Print ISSN: 2321-6379
Online ISSN: 2321-595X
DOI: 10.17354/ijssl/2017/116

Original Article

Comparing Two Replenishment Policy for Solving Economic Lot and Delivery Scheduling Problems in a Three-echelon Supply Chain

Hamidreza Kia, Seyed Hassan Ghodsypour

Department of Industrial Engineering and Management Systems, Amirkabir University of Technology, Tehran, Iran

Abstract

In this paper, we investigate the economic lot and delivery scheduling problem (ELDSP) for a three-stage supply chain. This problem is a combined lot sizing and sequencing problem. The supply chain includes multiple suppliers, multiple fabricators and only a single assembler. All of the parameters, such as demand rate, are deterministic and production setup times are sequence-independent. The common cycle time and integer multipliers policies are adapted as a replenishment policy for synchronization throughout the supply chain. Since the problem is NP-hard, we propose a hybrid metaheuristic algorithm (HGTSA) which combines a genetic algorithm with tabu search to find the solution in large-scale problems. The results of computational experiments demonstrate the efficiency of the proposed algorithm.

Key words: Supply chain, Lot sizing, Genetic algorithm, Tabu search

INTRODUCTION

The coordination of the chain members is one of the important objectives of supply chain, since members of each stage have different and even opposite profit. Synchronization can increase coordination among the chain members. The synchronization causes balance between decision making for internal production scheduling and the external delivery in the chain.One of the prevalent problems in the literature that examines these two cases simultaneously is economic lot and delivery scheduling problem (ELDSP).

Clark and Scarf (1960) present a recursive decomposition algorithm to determine optimal policies for a serial multi-echelon structure. Hahm and Yano (1992) study a supply chain consisting one supplier and one assembler and could determine the interval between production and delivery for a single item. Hahm and Yano (1995) develop this problem for multiple items and presented a heuristic algorithm for determining the common cycle time. Khouja (2000) study this problem with the assumption of rework cost for achieving the required quality and also variable production rate. Khouja (2003) formulate a three-stage supply chain and dealt with three coordination mechanisms. Jensen and Khouja (2004) present a heuristic algorithm for achieving the optimal solution in polynomial time by developing Hahm and Yano (1995). Clause and Ju (2006) develop the heuristic algorithm presented by Jensen and Khouja (2004). Their algorithm found the optimal solution of large scale problem at a shorter time.

Torabi et al (2006) analyze a two-echelon supply chain including one supplier and one assembler and multiple items with common cycle time. They proposed a hybrid genetic algorithm for the solution of ELDSP. Nikandish et al (2009) examine a three-echelon supply chain, including one supplier and multiple assemblers and multiple retailers with common cycle time and presented the optimal solution for medium scale problems. Osman and Demrli (2012) analyze the economic lot and delivery scheduling problem for a multi-stage supply chain with multiple items and develop an algorithm to find the optimal solution for a synchronized replenishment strategy. In the literature review, there is no

Access this article online

www.ijss-sn.com

Month of Submission : 06-2017
Month of Peer Review : 06-2017
Month of Acceptance : 07-2017
Month of Publishing : 07-2017

Corresponding Author: Seyed Hassan Ghodsypour, Department of Industrial Engineering and Management Systems, Amirkabir University of Technology, Tehran, Iran. Tel.: +98 21 64545390, E-mail: Ghodsypo@aut.ac.ir

contribution to ELDSP in such supply chains. Osman and Demrli (2012) suppose that integer multiplies policy are parameter while in our paper is variable.

This paper is organized as follows. Section 2 described the problem definition. In Section 3, the parameters, variables and mathematical modeling are provided. Section 4 presents the details of the proposed hybrid Algorithm. Section 5 provides the Computational results. Finally, conclusions are presented in Section 6.

PROBLEM DEFINITION

In this paper, we have focused on the coordination among the members of a three-stage supply chain. The supply chain includes multiple suppliers, multiple fabricators and only single assembler.

The suppliers convert raw materials into components. Fabricators transform these components into items which are applied for production of the final products by the assembler.

Other assumptions are as follows:

- Each node has a single production line at the stage of suppliers and fabricators.
- The carried quantity of each stage's node is as much as demand of the next stage.
- Just one shipment is allowed to be sent at the end of each cycle time.
- The values of demand and production rate are fixed and certain.
- The holding cost of each item per unit time is fixed and certain.
- The delivery cost is fixed.
- For suppliers and fabricators stage, the setup time is sequence-independent.
- For suppliers and fabricators stage, the setup cost is fixed and certain.
- Ordering cost is fixed for each retailer.

The objective of this study is achieving coordination among the members. Therefore, synchronization is necessary. Two types of synchronization, full synchronization by applying the common cycle time policy and the partial synchronization by applying the integer multipliers policy, is selected.

MATHEMATICAL MODELING

In formulating the problem, we follow Osman and Demirli (2010) and extension it for our paper. The parameters and variables of the model are defined as follows:

Indices

s: index of suppliers, $s = 1,2, ..., S$.
f: index of fabricators, $f = 1,2, ..., F$.
a: index of assemblers, $a = 1$.
i : index of items in each facility $f, s, a,$ or. r

Parameters

q: sequence position in each facility f or s:
n_s: index of the number of items at s^{th} supplier.
n_f: index of the number of items at f^{th} fabricator.
n_a: index of the number of items at assembler.
D_{si}: demand rate of i^{th} item at s^{th} supplier.
D_{fi}: demand rate of i^{th} item at f^{th} fabricator.
D_{ai}: demand rate of i^{th} item at assembler.
D_{si}: production rate of i^{th} item at f^{th} supplier.
P_{fi} : production rate of i^{th} item at f^{th} fabricator.
HB_{si}: holding cost per unit of th unprocessed item per unit time at s^{th} supplier.
HB_{fi} : holding cost per unit of i^{th} unprocessed item per unit time at f^{th} fabricator.
H_{ai}: holding cost per unit of i^{th} item per unit time at assembler.
HA_{si}: holding cost per unit of i^{th} processed item per unit time at th supplier.
HA_{fi} : holding cost per unit of i^{th} processed item per unit time at f^{th} fabricator.
d_s : transportation cost per delivery at s^{th} supplier.
d_f: transportation cost per delivery atth fabricator.
d_a: transportation cost per delivery at assembler.
c_{si} : production setup cost of ^{i}th item atth supplier.
c_{fi}: production setup cost of i^{th} item at f^{th} fabricator.
c_a : ordering cost at assembler.
t_{si} : production setup time of i^{th} item at s^{th} supplier.
t_{fi}: production setup time of ith item at f^{th} fabricator.

Variables

T: cycle time.
X_{siq}: 1 if item i is assigned to q^{th} position at s^{th} supplier, otherwise 0.
X_{fiq} : 1 if item i is assigned to q^{th} position atth fabricator, otherwise 0.

m_2 *and* m_1: integer multiplier for integer multipliers policy (i.e., the cycle time at the assembler is T, m_1T at any fabricator stage, and it equals m_2m_1T at any supplier stage).

Modeling

Min TC=

$$\sum_{s=1}^{S}\left[\begin{array}{l}\sum_{i=1}^{n_s}HB_{si}\sum_{q=1}^{n_s}\left(\frac{m_2m_1TD_{si}^2}{2P_{si}}X_{siq}+D_{si}X_{siq}\left(\begin{array}{l}t_{si}+\sum_{j=1}^{n_s}\sum_{p=1}^{q-1}X_{sjp}\\ j\neq i\\ \left(t_{sj}+\frac{m_2m_1TD_{sj}}{P_{sj}}\right)\end{array}\right)\right)\\ +\sum_{i=1}^{n_s}HA_{si}\sum_{q=1}^{n_s}\left(\begin{array}{l}\frac{m_2m_1TD_{si}^2}{2P_{si}}X_{siq}+D_{si}X_{siq}\sum_{\substack{j=1\\ j\neq i}}^{n_s}\sum_{p=q+1}^{n_s}X_{sjp}\\ \left(t_{sj}+\frac{m_2m_1TD_{sj}}{P_{sj}}\right)\end{array}\right)\end{array}\right]$$

$$+\sum_{f=1}^{F}\left[\begin{array}{l}\sum_{i=1}^{n_f}HB_{fi}\sum_{q=1}^{n_f}\left(\frac{m_1TD_{fi}^2}{2P_{fi}}X_{fiq}+D_{fi}X_{fiq}\left(\begin{array}{l}t_{fi}+\sum_{j=1}^{n_f}\sum_{p=1}^{q-1}X_{fjp}\\ j\neq i\\ \left(t_{fj}+\frac{m_1TD_{fj}}{P_{fj}}\right)\end{array}\right)\right)\\ +\sum_{i=1}^{n_f}HA_{fi}\sum_{q=1}^{n_f}\left(\begin{array}{l}\frac{m_1TD_{fi}^2}{2P_{fi}}X_{fiq}+D_{fi}X_{fiq}\sum_{\substack{j=1\\ j\neq i}}^{n_f}\sum_{p=q+1}^{n_f}X_{fjp}\\ \left(t_{fj}+\frac{m_1TD_{fj}}{P_{fj}}\right)\end{array}\right)\end{array}\right]$$

$$+\sum_{i=1}^{n_a}\left(H_{ai}D_{ai}\right)\frac{T}{2}+\frac{1}{T}\left[\begin{array}{l}\frac{1}{m_2m_1}\sum_{s=1}^{S}\left(d_s+\sum_{i=1}^{n_s}c_{si}\right)\\ +\frac{1}{m_1}\sum_{f=1}^{F}\left(d_f+\sum_{i=1}^{n_f}c_{fi}\right)+d_a+c_a\end{array}\right] \quad (1)$$

Subject to:

$$\sum_{i=1}^{n_s}\left[t_{si}+\frac{m_2m_1TD_{si}}{P_{si}}\right]\leq m_2m_1T \qquad \forall\ s=1,2,\ldots,S \quad (2)$$

$$\sum_{i=1}^{n_f}\left[t_{fi}+\frac{m_1TD_{fi}}{P_{fi}}\right]\leq m_1T \qquad \forall\ f=1,2,\ldots,F \quad (3)$$

$$\sum_{i=1}^{n_s}X_{siq}=1 \qquad \forall\ s=1,2,\ldots,S;\ \ q=1,2,\ldots,n_s \quad (4)$$

$$\sum_{q=1}^{n_f}X_{siq}=1 \qquad \forall\ f=1,2,\ldots,F;\ \ i=1,2,\ldots,n_f \quad (5)$$

$$\sum_{i=1}^{n_f}X_{fiq}=1 \qquad \forall\ f=1,2,\ldots,F;\ \ q=1,2,\ldots,n_f \quad (6)$$

$$\sum_{q=1}^{n_f}X_{fiq}=1 \qquad \forall\ f=1,2,\ldots,F;\ \ i=1,2,\ldots,n_f \quad (7)$$

$$X_{siq}\ \text{is binary} \qquad \forall\ s=1,2,\ldots,S;\ \ i=1,2,\ldots,n_s;\ \ q=1,2,\ldots,n_s \quad (8)$$

$$X_{fiq}\ \text{is binary} \qquad \forall\ f=1,2,\ldots,F;\ \ i=1,2,\ldots,n_f;\ \ q=1,2,\ldots,n_f \quad (9)$$

HYBRID ALGORITHM

Genetic algorithms (GAs) are intelligent stochastic search techniques inspired from the principle of 'survival-of-the-fittest' in natural evolution and genetics. GAs have been applied successfully for a wide variety of combinatorial optimization problems to find optimal or near optimal solutions since its introduction by Holland (1992).

GAs start with an initial set of solutions, called population. Each solution in the population is called a chromosome (or individual). The chromosomes are evolved through successive iterations, called generations, by genetic operators (selection, crossover and mutation) that mimic the principles of natural evolution. A'fitness value' is assigned to each individual according to a problem specific objective function. GA explore solutions with increasing fitness, i.e., the higher the fitness, the more likely the genes of a chromosome are propagated to the next generations (Naso et al., 2007, Torabi et al., 2006).

Although GA can be directly applied to complex combinatorial optimization problems, each generation of the algorithm must maintain a large population size. With the expansion of the problem size, the computational time needed will increase dramatically. Besides, GA usually converges prematurely, which is mainly caused by a lack of diversity in the population. In addition, the mutation operator is inadequate for a systematic local search. Compared with GA, tabu search (TS) has faster convergence rate. However, the search performance of TS greatly depends on the initial solution. So, GA explores well the search space while TS intensifies the search in promising regions. According to the strengths and weaknesses of these two algorithms, we apply TS to replace the mutation operator in GA and proposed Hybrid genetic-tabu search algorithm (HGTSA). We find sequnces of production (binary variables) from HGTSA while value of cycle time is obtained through an optimization process.

Encoding and Decoding

There are two different representation schemes (or formats) to represent the discrete part of a solution for the problem,

i.e. sequence vectors in different production line or node (Wardono and Fathi, 2004 and Cheng and Gen, 1997). We used Wardono and Fathi (2004). Suppose that we have two suppliers in the first stage and one fabricator in the middle stage that the fabricator has 4 items and the supplier 1 and 2 have 6 and 3 items, respectively.

In Figure 1, the code of the individual corresponding solutions is sequencing 3-5-4-2-1-6 at supplier 1 and 1-3-2 at supllier 2 and 4-2-3-1 at fabricator.

Selection

Selection is the process of determining the number of times, trials, a particular individual is chosen for reproduction and, thus, the number of offspring that an individual will produce. A roulette wheel selection procedure has been applied for selection operator in our algorithm. In this procedure, a real-valued interval, Sum, is determined as the sum of individual raw fitness values over all the individuals in the current population. Individuals are then mapped one-to-one into contiguous intervals in the range [0,Sum]. The size of each individual interval corresponds to the fitness value of the associated individual. To select an individual a random number is generated in the interval [0,Sum] and the individual whose segment spans the random number is selected. This process is repeated until the desired number of individuals has been selected.

Crossover

The crossover is basic operator for producing new chromosomes in the GA. It produces new individuals that have some parts of both parent's genetic material.

In the literature review, several crossover operators have been proposed (Iyer and Saxena, 2004 and Wang and Wu, 2004). Among them, the following crossover operators have been widely used: partially matched crossover (PMX) intending to keep the absolute positions of genes and linear order crossover (LOX) intending to preserve relative positions. Thus, we used these two operators in our initial tests, and found that the crossover LOX works better for the problem under consideration. This crossover operator works as follows:

Step 1. First, select a parent at random, and then choose a subsequence of components with a random size within 1 to n-1.

Step 2. Produce a proto-offspring by copying the subsequence into the corresponding positions of it.

3	5	4	2	1	6
1	3	2			
4	2	3	1		

Figure 1: A sample chromosome

Step 3. Place the remaining components from the other parent by making a left-to-right scan. An example of this operator is illustrated in Figure 2.

Tabu Search Based Mutation

The mutation operator is reimplemented with a tabu search algorithm. Here, we take the solution obtained by GA as the initial solution for the TS. The neighborhood structure is defined as swap and we can obtain a neighborhood solution when we choose two different genes randomly and then exchange their locations. Consequently, the two genes which have been exchanged are recorded as the element in the tabu list. Besides, when the new solution is better than the best-so-far solution, we accept it no matter whether the move exits in the tabu list. The fundamental idea underlying tabu search is to avoid repeated search in the same area of the solution space. To this end, tabu search involves some essential concepts such as tabu list, tabu length, and aspiration criterion.

Fitness Function

In order to mimic the natural process of survival of the fittest, the fitness evaluation function assigns to each member of the population a value reflecting their relative superiority. In our problem, solutions with lower costs imply better solutions. Therefore, the proposed fitness function for each individual is defined as:

$$f_k = \frac{1}{TC} \tag{10}$$

Where TC achieved from Eq. (1). In Eq. (1), three type variables exist. First type, binary variables that obtain from HGTSA, call sequencing. The second type, integer multipliers that we check all possible for them in the range of 1 to 3 (i.e., 3×3 situation). The third type, cycle time that for each sequencing and situation finds from Eq. (20) and among all situations for integer multipliers, best fitness register for each individual. Eq. (11) shows the value of T that minimizes the total cost derived by differentiating Eq. (1) with respect to T.

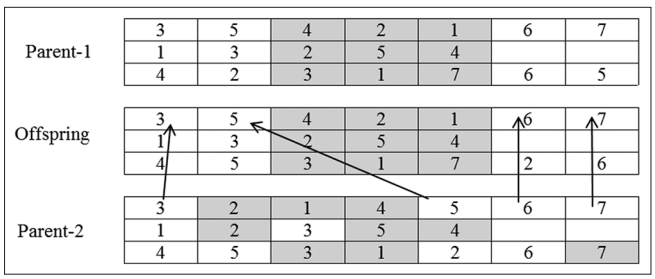

Figure 2: Illustration of LOX crossover

$$T_{dif}=\sqrt{\frac{\frac{1}{m_2 m_1}\sum_{s=1}^{S}\left(d_s+\sum_{i=1}^{n_s}c_{si}\right)+\frac{1}{m_1}\sum_{f=1}^{F}\left(d_f+\sum_{i=1}^{n_f}c_{fi}\right)+d_a+c_a}{0.5\sum_{i=1}^{n_a}\left(H_{ai}D_{ai}\right)+W_1+W_2+W_3+W_4+W_5+W_6}} \quad (11)$$

where

$$W_1=\sum_{s=1}^{S}\sum_{i=1}^{n_s}\sum_{q=1}^{n_s}\left(HB_{si}+HA_{si}\right)\frac{m_2 m_1 D_{si}^2}{2P_{si}} \quad (12)$$

$$W_2=\sum_{s=1}^{S}\sum_{i=1}^{n_s}\sum_{q=1}^{n_s}\sum_{\substack{j=1\\ j\neq i}}^{n_s}\sum_{p=1}^{q-1}HB_{si}D_{si}\frac{m_2 m_1 D_{si}}{P_{si}} \quad (13)$$

$$W_3=\sum_{s=1}^{S}\sum_{i=1}^{n_s}\sum_{q=1}^{n_s}\sum_{\substack{j=1\\ j\neq i}}^{n_s}\sum_{p=q+1}^{n_s}HA_{si}D_{si}\frac{m_2 m_1 D_{si}}{P_{si}} \quad (14)$$

$$W_4=\sum_{f=1}^{F}\sum_{i=1}^{n_f}\sum_{q=1}^{n_f}\left(HB_{fi}+HA_{fi}\right)\frac{m_1 D_{fi}^2}{2P_{fi}} \quad (15)$$

$$W_5=\sum_{f=1}^{F}\sum_{i=1}^{n_f}\sum_{q=1}^{n_f}\sum_{\substack{j=1\\ j\neq i}}^{n_f}\sum_{p=1}^{q-1}HB_{fi}D_{fi}\frac{m_1 D_{fi}}{P_{fi}} \quad (16)$$

$$W_6=\sum_{f=1}^{F}\sum_{i=1}^{n_f}\sum_{q=1}^{n_f}\sum_{\substack{j=1\\ j\neq i}}^{n_f}\sum_{p=q+1}^{n_f}HA_{fi}D_{fi}\frac{m_1 D_{fi}}{P_{fi}} \quad (17)$$

By considering Eqs. (2) and (3) that ensure the feasibility of T to cover setup and production times of all products, Eq. (18) and (19) achieved. Then the optimal value of T, Eq. (20), can be found as the maximum between feasible T resulting from Eqs. (2) and (3) and T obtained from Eq. (11).

$$T_{min1}=max\left\{\frac{\sum_{i=1}^{n_s}t_{si}}{m_2 m_1\left(1-\sum_{i=1}^{n_s}\frac{D_{si}}{P_{si}}\right)}\right\} \quad \forall\ s=1,2,\ldots,S \quad (18)$$

$$T_{min2}=max\left\{\frac{\sum_{i=1}^{n_f}t_{fi}}{m_1\left(1-\sum_{i=1}^{n_f}\frac{D_{fi}}{P_{fi}}\right)}\right\} \quad \forall\ f=1,2,\ldots,F \quad (19)$$

$$T_{Opt}=max\left\{T_{dif},T_{min1},T_{min2}\right\} \quad (20)$$

Termination Criterion

HGTSA is terminated if the best-so-far solution does not change for 50 consecutive generations or maximum number of generations is executed, whichever comes first.

COMPUTATIONAL RESULTS

In this section, in order to evaluate and compare the performance of proposed HGTSA and traditional GA, we consider thirteen different problem sizes that shown in Table 3. For each problem instance, 10 problems are randomly generated and the required parameters for these problems are shown in Table 1. Table 2 shows parameter setting for GTSA.

For each instance, we do not know the global optimal cost because of the nonlinear nature of the Eq.(1), and the prohibitive required computation time. Therefore, we have compared the total cost obtained for each instance by HGTSA, with the best known solution for the optimal total cost. Let $TC_{Algorithm}$ denotes the average total cost obtained via HGTSA or GA, and TC_{best} is the best known solution. We can calculate the relative percentage deviation (RPD) as:

$$RPD=\frac{TC_{Algorithm}-TC_{Best}}{TC_{Best}}\times 100\% \quad (21)$$

The results of HGTSA and GA with common cycle time and integer multipliers policies are summarized in Table 4. The second column in Table 4 gives RPD for HGTSA with common cycle time policy. For integer multipliers policy, the third and fourth columns give RPD for GA and HGTSA, respectively. The average RPD for the collection of instances is 27.38% for HGTSA and 38.79% for GA in integer multipliers policy. It is noted that the solution obtained from HGTSA is better than GA. Thus, this average value by itself in our experiments indicates the

Table 1: Data generation for parameters

Parameter	Distribution function	Parameter	Distribution function
D_{si}, D_{fi}, and D_{ai}	Uniform (1000,5000)	d_s, d_f and d_a	Uniform (100,1000)
P_{si}, P_{fi}	Uniform (6000,10000)	C_{si} and C_{fi}	Uniform (5000,9000)
HB_{si} and HB_{fi}	Uniform (10,50)	C_a	Uniform (5000,9000)
HA_{si}, HA_{fi}, and H_{ai}	Uniform (80,120)	t_{si} and t_{fi}	Uniform (0.001,0.025)

efficiency of HGTSA. Moreover, we observe that the RPD for the instances increases when the problem size increases. However, this increase can be due to degradation in the performance of HGTSA due to increase in corresponding solution space. Since we do not know the corresponding optimal cost, we cannot judge accurately in this context.

Looking to the fifth and sixth columns, it is clear that as the problem size increases the HGTSA consumes most of the solution time in comparison with GA while the HGTSA have a better solution in relation to GA. The last column of Table 4 demonstrates the percentage of cost savings attained by applying the integer multipliers policy in relation to common cycle time for HGTSA. The results of the other problems show that synchronizing the supply chain at the integer multipliers policy results in a cost reduction that can reach 17.89% compared to the common cycle time policy.

Table 2: Paremeter setting for HGTSA

Parameter	Value
Population size	500
Maximum number of generations	1000
Maximum number of generations without improvement	10
The tabu length	8
Crossover probability	0.8
Tabu search probability	0.2

CONCLUSIONS AND FUTURE EXTENSIONS

In this paper, we investigate the economic lot and delivery scheduling problem (ELDSP) for a three-stage supply chain. This problem is a combined lot sizing and sequencing problem. The supply chain includes multiple suppliers, multiple fabricators and only a single assembler. All of the parameters, such as demand rate, are deterministic and production setup times are sequence-independent. The common cycle time and integer multipliers policies are adapted as a replenishment policy for synchronization throughout the supply chain. Since the problem is NP-hard, we propose a hybrid metaheuristic algorithm (HGTSA) which combines a genetic algorithm with tabu search to find the solution in large-scale problems. The results of computational experiments demonstrate the efficiency of

Table 3: Configuration of instance problems

#	Supply chain configuration	n_s	n_f	Number of variables	Number of constraints
1	1×1	5	5	50,778	17,883
2	2×2	4,2	5,2	39,610	14,754
3	3×2	6,1,2	8,7	464,114	106,262
4	3×3	3,2,3	2,3,2	592,103	137,933
5	4×3	3,5,3,2	3,7,3	385,062	102,129
6	4×4	6,7,4,8	4,6,8,9	1,554,407	314,002
7	5×5	9,3,5,6,4	6,7,3,2,8	1,802,680	380,437
8	6×4	7,5,8,9,10,2	2,3,6,7	1,268,139	265,533
9	7×6	4,6,8,5,9,1,4	3,5,2,1,8,6	2,850,329	520,895
10	8×8	4,5,8,6,4,3,5,8	4,6,1,2,4,7,8,4	2,110,901	422,022
11	10×3	3,5,3,6,9,1,2,2,1,3	14,12,19	4,721,938	921,150
12	6×7	6,18,5,9,11,4	6,8,17,5,6,12,3	2,138,363	440,825
13	10×10	7,5,3,6,9,1,2,2,1,3	9,5,8,9,1,2,9,3,5,7	5,576,079	1,069,616

Table 4:

Instance	Common cycle time policy	Integer multipliers policy				Comparison between common cycle time and Integer multipliers policy (% cost saving)
	RPD for HGTSA (%)	RPD for GA (%)	RPD for HGTSA (%)	Average CPU time for GA	Average CPU time for HGTSA	
1	13.56	19.34	16.34	0.26 min	0.47 min	6.01
2	15.23	19.78	12.14	0.31 min	0.42 min	7.23
3	18.98	21.03	15.89	2.03 min	6.19 min	12.08
4	189.45	25.18	20.08	4.54 min	8.33 min	1.16
5	22.34	29.56	19.23	1.35 min	4.66 min	14.29
6	29.56	30.78	23.67	21.07 min	45.43 min	17.89
7	28.71	36.62	27.09	34.19 min	1.01 h	13.25
8	32.66	41.99	31.89	15.12 min	31.73 min	9.08
9	37.89	44.74	34.29	1.09 h	2.32 h	14.07
10	42.11	48.67	34.53	59.48 min	2.16 h	11.65
11	44.80	50.56	37.72	2.29 h	4.99 h	7.41
12	50.73	57.61	39.23	42.89 min	1.20 h	13.06
13	53.91	59.07	43.88	2.66 h	4.62 h	15.69

the proposed algorithm. A cost reduction up to 17.89% can be accomplished by applying the integer multipliers policy rather than the common cycle time policy to synchronize the supply chain.

Several extensions to the research presented in this paper could be conducted. The integer power of two multiplier mechanism could be investigated to synchronize the supply chain. The proposed inventory model representing the integer multiplier mechanism can be solved by a more efficient approach. Developing another optimization algorithm to solve this model, instead of metaheuristic will certainly shorten the solution time and facilitate handling larger supply chain configurations.

REFERENCES

Clark, A.J. and Scarf, H., 1960. Optimal policies for a multi-echelon inventory problem. *Management science*, 6(4), 475-490.

Clausen, J., Ju, S., 2006. A hybrid algorithm for solving the economic lot and delivery scheduling problem in the common cycle case. *European Journal of Operational Research* 175 (2), 1141–1150.

Cheng, R. and Gen, M., 1997. Parallel machine scheduling problems using memetic algorithms. *Computers & Industrial Engineering*, 33(3), 761-764.

Hahm, J., Yano, C.A., 1992. The economic lot and delivery scheduling problem: the single item case. *International Journal of Production Economics* 28 (2), 235–252.

Hahm, J., Yano, A.C., 1995. Economic lot and delivery scheduling problem: the common cycle case.*IIE Transactions* 27 (2), 113–125.

Holland, J. H., 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Iyer, S.K. and Saxena, B., 2004. Improved genetic algorithm for the permutation flowshop scheduling problem. *Computers & Operations Research*, 31(4), 593-606.

Jensen, M.T., Khouja, M., 2004. An optimal polynomial time algorithm for the common cycle economic lot and delivery scheduling problem. *European Journal of Operational Research* 156 (2), 305–311.

Khouja, M., 2000. The economic lot and delivery scheduling problem: common cycle, rework, and variable production rate. IIE Transactions 32 (8), 715–725.

Khouja, M., 2003. Optimizing inventory decisions in a multi-stage multi-customer supply chain. *Transportation Research Part E: Logistics and Transportation Review*, 39(3),193-208.

Naso, D., Surico, M., Turchiano, B. and Kaymak, U., 2007. Genetic algorithms for supply-chain scheduling: A case study in the distribution of ready-mixed concrete. *European Journal of Operational Research*, 177(3), 2069-2099.

Nikandish, N., Eshghi, K., Torabi, S.A., 2009. Integrated procurement, production and delivery scheduling in a generalized three stage supply chain. *Journal of Industrial and Systems Engineering* 3 (3), 189–212.

Osman, H., &Demirli, K., 2012. Economic lot and delivery scheduling problem for multi-stage supply chains. International *Journal of Production Economics* 136(2), 275-286.

Torabi, S.A., Ghomi, S.F. and Karimi, B., 2006. A hybrid genetic algorithm for the finite horizon economic lot and delivery scheduling in supply chains. *European Journal of Operational Research*, 173(1), 173-189.

Wardono, B. and Fathi, Y., 2004. A tabu search algorithm for the multi-stage parallel machine problem with limited buffer capacities. *European Journal of Operational Research*, 155(2), 380-401.

Wang, H.F. and Wu, K.Y., 2004. Hybrid genetic algorithm for optimization problems with permutation property. *Computers & Operations Research*, 31(14), 2453-2471.

How to cite this article: Kia H, Ghodsypour SH. Comparing Two Replenishment Policy for Solving Economic Lot and Delivery Scheduling Problems in a Three-echelon Supply Chain. Int J Sci Stud 2017;5(4):854-860.

Source of Support: Nil, **Conflict of Interest:** None declared.