used for a wide variety of possible combination of suppliers and components without modification to the EA program code. Fig. 3 is a graphical representation of an individual with G suppliers. The first three locations within an individual contains the total cost (TC), synchronized delivery cycle time (T), and the maximum τ for all suppliers. The next part of the individual is populated with a sequence of integers that represent a possible sequence in which to produce the components for the first supplier. Following this sequence, additional

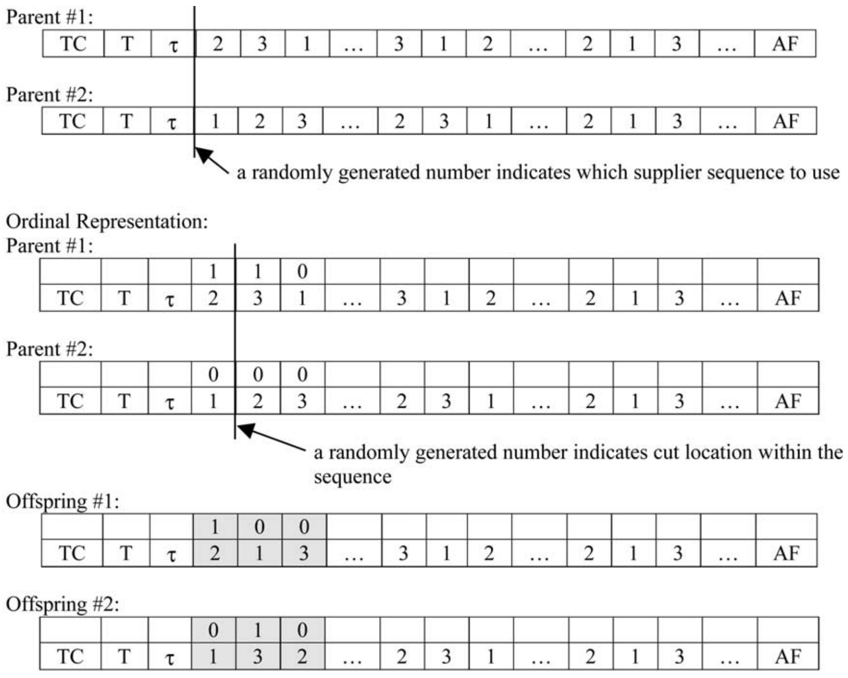

Parent #1:
TC T τ 2 3 1 … 3 1 2 … 2 1 3 … AF
Parent #2:
TC T τ 1 2 3 … 2 3 1 … 2 1 3 … AF
a randomly generated number indicates which supplier sequence to use
Ordinal Representation:
Parent #1:
1 1 0
TC T τ 2 3 1 … 3 1 2 … 2 1 3 … AF
Parent #2:
0 0 0
TC T τ 1 2 3 … 2 3 1 … 2 1 3 … AF
a randomly generated number indicates cut location within the sequence
Offspring #1:
1 0 0
TC T τ 2 1 3 … 3 1 2 … 2 1 3 … AF
Offspring #2:
0 1 0
TC T τ 1 3 2 … 2 3 1 … 2 1 3 … AF

Fig. 5. Low level cross-over operator process.

information about the supplier is added to the individual, which pertains to the IS. The remaining array elements of the individual contains similar information for each supplier in order of the material flow in the chain. The last entry in the individual array is the cost to the AF.

The mechanisms used to manipulate the individuals within the EA, known as genetic operators, are another important characteristic of any EA. Two operators were created, the high level crossover operator shown in Fig. 4 and the low level crossover operator shown in Fig. 5. These two operators were developed to address the unique nature of the problem and the structure of the chosen data representation. As can be seen in Fig. 3, each individual is composed of many sub-chromosomes and other data related to the whole supply chain. Each sub-chromosome represents one supplier and appears within the individual, a large composite chromosome, in the order in which the supplier appears within the supply chain. In addition, each sub-chromosome contains two separate parts. The first part contains a component production sequence for that supplier. The second part contains pertinent information about the supplier. Therefore, because of the unique nature of the sub-chromosomes and their non-interchangeable relationship within the large composite chromosome (i.e. the individual) traditional crossover and mutation genetic operators could not be used.

Because the first part of each sub-chromosome is ostensibly a Traveling Salesman Problem (TSP) where each supplier's production sequence is dependent on the synchronized cycle time T for the supply chain, traditional genetic operators used within a sub-chromosome would inevitably create infeasible solutions. The infeasibility can result from including any given component more than once and excluding one or more components from the production sequence. While some EAs allow for the inclusion of such infeasible solutions, or include repair algorithms to transform them into feasible solutions, the proposed EA was designed to control alterations so that only feasible solutions will be created (Michalewicz, 1992).

For the high level crossover operator, the program randomly generates two numbers that identify two individuals from the present generation of individuals. The individual with the best (lowest) total cost is chosen to be the first parent. This process is then repeated to find a second parent. The high level genetic operator then treats each sub-chromosome as a gene and alternately swaps these genes between two parents in order to create two new offspring individuals. This technique maintains the order of the sub-chromosomes within the composite individual, introduces random change into the supply chain, and creates two feasible offspring individuals. These two new individuals are added to the next generation's population.

The low level genetic operator was developed to introduce changes into corresponding sub-chromosomes of two individuals while maintaining feasibility. This was realized by implementing an ordinal representation technique that has been previously used in TSPs (Michalewicz, 1992). The ordinal representation technique allows for the introduction of random changes within the component production sequence part of a sub-chromosome while maintaining its feasibility.

As with the high level crossover, two parents are selected by choosing the best two individuals from two pairs of randomly selected individuals from the present generation. A randomly generated number is used to select a supplier within the supply chain. This supplier's sequence becomes subject to modification. Another randomly generated number is produced to indicate a cut location for a one-point crossover. The corresponding supplier sequence within each of the two parents is then translated into an ordinal representation prior to a one-point crossover operation. After the crossover is completed on the ordinal representation of each parent's chosen supplier sequence, the resulting ordinal representation is translated back to its original format. This creates two new offspring individuals that are subsequently added to the population of individuals for the next generation.

Table 2
Average number of generations for the EA to find a solution (average of five problems for each problem size)

Number of components	Number of suppliers				
	2	3	4	5	6
2	**1**	**1**	**1**	**1**	**1**
3	**1**	**1.4**	**2.9**	**6**	**8.2**
4	**1.5**	**6.9**	**13.8**	**23**	27.8
5	**7**	**16.6**	34.2	43.6	45.2
6	**14**	30.2	45.2	49.8	54.2

Several selection methods are incorporated within the EA. An elitist selection method is used to ensure that the best solution from each generation is retained in the population for the next generation. In addition, for both the high and low level operators, a tournament selection is conducted to choose the best individuals to be used as parents. Finally, a tournament selection is used to fill in the remaining individuals of the population for the next generation from the previous generation's population. Therefore, it is possible that a given individual may be used to create the new population for undetermined number of times.

The termination condition for the EA states that if all members of the population have the same total cost value (i.e. all members of the population have converged and have the same value for total cost) then the program should stop. Otherwise, processing will terminate when the program has cycled through a predetermined number of generations.

5. Results

To test the performance of the EA, an enumeration procedure that identifies the global minima was used. The enumeration procedure was used because there are no other algorithms for solving the problem when dealing with many suppliers in series. The results of the EA are compared to the results from the enumeration procedure and both the dollar and percent differences are calculated. The number of local minimums for each size in Table 1 are computed using $(J!)^G$. The 17 problem sizes shown in bold in the table were used to test the EA.

A number of preliminary tests using three supplier, five-component (3×5) problems were run to determine the optimal combination of population size, maximum number of generations, and the probability of the two crossover operators. For this research, a population size of 500, a probability of high level crossover of 0.2, and a probability of low level crossover of 0.79 were found to be the best. The maximum number of generations was set to equal 100. However, for all tests of the EA, the best solutions have consistently been found in relatively small number of generations, well below 100. As expected, the larger and more complex the problem, the more generations, on average, are required to identify the best solution. Table 2 shows the average number of generations it takes the EA to find a good solution.

In addition, it is interesting to note the difference in the running time of the EA as opposed to the enumeration procedure. The top number in each cell of Table 3 represents the run time of the enumeration procedure, while the lower number is the run time of the EA. As the table shows, as the problems

Table 3
Average run time for enumeration method vs. EA

Components	Suppliers				
	2	3	4	5	6
2	0	0	0	0	0.01
	0.54	0.68	0.80	0.96	1.04
3	0.02	0.03	0.02	0.11	0.75
	0.67	0.83	1.01	1.2	1.34
4	0.02	0.14	4.49	135.51	3252
	0.8	1.03	1.22	1.43	1.67
5	0.14	22.06	1[a]	88[a]	10,589[a]
	0.98	1.22	1.45	1.72	2
6	5.6	1[a]	806[a]	580,608[a]	418,037,760[a]
	1.13	1.38	1.68	2.06	2.34

[a] Estimated and given in hours. All other estimates are in seconds.

become larger, the EA quickly becomes much faster than the enumeration procedure. For example, for an average five-supplier and four-component problem, the EA took 1.43 s to find a solution vs. 135.51 s for the enumeration procedure.

To verify that the EA performs well for a broad range of problems the following experiment was conducted. Table 4 shows the seven uniform distributions that were used to generate problem parameters to test the EA. The three parameters used are (1) the ratio of the setup cost to the holding cost (S_{jg}/I), (2) the tightness of the capacity constraint ($Y_g = \sum_{j=1}^{n} p_{jg}D_j$), and (3) the amount of value added at each supplier (u_{jg}).

Thirty problems were generated for each of the seven different distributions of Table 4 resulting in 210 randomly generated problems for each of the 17 different problem sizes shown in bold in Table 1 for a total of 3570 problems.

For clarity, problems are classified as type I or type II. Type I problems are ones for which the EA identified the global optimal solution. Type II problems are ones in which the EA did not identify the global optimal solution (which was found using enumeration). Table 5 shows the percentage of all type I and type II problems for each tested distribution. From this table it can be observed that the lowest type I

Table 4
Distributions used to generate test problem parameters

Parameter group	S_{jg}/I	Y_g	u_{jg}
1	$U[10–15]$	$U[0.85, 0.95]$	$U[30, 60]$
2	$U[10–15]$	$U[0.55, 0.65]$	$U[30, 60]$
3	$U[10–15]$	$U[0.55, 0.65]$	$U[30, 60] + j \times U[20, 25]$
4	$U[10–15]$	$U[0.85, 0.90]$	$U[30, 60]$
5	$U[20–25]$	$U[0.55, 0.65]$	$U[30, 60]$
6	$U[20–25]$	$U[0.55, 0.65]$	$U[30, 60] + j \times U[20, 25]$
7	$U[20–25]$	$U[0.85, 0.90]$	$U[30, 60]$

Table 5
Summary of all type I and type II problems for each group

Group	Type I (%)	Type II (%)
1	97.6471	2.3529
2	97.8431	2.1569
3	96.2745	3.7255
4	96.8627	3.1373
5	96.6667	3.3333
6	96.2745	3.7255
7	96.6667	3.3333
Average	96.8908	3.1092

percentage is 96.2745%, the average type I percentage for all problems is 96.8908%, and the highest type I percentage is 97.8431%.

Table 6 breaks down type II problems by size. Note that only five sizes out of the 17 tested resulted in type II problems. The 2×6 size (two-supplier and six-component) experienced more type II problems than any other size even though it is smaller. One possible explanation for this is that there are many alternate near optimum solutions clustered together in different regions of the search space.

When type II problems are examined, it is easy to get a clear understanding of how the EA performed even when it failed to identify the global minimum. Type II problems have extremely small errors. Table 7 shows the average percent difference for all type II problems by size. From this table we can see that although the 2×6 problems have the largest frequency of type II problems, they have the lowest average percent difference (0.001730%). The highest average percent differences belong to 5×4 and 3×5 problems at 0.003818 and 0.003331%, respectively. The largest type II problem percent difference of 0.011125% is found for a 3×5 problem. For this problem, the EAs solution was \$334,313.844, and the enumeration solution process was \$334,276.656, resulting in \$37.188 difference.

6. Discussion and conclusion

We propose an EA which identifies an optimal, or near optimal, synchronized delivery cycle time and suppliers' component sequences for a multi-supplier, multi-component simple supply chain. The EA

Table 6
Summary of type II problem by problem size

Size	Count	Frequency (%)
2×5	4	3.60
2×6	45	40.54
3×5	36	32.43
4×4	7	6.31
5×4	19	17.12
Total	111	100.00

Table 7
Average percent difference of all type II problem by problem size

Size	Average (%) difference
2×5	0.002568
2×6	0.001730
3×5	0.003331
4×4	0.002690
5×4	0.003818

also calculates a synchronized delivery cycle time for the entire supply chain, the cumulative cost throughout the supply chain, and the cost to each supplier.

The EA has been shown to be very efficient at finding, if not the global optimal solution, at least a very near optimal solution to complex permutation problems. Of particular interest is the speed at which the EA is able to identify solutions for supply chain synchronization problems.

This work addresses a unique type of problem from an evolutionary computing perspective. Past work that focused on solving complex scheduling problems using evolutionary computing techniques, such as Khouja, Michalewicz, and Vijayaragavan (1998), Khouja, Michalewicz, and Wilmot (1998), Khouja, Michalewicz, and Satoskar (2000), typically focused on solving scheduling problems for a single manufacturing facility and used classical binary and Grey representation, both crossover and mutation genetic operators, allowed individuals within the population to contain infeasible solutions, and used two-point crossover. This is in contrast to the proposed EA which used floating point representation (eliminating the need for mapping techniques), used two hybrid crossover genetic operators, allowed for only feasible solutions, and employed one-point crossover. As can be seen from Fig. 3, each individual is made up of a sub-chromosomes for each supplier as well as aggregate information about the supply chain and information pertaining to the final AF. While the data stored within an individual is complex, the data structure is designed to be easy to read and dynamic, allowing the EA to solve a variety of different size problems. In addition, the maximum number of generations for this EA was set relatively low, 100, because the EA identified good solutions in a very small number of generations, and the population size was set rather high, 500, based on the results of preliminary tests.

Future research may focus on the modification of the two genetic operators, or perhaps the introduction of a third genetic operator, to further improve the EAs performance. The use of two-point crossover may also prove to be beneficial. Another possibility is to create a new population by performing a tournament selection on a current population before the genetic operators are applied. This technique may increase selective pressure, thereby improving the performance of the EA. The initial results reported in this work are very promising and it is anticipated that the proposed EA will be expanded in future research to address complex supply chain problems shown in Fig. 1. It appears that EAs may be used to provide real time solutions to complex practical business problems.

References

Baker, K. R. (1974). *Introduction to sequencing and scheduling*, New York: Wiley.

Hahm, J., & Yano, C. A. (1992). The economic lot delivery scheduling problem: The single item case. *International Journal of Production Economics*, *28*, 235–252.

Hahm, J., & Yano, C. A. (1995a). The economic lot delivery scheduling problem: The common cycle case. *IIE Transactions*, *27*, 113–125.

Hahm, J., & Yano, C. A. (1995b). The economic lot delivery scheduling problem: Models for nested schedules. *IIE Transactions*, *27*, 126–139.

Handfield, R. B., & Nichols Jr., E. L. (1999). *Introduction to supply chain management*, Upper Saddle River, NJ: Prentice-Hall.

Kaspi, M., & Rosenblatt, M. J. (1991). On the economic ordering quantity for jointly replenishment items. *International Journal of Production Research*, *29*, 107–114.

Khouja, M. (2000). *Synchronization in supply chains: Implications for design and management*, unpublished.

Khouja, M., Michalewicz, M., & Vijayaragavan, P. (1998). Evolutionary algorithm for economic lot and delivery scheduling problem. *Fundamenta Informaticae*, *35*, 113–123.

Khouja, M., Michalewicz, Z., & Wilmot, M. (1998). The use of genetic algorithms to solve the economic lot size scheduling problem. *European Journal of Operational Research*, *110*, 509–524.

Khouja, M., Michalewicz, M., & Satoskar, S. (2000). A comparison between genetic algorithms and the RAND method for solving the joint replenishment problem. *Production Planning and Control*, *11*, 556–564.

Michalewicz, Z. (1992). *Genetic algorithms + data structures = evolution programs*, New York: Springer.

Simchi-Levi, D., Kaminsky, P., & Simchi-Levi, E. (2000). *Designing and managing the supply chain*, New York: Irwin McGraw-Hill.