Int J Adv Manuf Technol (2014) 71:365–376
DOI 10.1007/s00170-013-5476-x

ORIGINAL ARTICLE

Optimization of vendor managed inventory of multiproduct EPQ model with multiple constraints using genetic algorithm

Seyed Hamid Reza Pasandideh · Seyed Taghi Akhavan Niaki · Mohammad Hemmati Far

Received: 20 November 2010 / Accepted: 4 November 2013 / Published online: 27 November 2013

Abstract The aim of this paper is to investigate the vendor managed inventory (VMI) problem of a single-vendor single-buyer supply chain system, in which the vendor is responsible to manage the buyer's inventory. To include an extended applicability in real-world environments, the multiproduct economic production quantity model with backordering under three constraints of storage capacity, number of orders, and available budget is considered. The nonlinear programming model of the problem is first developed to determine the near optimal order quantities along with the maximum backorder levels of the products in a cycle such that the total VMI inventory cost of the system is minimized. Then, a genetic algorithm (GA) based heuristic is proposed to solve the model. Numerical examples are given to both demonstrate the applicability of the proposed methodology and to fine tune the GA parameters. At the end, the performance of the proposed GA is compared to the one of the LINGO software using different problem sizes. The results of the comparison study show that, while the solutions do not differ significantly, the proposed GA reaches near optimum solutions in significantly less amount of CPU time.

S. H. R. Pasandideh
Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran
e-mail: shr_pasandideh@khu.ac.ir

S. T. A. Niaki (✉)
Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran
e-mail: Niaki@Sharif.edu

M. Hemmati Far
Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
e-mail: m.hemmatifar@gmail.com

Keywords Vendor managed inventory · Economic production quantity · Multiproduct · Limited storage · Limited budget · Limited number of orders · Genetic algorithm

1 Introduction and literature review

Satisfying customers' demand is one of the keys to the success of companies. In supply-chain management (SCM), a series of organizations integrate and cooperate in order to improve the competitive capabilities of the whole chain [10]. Business paradigm has recently changed tremendously. Individual businesses no longer compete as solely autonomous entities, but rather work together as a supply chain. Perhaps, this is one of the main reasons researchers and managers pay such significant attention to the business integration [16]. Due to globalization and increasing competition, increasing attention is given to supply chain integration [27].

Since the single-vendor single-buyer supply chain problem is the building block for wider supply chains, it has received an increasing attention in recent years. The global supply chain can be very complex and link-by-link understanding of joint policies can be very useful [2]. In the single-vendor single-buyer problem, the vendor manufactures a product in lots and delivers the produced lot to a buyer in number of shipments. The objective of this model is to determine the production lot size and shipments schedule that minimize the total cost of the vendor–buyer system [7].

One of the well-known concepts in SCM is the vendor-managed inventory (VMI) (see, e.g., [6, 8]) and many successful businesses such as Wal-Mart and JC Penney have demonstrated the benefits of VMI [4, 9]. Within the VMI model, the buyer provides the vendor with information on its sales and inventory level and the vendor determine the replenishment quantity at each period based on the information. Throughout

the VMI model, the vendor can set up efficient replenishment plans, while the buyer can receive appropriate amounts of replenishment on time [14, 17]. VMI has some advantages for both parties. Customer service levels may increase in terms of the reliability of product availability because the vendor can use the information collected on the inventory levels at the buyer to better anticipate future demand [15].

Magee [18] when discussing who should have authority over the control of inventories described an early conceptual framework of VMI. However, interest in the concept has only really developed during the 1990s. Dong and Xu [9] presented an analytical model to evaluate the short- and long-term impact of VMI on supply chain profitability by analyzing the inventory systems of the parties involved. Yao et al. [32] using the same assumptions as Dong and Xu's [9] presented an analytical model to determine how key logistics parameters, most notably ordering costs and inventory carrying charges, can affect the benefits to be derived from VMI. However, they assumed the order quantity for the supplier was an integer multiple of the buyer's replenishment quantity. Van Der Vlist et al. [31] extended the Yao et al. [32] model along with the costs of shipments from the supplier to the buyer. Two situations of no-VMI and VMI were modeled in their research. Sofifard et al. [29] presented an analytical model for a single-buyer single-supplier model to explore the effects of collaborative supply-chain initiatives such as VMI with the economic production quantity (EPQ) manner.

In two-echelon single-vendor multiple-buyers supply chain model under VMI mode of operation, Jasemi [13] developed a supply chain model with single-supplier n-buyers, where he compared the VMI system with the traditional types. He also made a pricing system for profit sharing between parties. Furthermore, Nachiappan and Jawahar [23] proposed a nonlinear integer-programming model with a genetic algorithm (GA) based heuristic to find the optimal sales quantity of each buyer.

Farahani and Elahipanah [10] developed a new model for a distribution network in a three-echelon supply chain, which not only minimizes the total costs but also follows just-in-time distribution purposes in order to better represent the real-world situations. In their research, a GA was designed to find the Pareto fronts of the large-size problem instances of the multiobjective mixed-integer linear-programming problem.

In this paper, in order to determine the products' near optimal order quantities and the maximum backorder levels in a cycle such that the total inventory costs of the VMI system is minimized, a new mathematical model is first developed for the single-vendor single-buyer supply chain problem. In this problem, there are multiproducts, the EPQ model is utilized with finite production rate, and the shortage is allowed and backordered. The constraints are storage capacity, number of orders, and the available budget. Since the model of the problem becomes nonlinear, a GA-based heuristic is then proposed to solve it. Numerical examples are presented to demonstrate the application of the proposed methodology, to find the significant parameters of the proposed GA, and to tune the parameters accordingly. To do this, the backward elimination method of the SAS 9.1 computer software with a quadratic regression function is first used to find the significant parameters. Then, the LINGO 8.0 software is employed to solve the regression model and to find the optimal value of the proposed GA parameters. Finally, the tuned GA will be used in MATLAB 7.6.0.324 software to find the near optimal order quantities and the maximum backorder level of the products.

In short, the highlights of the differences of this research with the above studies are as follow:

- Incorporating several products along with shortages to the VMI problem
- Adding additional constraints to the VMI problem to make it more realistic
- Proposing a new modeling to the VMI problem
- Employing a meta-heuristic algorithm to solve the new VMI model
- Calibrating the parameters of the proposed meta-heuristic algorithm to obtain better near optimum solutions.

The rest of the paper is organized as follows: In Section 2, the problem is defined in more details. Section 3 is dedicated to the mathematical formulation of the problem. The proposed GA is developed in Section 4. The test problems, the parameters tuning, and the computational results are discussed in Section 5. Finally, in Section 6, conclusions are provided and some areas of further research are proposed.

2 The problem definition and assumptions

In a supply chain without VMI, the vendor observes consumer demand only indirectly through the buyer's ordering policy. In fact, the buyer appears to be the "leader" in this relationship and the vendor just takes the order quantity from the buyer and makes the necessary delivery, not having any responsibility for the production holding. Now two parties decide to adopt a VMI system, e.g., the buyer no longer manages its inventory system and leaves it to the vendor to determine inventory levels, order quantities, lead times, etc. In a supply chain with VMI, the vendor's information system directly receives consumer demand data. As a result, the vendor has now the combined inventory with order setup and holding cost [9]. The vendor with regard to his own inventory cost that equals to the total cost of the supply chain determines the timing and the quantity of production in every cycle. The major difference between not using and using VMI is that the vendor determines the buyer's order quantity in a VMI system [32].

The problem at hand arises from a single-buyer single-vendor inventory control environment that uses the VMI system. In this problem, there are several products and the EPQ model is utilized with practical instances of finite production rate, backorders, limited warehouse space, limited number of orders, and limited budget. Moreover, compared to programming cycles that can be months or year, the lead time that is less than a day can be neglected and that the selling prices are constant during the programming horizon. The objective is to find the products' order quantities and their maximum backorder levels per cycle such that the total VMI inventory cost is minimized while the constraints are satisfied.

In short, the following assumptions are used for the mathematical formulation of the problem:

(a) There is a single-vendor single-buyer supply chain based on vendor–buyer's perspective.
(b) There are n products.
(c) The planning horizon is infinite.
(d) For each product, shortage is allowed and backordered $(\widehat{\pi} \neq 0 \text{ and } \pi = 0)$.
(e) Deliveries of the orders are assumed instantaneous, that is, the lead time is zero.
(f) The selling prices of all products in the planning period are fixed (the quantity discount is not allowed).
(g) The production rate for all products is continuous and finite (EPQ model), where in each cycle we have T_{P_j} (period of production) and T_{D_j} (period of idleness without any production).
(h) The costumer's demand rate for all products is known and constant.
(i) The vendor's storage capacity for all products is limited (not more than F).
(j) The total available budget of the system is limited (not more than X).
(k) The total number of orders for all products is limited (not more than M).

3 Mathematical model

The following notations are used to model the problem at hand:

3.1 Notations

For $j=1,2,\ldots,n$, define the parameters and variables of the model as:

Parameters

A_{Vj} The vendor's fixed ordering cost per order of the jth product
A_{Bj} The buyer's fixed ordering cost per order of the jth product
D_j The buyer's demand rate of product j in a period
P_j The production rate of product j in each period
π The fixed backorder cost per unit (not depending on the time)
$\widehat{\pi}$ The fixed backorder cost per unit per time unit
h_{Bj} The holding cost of product j per unit held in the buyer's store in a period ($h_{Bj}=iC_j$)
i The fixed interest rate (rate of the holding cost that is not dependent on the time)
C_j The buyer's procurement cost per unit of product j
f_j Space occupied by each unit of product j
F The vendor's available storage capacity for all products
M The total number of orders for all products in each cycle
X The total available budget in each cycle
n The number of products

Variables

Q_j The order quantity of product j in a cycle
b_j The maximum backorder level of product j in a cycle
$\mathrm{TC_{VMI}}$ The total inventory costs of the VMI supply chain
$\mathrm{KB_{no\ VMI}}$ The buyer's inventory cost before utilizing the VMI system
$\mathrm{KB_{VMI}}$ The buyer's inventory cost after utilizing the VMI system
$\mathrm{KV_{no\ VMI}}$ The vendor's inventory cost before VMI
$\mathrm{KV_{VMI}}$ The vendor's inventory cost after VMI

The inventory graph of the problem at hand is similar to the one of the EPQ model and is given in Fig. 1, where T_{P_j} denotes period of production and T_{D_j} indicates period of idleness (without any production). In this graph, at the start of the first cycle (time=0), both the inventory and the backorder are assumed zero.

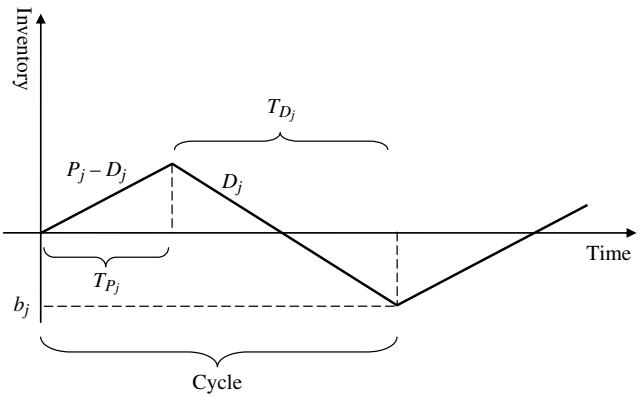

Fig. 1 The inventory graph of the problem (EPQ with shortage)

3.2 The inventory cost of the non-VMI supply chain

Referring to Cardenas-Barron [3], the inventory costs of the buyer and the vendor before implementation of the VMI system is calculated as follows:

$$\mathrm{KB}_{\mathrm{noVMI}} = \sum_{j=1}^{n} \left(\frac{D_j}{Q_j} A_{B_j} + \frac{\widehat{\pi} + h_{B_j}}{2\rho_j Q_j} b_j^2 - h_{B_j} b_j + \frac{h_{B_j}\rho_j}{2} Q_j \right) \quad (1)$$

$$\mathrm{KV}_{\mathrm{noVMI}} = \sum_{j=1}^{n} \frac{D_j}{Q_j} A_{V_j} \quad (2)$$

where $\rho_j = \left(1 - \frac{D_j}{P_j}\right)$.

We note that the near optimal values of the order quantities (Q_j) and the backorder levels (b_j) for the non-VMI chain are determined by the buyer using Eq. (1) and there is no role for the vendor. The total cost of the non-VMI supply chain is obtained by summation of the inventory costs of the buyer and the vendor. While the first term in Eq. (1) shows the ordering cost, the other terms refer to shortage and holding costs. Note that the ordering cost is assumed proportional to the number of orders and that the holding and shortage costs are obtained based on the areas under the inventory and shortage in Fig. (1). Besides, Eq. (2) models the ordering cost involved in vendor's inventory.

3.3 The inventory cost the VMI supply chain

After the implementation of VMI, the inventory costs of both the buyer and the vendor, and hence the total inventory costs of the integrated supply chain, are calculated as follows:

$$\mathrm{KB}_{\mathrm{VMI}} = 0 \quad (3)$$

$$\mathrm{KV}_{\mathrm{VMI}} = \sum_{j=1}^{n} \frac{D_j}{Q_j} A_{V_j} + \sum_{j=1}^{n} \left(\frac{A_{B_j} D_j}{Q_j} + \frac{\widehat{\pi} + h_{B_j}}{2\rho_j Q_j} b_j^2 - h_{B_j} b_j + \frac{h_{B_j}\rho_j}{2} Q_j \right) \quad (4)$$

$$\mathrm{TC}_{\mathrm{VMI}} = \mathrm{KB}_{\mathrm{VMI}} + \mathrm{KV}_{\mathrm{VMI}} = \sum_{j=1}^{n} \left[\frac{D_j}{Q_j} (A_{V_j} + A_{B_j}) + \frac{\widehat{\pi} + h_{B_j}}{2\rho_j Q_j} b_j^2 - h_{B_j} b_j + \frac{h_{B_j}\rho_j}{2} Q_j \right] \quad (5)$$

Equations (3), (4), and (5) have a theoretical contribution in a sense that after the VMI implementation, the buyer no longer manages its inventory and leaves it to the vendor to determine inventory levels, order quantities, and lead time. Note that the explanations made on the derivation of the terms in Eq. (1) can also be used for the terms in Eq. (5).

Now, the goal is to determine the values of the production order quantities and the maximum backorder level (determined by the vendor) in a cycle such that the total cost of the supply chain under VMI system [given in Eq. (5)] is minimized while the constraints are fulfilled. The constraints are:

1. The capacity of the vendor's warehouse space to store the items is limited.
2. The total number of order for all items is limited.
3. The total available budget is limited.

Incorporating the constraints, Eq. (5) becomes

$$\text{Min } \mathrm{TC}_{\mathrm{VMI}} = \sum_{j=1}^{n} \left(\frac{D_j}{Q_j} (A_{V_j} + A_{B_j}) + \frac{\widehat{\pi} + h_{B_j}}{2\rho_j Q_j} b_j^2 - h_{B_j} b_j + \frac{h_{B_j}\rho_j}{2} Q_j \right)$$

s.t.

$$\sum_{j=1}^{n} \rho_j f_j Q_j \le F$$

$$\sum_{j=1}^{n} \frac{D_j}{Q_j} \le M$$

$$\sum_{j=1}^{n} C_j Q_j \le X$$

$$h_{Bj} = iC_j$$

$$\rho_j = \left(1 - \frac{D_j}{P_j}\right)$$

$$Q_j, b_j \ge 0 \quad ; \quad j = 1, \ldots, n \quad (6)$$

In the next section, we will present an algorithm to solve model (6).

4 A solution algorithm

The formulation given in Eq. (6) is a nonlinear-programming model. The nonlinear programming characteristic causes the model to be adequately hard to solve by exact methods [11]. Accordingly, a meta-heuristic search algorithm is needed to solve the model.

Over the last 30 years, there has been a growing interest in problem solving systems based on the principles of evolution and heredity. One type of evolutionary systems is the GA; a random evolutionary search algorithm that mimics the principles of natural genetics. GA, introduced by Holland [12], works differently compared to the classical search

	1	2	…	n
Q	[Q_1	Q_1	…	Q_n]
b	[b_1	b_1	…	b_n]

Fig. 2 A typical chromosome

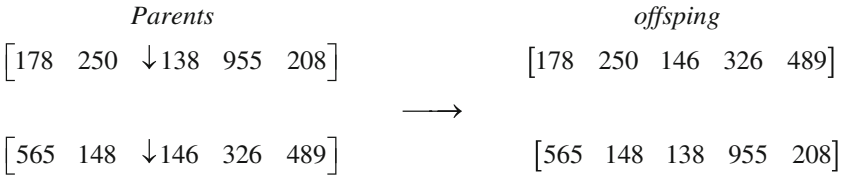

Fig. 3 An example of the crossover operation

and optimization methods. Due to its broad applicability, ease of use, and global perspective, GA has been increasingly applied to various search and optimization problems.

Recently, GA has been receiving great attention, and it has successfully been applied to other problems in the supply chain environment (see, for example, [5,24–26,30]). Since genetic algorithm has been successful in solving models similar to the model in Eq. (6) [11], it will be utilized to solve it in the following subsections.

4.1 GA algorithm in general and initial conditions

In general, a real-coded GA algorithm works in the following steps:

1. Initialization
   - 1-1. Parameter setting (the probabilities of crossover and mutation operations, P_C,P_m, stopping criterion, population size, selection strategy, crossover operator, mutation operator, and number of generation)
   - 1-2. Initialize population (randomly)
2. Fitness evaluation
   Repeat
3. Individual selection for mating pool (size of mating pool= population size)
4. Crossover operation (for each consecutive pair apply crossover with probability P_C). We use the roulette wheel selection for this operation.
5. Mutate children (for each new generation apply mutation with probability P_m)
6. Replace the current population by the resulting mating pool
7. Fitness evaluation (determining the minimum of the total cost of the VMI supply chain) until a stopping criterion is met.

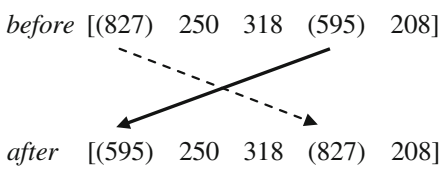

Fig. 4 An example of the swapping mutation operation

Furthermore, the required initial information to start the GA is:

1. Population size: It is the number of the chromosomes or scenarios that we will keep in each generation, denoted by NC.
2. Crossover rate: This is the probability of performing a crossover in the GA method, denoted by P_C.
3. Mutation rate: This is the probability of performing mutation in the GA method denoted by P_m.

4.2 Chromosome

GA is known as a problem-independent approach; however, the chromosome representation is one of the critical issues when applying it to optimization problems.

Table 1 The input data for the test problems

Item(j)	D_j	P_j	A_{V_j}	A_{B_j}	C_j	f_j
1	420	430	1	3	13	3
2	360	360	2	2	30	2
3	540	550	3	1	23	3
4	390	400	5	4	6	1
5	480	480	2	2	13	4
6	510	530	4	2	20	3
7	530	535	1	3	16	2
8	380	385	2	1	10	1
9	430	435	3	4	6	3
10	580	585	4	2	26	4
11	420	430	1	3	13	3
12	360	360	2	2	30	2
13	540	550	3	1	23	3
14	390	400	5	4	6	1
15	480	490	2	2	13	4
16	510	530	4	2	20	3
17	530	535	1	3	16	2
18	380	390	2	1	10	1
19	430	440	3	4	6	3
20	580	590	4	2	26	4

$F = 100,000, M = 60, X = 470,000, i = 0.3, \pi = 0, \widehat{\pi} = 3$

In the proposed GA method, a matrix of two rows and n columns represents a chromosome. The elements of the first row show the order quantities of the products and the elements of the second row indicate the maximum backorder levels of the products. Figure 2 shows the general form of a chromosome.

4.3 Evaluation

In a GA method, as soon as a chromosome is generated, a fitness value is needed to be assigned for it. In an optimization problem, the fitness value is determined by evaluating the objective function based on the given elements of the chromosome. Since there are limits on the storage capacity, total number of orders, and budget in the model of the problem given in Eq. (6), some generated chromosomes may not be feasible. In order to control infeasible solutions, although there are different ways such as the penalty policy [11], since the solution for model (6) requires a large-size population, in this research, an infeasible chromosome will be removed from the pool as soon as generated.

4.4 Initial population

In this stage, a collection of chromosomes is randomly generated.

4.5 Crossover

In a crossover operation, it is necessary to mate pairs of chromosomes to create offspring. We perform this by selecting a pair of chromosomes from the generation randomly with probability P_C. There are many different types of crossover operators such as one-, two-, multiple-point, uniform, linear, blend, or simulated binary codes. In this research, based on a pilot study using trial and error, the following one-point crossover has been found the most effective operator:

(a) Choose a random point
(b) Split parents at the selected crossover point
(c) Create children by exchanging tails

Figure 3 shows a graphical representation of the crossover operation for the order quantity row of the chromosome

Table 2 The initial results for small-size test problems by GA

Index of test problems	P_C	P_m	NC	n	Fitness (minimum)
1	0.45	0.005	70	3	6,519.47
2	0.49	0.009	80	3	6,340.51
3	0.52	0.013	90	3	6,152.14
4	0.56	0.017	100	3	5,338.34
5	0.59	0.021	110	3	8,627.28
6	0.63	0.025	120	3	7,206.43
7	0.66	0.029	130	3	7,188.82
8	0.70	0.033	140	3	6,422.51
9	0.73	0.037	150	3	7,311.14
10	0.77	0.041	160	3	8,830.32
11	0.80	0.045	170	3	6,913.98
12	0.84	0.049	180	3	8,158.60
13	0.45	0.005	70	5	5,585.51
14	0.49	0.009	80	5	8,081.41
15	0.52	0.013	90	5	7,482.84
16	0.56	0.017	100	5	7,938.98
17	0.59	0.021	110	5	5,472.17
18	0.63	0.025	120	5	6,198.48
18	0.66	0.029	130	5	7,593.73
20	0.70	0.033	140	5	7,807.91
21	0.73	0.037	150	5	5,941.89
22	0.77	0.041	160	5	6,751.67
23	0.80	0.045	170	5	7,471.12
24	0.84	0.049	180	5	6,463.37

Table 3 The initial results for medium-size test problems by GA

Index of test problems	P_C	P_m	NC	n	Fitness (minimum)
1	0.45	0.005	70	8	10,585.76
2	0.49	0.009	80	8	13,463.06
3	0.52	0.013	90	8	10,087.05
4	0.56	0.017	100	8	8,984.82
5	0.59	0.021	110	8	12,098.35
6	0.63	0.025	120	8	9,420.12
7	0.66	0.029	130	8	19,109.75
8	0.70	0.033	140	8	11,325.97
9	0.73	0.037	150	8	10,350.13
10	0.77	0.041	160	8	17,159.48
11	0.80	0.045	170	8	20,131.62
12	0.84	0.049	180	8	19,319.53
13	0.45	0.005	70	10	10,168.09
14	0.49	0.009	80	10	17,258.73
15	0.52	0.013	90	10	18,998.75
16	0.56	0.017	100	10	19,076.90
17	0.59	0.021	110	10	10,132.49
18	0.63	0.025	120	10	16,532.39
18	0.66	0.029	130	10	16,796.93
20	0.70	0.033	140	10	9,504.57
21	0.73	0.037	150	10	26,325.27
22	0.77	0.041	160	10	16,738.38
23	0.80	0.045	170	10	14,947.45
24	0.84	0.049	180	10	22,029.88

matrix when $n=5$. The exchanged tails in the offspring are shown in parentheses. A similar crossover operator can be depicted for the maximum backorder level as well.

4.6 Mutation

Mutation is the second operation in the GA methods for exploring new solutions. In this operation, a chromosome of the generation is first randomly selected, and then is used for the mutation. While there are other types of mutation like the random mutation [19], in this research, the swap operator, in which the places of two randomly selected genes are exchanged, is used [1]. Figure 4 shows a graphical representation of the mutation operation by swapping when $n=5$. The randomly selected genes are shown within parentheses.

4.7 Chromosome selection

In genetic algorithm, the selection operator is used to guide the search process towards more promising regions in a search space. In this research, the roulette wheel procedure is employed to select the chromosomes, in which better solutions get higher chance to become parents of the next generation solutions. This selection is based on the fitness value of each chromosome. We select NC chromosomes among the parents and offspring with the best fitness values.

Table 4 The initial results for large-size test problems by GA

Index of test problems	P_C	P_m	NC	n	Fitness (minimum)
1	0.45	0.005	70	17	22,200.67
2	0.49	0.009	80	17	20,594.46
3	0.52	0.013	90	17	30,476.86
4	0.56	0.017	100	17	22,342.36
5	0.59	0.021	110	17	39,177.78
6	0.63	0.025	120	17	25,021.81
7	0.66	0.029	130	17	31,293.48
8	0.70	0.033	140	17	16,103.66
9	0.73	0.037	150	17	17,275.84
10	0.77	0.041	160	17	18,559.15
11	0.80	0.045	170	17	26,800.53
12	0.84	0.049	180	17	35,142.35
13	0.45	0.005	70	20	26,738.04
14	0.49	0.009	80	20	29,971.93
15	0.52	0.013	90	20	21,559.58
16	0.56	0.017	100	20	67,837.56
17	0.59	0.021	110	20	39,581.15
18	0.63	0.025	120	20	22,249.81
18	0.66	0.029	130	20	14,725.20
20	0.70	0.033	140	20	35,413.43
21	0.73	0.037	150	20	40,720.66
22	0.77	0.041	160	20	20,989.57
23	0.80	0.045	170	20	42,879.01
24	0.84	0.049	180	20	61,013.43

Table 5 Summarized GA output for three problem sizes (the minimum fitness values)

	P_C	P_m	NC	n	Fitness
Small	0.56	0.017	100	3	5,338.34
	0.59	0.021	110	5	5,472.17
Medium	0.56	0.017	100	8	8,984.82
	0.70	0.033	140	10	9,504.57
Large	0.70	0.033	140	17	16,103.66
	0.66	0.029	130	20	14,725.20

4.8 Stopping criterion

The last step in the methodology is to check if the method has found a solution that is good enough to meet the user's expectations. Stopping criteria is a set of conditions such that when the method satisfies them, a good solution is obtained. In this research, we stop after 600 generations. We note that the number of generations depends on the problem size.

Table 6 Summarized SAS output (backward elimination) for three problem sizes

Size of problems	Variable	Estimate
Small ($n=3$ and 5)	Intercept	−605,707
	x_1	1,485,775
	x_4	148,509
	x_1x_2	−5,949,2635
	x_1x_4	−359,969
	x_2x_3	205,026
	$x_1x_2x_4$	1,435,0216
	$x_2x_3x_4$	−49,368
Medium ($n=8$ and 10)	Intercept	3,558.13261
	x_1x_4	1,974.95182
Large ($n=17$ and 20)	Intercept	−30,633
	x_4	−31,898
	x_1x_4	84,055
	$x_1x_2x_4$	−1,694,569
	$x_1x_2x_3x_4$	4,748.21409

Table 7 The results of solving the regression functions by LINGO

The size of problems	GA parameters and variable	Optimal value
Small (n=3 and 5)	x_1	0.8500000
	x_2	0.5000000E−01
	x_3	70.00000
	x_4	3.000000
Medium (n=8 and 10)	x_1	0.4500000
	x_2	0.5000000E−02
	x_3	70.00000
	x_4	8.000000
Large (n=17 and 20)	x_1	0.4500000
	x_2	0.5000000E−01
	x_3	70.00000
	x_4	20.00000

5 Numerical examples

In order to demonstrate the application of the proposed methodology and to study its performances, some numerical examples as test problems are given in this section. Based on the size and hence the required computer CPU time, these examples are classified into three categories of small, medium, and large. The input data of the numerical examples for small (n=3 and 5 products), medium (n=8 and 10 products), and large-size (n=17 and 20 products) test problems are given in Table 1. For all products of these problems $\widehat{\pi}$ and π are assumed 3 and 0, respectively. Furthermore, the total available budget is 470,000, the total available warehouse space is 100,000, a maximum of 60 orders can be placed, and the interest rate for holding the items is 0.3.

The GA to solve Eq. 6 was coded in MATLAB 7.6.0.324 software. All the test problems are solved on a Pentium 4 computer with 512 MB RAM and 2.40 GHz CPU. Prior to GA implementation to obtain near optimum solutions, its parameters are first calibrated. Similar to Pasandideh et al. [25, 26], the empirical optimization of tuning GA parameters of this research is made of two main steps. First, a significant relationship between the fitness function and the GA parameters is estimated using a regression approach. Second, the optimal values of the parameters are found by solving a constrained optimization model that involves the estimated relationship. The constraints are simply the lower and the upper bounds of the parameters.

Table 9 The near optimal results for medium-size problems by GA

P_C	P_m	NC	n	Q_j	b_j	Fitness	CPU t
0.45	0.005	70	8	402.14, 822.60, 240.19, 822.60, 809.16, 105.52, 402.14, 434.17	14.41, 16.46, 17.49, 13.38, 12.35, 18.52, 18.18, 13.03	5,383.45	24.08

The steps involved in solving the numerical examples based on Eq. (6) are as follows:

- Obtain an initial solution using the MATLAB software
- Find the significant GA parameters using the backward elimination regression algorithm by SAS software
- Determine the near optimal values of the GA parameters using the NLP routine of the LINGO software
- Obtain near optimal solutions of Q_j and b_j by the parameter-tuned GA using the MATLAB software.

5.1 Obtaining an initial solution

The initial results for small, medium, and large-size test problems are summarized in Tables 2, 3, and 4, respectively. Table 5 shows the summarized output for the three problem classes, where NC denotes the number of chromosomes and the fitness function values are recorded as the minimum fitness obtained by 10 times running of the developed GA.

5.2 Finding the significant GA parameters

Applied researchers frequently use automated variable selection methods to identify significant independent predictors of an outcome or for developing parsimonious regression models [20]. Several methods may be used in selecting the appropriate subset of variables for a regression model involving multiple independent variables. In these methods, the lengthy and cumbersome procedure of utilizing the “all possible regressions” is usually avoided. Instead, stepwise-type procedures are followed, which involve evaluating a small subset of regression models by adding, deleting, or simultaneously adding and deleting regressors one at a time [22]. These methods are known as forward selection, backward

Table 8 The near optimal results for small-size problems by GA

P_C	P_m	NC	n	Q_j	b_j	Fitness	CPU t
0.85	0.05	70	3	202.98, 829.28, 202.98	13.51, 20.26, 15.76	2,223.48	22.71

elimination, and stepwise regression methods, respectively. The backward elimination algorithm is often less adversely affected by correlations among the regressors than are other methods [22]. The forward selection method is generally used to provide initial screening of a large number of independent and important variables, especially when multicollinearity is considered as a potential problem. Whereas when modest-sized set of independent variables are used backward elimination procedure is more efficient and helps to fine tune the model and may result in overall high R^2 value [28].

In this research, the GA parameters are P_C, P_m, and NC, which are represented by x_1, x_2, and x_3 for convenience. Moreover, since the number of products (n) affects both the quality of the near optimum solution and the required CPU time, another variable (x_4 for convenience) is considered in finding the near optimal solution. Then, the backward elimination procedure of the SAS 9.1 software using a quadratic regression model was employed to find the significant parameters of the GA method. A sample of the quadratic regression function along with the variables used in the problems (by Lingo software) is as follows.

The quadratic regression function is:

```
Y= ß0 + ß1x1 + ß2x2 + ß3x3 + ß4x4 + ß5(X1*X2) + ß6(X1*X3) + ß7(X1*X4)+ ß8(X2*X3) +
ß9(X2*X4) + ß10(X3*X4) + ß11(X1*X2*X3) + ß12(X1*X2*X4) + ß13(X13=X1*X3*X4) +
ß14(X2*X3*X4)+ ß15(X1*X2*X3*X4) + ß16(x1*x1) + ß17(x2*x2) + ß18(x3*x3) + ß19(x4*x4)
```
(7)

The SAS variables are:

```
X5=X1*X2;
X6=X1*X3;
X7=X1*X4;
X8=X2*X3;
X9=X2*X4;
X10=X3*X4;
X11=X1*X2*X3;
X12=X1*X2*X4;
X13=X1*X3*X4;
X14=X2*X3*X4;
X15=X1*X2*X3*X4;
x16=x1*x1;
x17=x2*x2;
x18=x3*x3;
x19=x4*x4;
```

Where Y denote the estimated fitness function and β_i (i=0–19) denotes the estimated coefficients, respectively.

The initial results of GA (Tables 2, 3, and 4) were used as input data for the SAS software. After running the backward elimination procedure for each test problem sizes, significant GA parameters were determined and are summarized in Table 6. Based on the results of this table, the estimated regression functions are

$$Y_{\text{small}} = -605,707 + 1,485,775x1 + 148,509x4 - 59,492,635x1x2 - 359,969x1x4 + 205,026x2x3 + 14,350,216x1x2x4 - 49,368x2x3x4 \quad (8)$$

$$Y_{\text{medium}} = 3,558.13261 + 1,974.95182x1x4 \quad (9)$$

$$Y_{\text{large}} = -30,633 - 31,898x_4 + 84,055x1x4 - 1,694,569x1x2x4 + 4,748.21409x1x2x3x4 \quad (10)$$

where Y_{small}, Y_{medium}, and Y_{large} denote the estimated fitness functions of the small, the medium, and the large-size problems, respectively.

5.3 Determining the optimal values of the GA parameters

The decision variables of the optimization models in Eqs. (8), (9), and (10) are the parameters of GA. In a typical GA procedure, the ranges of the parameters are usually assumed to be $0.45 \le x1 \le 0.85$, $0.005 \le x_2 \le 0.05$, and $70 \le x_3 \le$

Table 10 The near optimal results for large-size problems by GA

P_C	P_m	NC	n	Q_j	b_j	Fitness	CPU t
0.45	0.05	70	20	479.68, 627.16, 329.75, 329.75, 1,839.90, 329.75, 479.68, 329.75, 1,186.73, 798.91, 798.91, 627.16, 1,275.02, 329.75, 1,979.50, 479.68, 1,401.77, 627.16, 627.16, 479.68	21.95, 28.71, 30.40, 20.26, 21.39, 27.02, 29.83, 24.21, 24.21, 32.65, 23.64, 30.40, 27.02, 21.95, 29.83, 28.71, 21.39, 20.26, 23.64, 32.65	11,982.71	44.23

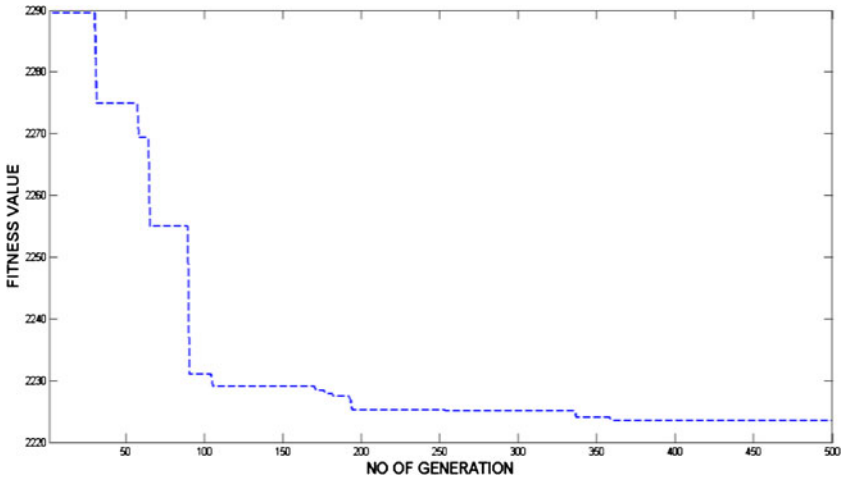

Fig. 5 The trend of the fitness value optimization for small-size problems

180 [21]. Besides, x_4 takes various ranges for different problem sizes. Based on these ranges, the LINGO 8.0 software is employed to solve the regression functions given in Eqs. (8), (9), and (10). The models along with the results for small, medium, and large-size problems along with the optimal values of the GA parameters are summarized in Table 7.

5.4 The near optimal solution

The parameter-tuned GA of Section 5.3 is used to find the near optimal values of the order quantities and the maximum backorder levels of the products in different test problems. For example, regarding what was derived in Table 7, the near optimal results for small-, medium-, and large-size problems using the MATLAB software are presented in Tables 8, 9 and 10, respectively. In order to increase the probability of finding a good near optimum solution, the product units are assumed to take continuous values. As a result, in Tables 8, 9, and 10, the near optimal values of the order quantity (Q_j) and the maximum backorder level (b_j) are fractional. In this table, "CPU t" denotes the CPU time of solving the problem in second. In this case, the decision maker can either use mathematical approach or take managerial opinion to change them into integers for practical usages. Figures 5, 6, and 7 show the trend of the fitness value optimization of the parameter-tuned GA for small-, medium-, and large-size problems, respectively.

5.4.1 The difference in fitness value (cost saving)

To examine the percentage of the fitness value reduction (cost savings) obtained by the implementation of the parameter tuning process of Section 5.4 define

$$\text{Cost saving percentage} = \frac{TC_{\text{initial}} - \text{TC}_{\text{tuned}}}{\text{TC}_{\text{initial}}} \times 100 \qquad (11)$$

where $\text{TC}_{\text{initial}}$ denotes the near optimal total inventory cost obtained by the untuned GA (Table 5) and TC_{tuned} shows the

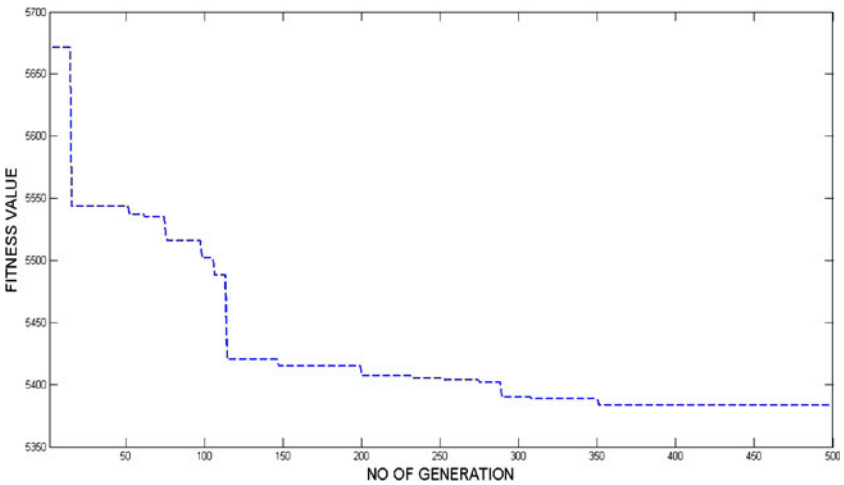

Fig. 6 The trend of the fitness value optimization for medium-size problems

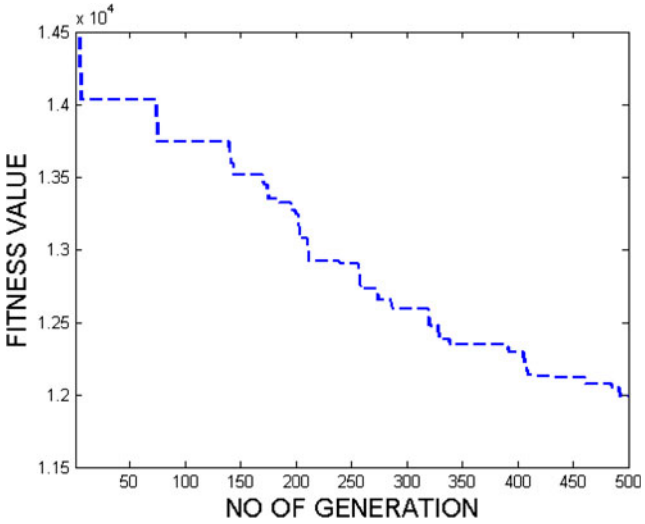

Fig. 7 The trend of the fitness value optimization for large-size problems

near optimal total inventory cost of the parameter-tuned GA (Table 10). Then, based on the results of Tables 5 and 10, for large-size problems, we have

$$\text{Cost saving(large size)} = \frac{14,725.20-11,982.71}{14,725.20} \times 100 = 18.62\% \quad (12)$$

In other words, the proposed fine-tuning process results in 18.62 % reduction in the total inventory cost of the VMI supply chain.

5.5 Justification of the obtained results

To compare the efficiency of the proposed GA in terms of both the fitness function value and CPU time of execution, the models are solved by the LINGO software as well. In each problem, 24 instances of small, medium, and large are solved by the two methods and the average of fitness value and CPU time are obtained. The results that are summarized in Table 11 show that the proposed GA is more efficient than the LINGO software. More specifically, while both procedures obtain the solutions with almost the same quality, GA requires less CPU time.

Table 11 The comparative results for test problems by GA and LINGO

	Fitness (average)		CPU t (average)	
Problem type	GA	LINGO	GA	LINGO
Small	2,207.69	2,206.98	31.42	33.42
Medium	5,202.50	5,201.66	31.53	66.31
Large	11,989.91	11,988.65	47.29	189.78

6 Conclusions and future research

In this research, the VMI system of a single-buyer single-vendor supply chain, in which there are several products, the EPQ model with finite production rate is considered, shortage is allowed in backorder case, there are limited warehouse spaces, limited budget, and limited number of orders, was first modeled and shown to be of a nonlinear programming type. The objective was to determine the order quantities and the maximum backorder levels of the products such that the total VMI inventory cost of the supply chain is minimized and the constraints are satisfied. A GA-based heuristic was then proposed to solve the developed model. Next, numerical examples were presented to demonstrate the application of the proposed methodology, to find the significant parameters of the proposed GA, and to find the optimal values of the significant parameters. Finally, the parameter-tuned GA was run to obtain the near optimal values of the decision variables. The results of the numerical example on large-size test problems showed that an impressive cost savings was obtained by tuning the parameters of the proposed GA. Furthermore, to justify the quality of the obtained results by proposed GA, some problem instances were also solved by the LINGO software. The results of the comparison study (GA with LOINGO) showed that, while the proposed method finds solutions very close to optimum, it requires much less CPU time.

For future researches in this area, we recommend the following:

(a) In addition to the storage capacity, total number of orders, and total available budget limitations, we may consider other constraints such as service rates.
(b) Other search-heuristic algorithms such as simulated annealing may also be employed to solve the nonlinear programming model of the problem. In this regards, a comparison study may be conducted to assess the effectiveness of the proposed GA.
(c) Instead of backorder assumption, one may consider the lost sale, too.
d) Some other assumptions such as nonzero lead-time can be incorporated to the problem at hand.
(e) There may be quantity discounts on the selling prices of the products.
(f) A multi-echelon supply chain in the context of VMI may be of interest to be investigated.

References

1. Altiparmak F, Gen M, Lin L, Karaoglan I (2009) A steady-state genetic algorithm for multi-product supply chain network design. Comput Ind Eng 56:521–537

2. Ben-Daya M, Darwish M, Ertogral K (2008) The joint economic lot sizing problem: review and extensions. Eur J Oper Res 185:726–742
3. Cardenas-Barron LE (2001) The economic production quantity (EPQ) with shortage derived algebraically. Int J Prod Econ 70:289–292
4. Cetinkaya S, Lee CY (2000) Stock replenishment and shipment scheduling for vendor-managed inventory systems. Manag Sci 46: 217–232
5. Chen X, Wan W, Xu X (1998) Modeling rolling batch planning as vehicle routing problem with time windows. Comput Oper Res 25: 1127–1136
6. Cheung L, Lee HL (2002) The inventory benefit of shipment coordination and stock rebalancing in a supply chain. Manag Sci 48:300–306
7. Darwish MA (2009) Economic selection of process mean for single-vendor single-buyer supply chain. Eur J Oper Res 199: 162–169
8. Disney SM, Towill DR (2003) The effect of vendor managed inventory (VMI) dynamics on the bullwhip effect in supply chains. Int J Prod Econ 85:199–215
9. Dong Y, Xu K (2002) A supply chain model of vendor managed inventory. Transp Res Part E Logist Transp Rev 38:75–95
10. Farahani RZ, Elahipanah M (2008) A genetic algorithm to optimize the total cost and service level for just-in-time distribution in a supply chain. Int J Prod Econ 111:229–243
11. Gen M (1997) Genetic algorithm and engineering design. Wiley, New York
12. Holland JH (1975) Adoption in neural and artificial systems. The University of Michigan Press, Ann Arbor
13. Jasemi M (2006) A vendor-managed-inventory, Master of Science Thesis, Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran (in Farsi)
14. Kaipia R, Holmstrom J, Tanskanen K (2002) VMI: What are you losing if you let your customer place orders? Prod Plan Control 13:17–25
15. Kleywegt AJ, Nori VS, Savelsbergh MWP (2004) Dynamic programming approximations for a stochastic inventory routing problem. Transp Sci 38:42–70
16. Lambert DM, Cooper MC (2000) Issues in supply chain management. Ind Mark Manag 29:65–83
17. Lee HL, So KC, Tang CS (2000) The value of information sharing in a two level supply chain. Manag Sci 46:626–643
18. Magee JF (1958) Production planning and inventory control. McGraw Hill, New York
19. Michalewicz Z (1994) Genetic algorithms+data structures=Evolution program. Springer, Berlin
20. Miller A (2002) Subset selection in regression, 2nd edn. Chapman & Hall/CRC, Boca Raton
21. Mitchell M (1999) An introduction to genetic algorithms. MIT Press, Boston
22. Myers RH, Montgomery DC (2002) Response surface methodology—process and product optimization using designed experiments. Wiley, New York
23. Nachiappan S, Jawahar N (2007) A genetic algorithm for optimal operating parameters of VMI system in a two-echelon supply chain. Eur J Oper Res 182:1433–1452
24. Park Y (2001) A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines. Int J Prod Econ 73: 175–188
25. Pasandideh SHR, Niaki STA, Aryan-Yeganeh J (2010) A parameter-tuned genetic algorithm for multi-product economic production quantity model with space constraint, discrete delivery orders, and shortages. J Adv Eng Softw 41:306–314
26. Pasandideh SHR, Niaki STA, Mirhosseyni SS (2010) A parameter-tuned genetic algorithm to solve multi-products EPQ model with defective items, rework, and constrained space. Int J Adv Manuf Technol 49:827–837
27. Pourakbar M, Farahani RZ, Asgari N (2007) A joint economic lot-size model for an integrated supply network using genetic algorithm. Appl Math Comput 189:583–596
28. Samal AR, Mohanty MK, Fifarek RH (2008) Backward elimination procedure for a predictive model of gold concentration. J Geochem Explor 97:69–82
29. Sofifard R, Hosseini SMM, Farahani RZ (2007) The study of vendor-managed-inventory in supply chain with mathematical model. Master of Science Thesis, Department of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran (in Farsi)
30. Taleizadeh AA, Niaki STA, Aryanezhad MB, Fallah-Tafti A (2010) A genetic algorithm to optimize multi-product multi-constraint inventory control systems with stochastic replenishments and discount. Int J Adv Manuf Technol 51:311–323
31. Van Der Vlist P, Kuik R, Verheijen B (2007) Note on supply chain integration in vendor-managed inventory. Decis Support Syst 44: 360–365
32. Yao YL, Evers PT, Dresner ME (2007) Supply chain integration in vendor-managed inventory. Decis Support Syst 43:663–674