Journal of Optimization in Industrial Engineering
Vol.13, Issue 1, Winter & Spring 2020, 55-66
DOI:10.22094/joie.2017.445.0

Research Article

Presenting a Joint Replenishment-location Model Under all-units Discount and Solving by Genetic Algorithm and Harmony Search Algorithm

Reza Abdollahi Sharbabaki [a], Seyed Hamidreza Pasandideh [b,*]

[a] Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran
[b] Faculty of Engineering, Department of Industrial Engineering, Kharazmi University, Tehran, Iran
Received 19 February 2015; Revised 27 May 2015; Accepted 27 June 2015

Abstract

In this paper a model is propose for joint replenishment and locations distribution centers (DCs) of a distribution system that is responsible for ordering and dispatching shipments of a single product to DCs. The warehouse spaces DCs are limited and these can determine amount of requirement product by considering proposed discount. The propose model is develop to minimize total costs consists of locating, ordering, holding and purchasing under condition all-units discount. In this model number and location of DCs, joint replenishment frequencies and optimum order quantity of each DC are defined. To solve joint replenishment problem (JRP) determining optimal limits upper and lower replenishment cycle time T is very important. For determining this limits we use quantity discount RAND algorithm (QD-RAND). To use this method need to determine the location each of DCs. Therefore, first we consider the limits between a very small amount and 2 then the model solved by genetic algorithm (GA). After obtaining the optimal upper and lower replenishment cycle time T, the model will be resolved by harmony search algorithm (HSA) and GA. The parameters of all algorithms are first calibrated by means of the response surface methodology (RSM). The comparison results based on different problem sizes are in favor of HS.

Keywords: Joint replenishment problem; Location; All-units discount; Genetic Algorithm; Harmony Search.

1. Introduction and Literature Review

Joint replenishment (JR) is an applicable inventory problem in which group items into the same order from a supplier or a same place to achieve the purpose of sharing the main preparation costs. Also JRP is relevant when replenishing a single item in multiple locations helping companies to develop strategies that will help exploiting economies of scale combining shipments to multiple locations. JR of multiple locations is possible when all of these locations are centrally controlled or when these locations are in coalition for joint replenishment. This is the case of some franchise stores which are located in the same city or ATM machines belonging to same financial institution (Silva and Gao, 2013). Generally in JRP, it is assumed that unit cost is constant, no matter what quantity is purchased. But in reality, suppliers may induce their customers to place larger orders by offering them quantity discounts. If the quantity purchased is greater than a specified "price break" quantity, the cost per unit is reduced. Two types of price break schedule can be considered (all-units and incremental discount schedule). The all-units discount applies the discounted price to all units beginning with the first unit, if the quantity purchased exceeds the price break quantity. The incremental discount schedule applies the discounted price only to those units over the price break quantity. Cha and Moon (2005) modeled the joint replenishment problem for multiple products considering all-units discount and constant demand. They developed a heuristic algorithm and an intelligent algorithm to solve JRP and explained these algorithms by numerical examples. Taleizadeh et al. (2010) considered an optimizing multiproduct multi constraint inventory control systems with stochastic replenishment intervals and discount. In this research they considered that the period between two replenishments is independent and also added the constraints of warehouse space and budget. In this model the incremental discounts to purchase products are considered, and a combination of backorder and lost sales are taken into account for the shortages. They used genetic algorithm and simulated annealing to solve this problem.

There are two common kinds of JRP: the single-buyer JRP (SJRP) and multi-buyer JRP (MJRP). In real conditions many studies have been conducted about SJRP (Wang and et al., 2012). Hoque (2006) modeled the JRP with storage capacity, transport capacity, and budget constraints. The

*Corresponding author Email addresses: pasandid@yahoo.com

main objective is calculate the appropriate lower bound of the basic cycle time for minimizing the total cost. Porras and Dekker (2006) modeled the JRP for M items under certain demand with minimum order quantity constraint for each item in the replenishment order. They obtained the range of basic cycle time and proposed an efficient global optimization method to solve the JRP with constraints. The proposed algorithm was tested with data from a real case and some additional numerical experiments. As for the MJRP, there are a few literature reviews. The MJRP is a common method for multi-branch companies that their branches order a group of items from a supplier. Obviously, JRP among the branches reduce the firm's ordering and inventory costs. (Chan and et al., 2006). Chan et al. (2003) presented GA for solving MJRP problem. Li (2004) designed RAND algorithm for solving MJRP problem. Lu et al. (2010) solved the MJRP problem with RAND algorithm considering modeling resource constraint.

There are two strategies to solve joint replenishment problem: A Direct Grouping Strategy (DGS) and Indirect grouping strategy (IGS). Under DGS, products are partitioned into a predetermined number of sets and the products within each set are jointly replenished with the same cycle time. Under IGS a replenishment is made at regular time-intervals and each product has a replenishment quantity sufficient to last for exactly an integer multiple of the regular time interval. Groups in IGS are indirectly formed by products having the same integer multipliers. The literature suggests that IGS outperforms DGS for high major ordering cost because many products can be jointly replenished when using an IGS (Khouja and Goyal, 2008). Arkin et al. (1989) proved that JRP is a NP-hard problem. Thus different methods are proposed to solve JPR. Generally, these methods can be divided in three groups: (1) heuristic methods, (2) meta-heuristic methods, and (3) special methods. RAND method can be noted as one of the heuristic methods for solving JRP that was presented by Kaspi and Rosenblatt (1991). Also Goyal and Deshmukh (1993), van Eijs (1993), Hariga (1994), Viswanathan (1996) and Fung and Ma (2001) presented other heuristic algorithms to solve this problem. For further studies on this topic you can refer article Khouja and Goyal (2008). The main objective in each heuristic algorithm is to find upper and lower limit for T (T_{min}, T_{max}). Meta-heuristic methods for solving JRP: Hong and Kim (2009) proposed a GA to solve JRP with exact inventory cost. Khouja et al. (2000) presented a GA to solve JRP and compared its efficiency with RAND algorithm. There are several special methods for solving JRP such as the power-of-two (PoT) policy (Lee & Yao 2003) and the evolutionary computing (Olsen 2005) to solve JRP.

In recent years many studies have been done to consider JRP with logistic activities (like delivery, location and routing). Wang et al. (2012) considered Joint replenishment and delivery (JRD) problem as an important applicable managerial problem under stochastic demand. They designed an effective and efficient hybrid differential evolution algorithm (HDE) based on the differential evolution algorithm (DE) and GA to solve this problem. They compared HDE with GA and according to obtained results they found out HDE is faster than GA and has a higher convergence rate. Qu et al. (2013) modeled JRD where a warehouse orders different products from suppliers and delivers them to retailers. The objective is to determine grouping and scheduling decisions and specify order quantities and deliver them to retailers in order to minimize the costs. They designed Adaptive hybrid differential evolution (AHDE) algorithm to find the optimal solution.

About JRP-location there are very few studies. Silva and Gao (2013) published the first article about JRP-location. The model as a facility location model not only includes the location fixed cost but also considers inventory replenishment cost. They proposed a two-stage heuristic algorithm to solve the model. In the first stage distribution centers will be located and their total location cost is computed according to this rule: demand points are always dedicated to the nearest distribution center. In the second stage JRP will be solved according to specified places in first stage. In this stage Vishwanathan algorithm is used to determine the best policy of Joint replenishment. They propose a Greedy Randomized Adaptive Search Procedure (GRASP) to solve the problem. Wang et al. (2013) proposed Hybrid Self-Adapting Differential Evolution Algorithm to solve JRP-location and compared its efficiency with GA and HDE algorithms. Qu et al. (2014) modeled location-inventory problem considering joint replenishment and independent replenishment for several products, stochastic and independent demand and deficit cost. They designed a two-stage solution method. In the first stage two sets of customers allocated to open DCs are defined and total annual cost for each open DC is obtained by dedicating sets of customers to sets of open DCs. In second stage total open DCs cost in all potential locations will be sorted descending and optimal locations are the first ones in the sorted area.

The remainder of the paper is organized as follows. In Section 2, the problem is stated. In Section 3, the proposed mathematical model is stated. In Section 4, the solution approaches are described. Section 5 demonstrates the parameter tuning and statistical comparison of the proposed algorithms on several problem instances of different sizes. Finally, conclusions and future studies come in Section 6.

2. Problem Definition

In this paper we examine the JRP and location of DCs in a distributed system with a centralized decision maker that is responsible for ordering and dispatching shipments of a single item to the distribution centers. The warehouse spaces DCs are limited and these can determine amount of requirement product by considering proposed discount. The model seeks to minimize the total costs of joint replenishment and costs of locating DCs, which joint replenishment cost is includes ordering, purchase under all-units discount and holding. The model presented in this paper is an integrated approach between the decisions of the location and inventory. In this model number and location DCs, joint replenishment frequencies and optimum order quantity of each DC are defined as the total location costs and joint replenishment costs be minimized.

2.1. Applications

This condition is applied in locating DCs of concessionaire companies that are located in same cities.

3. Modeling

This study develops model of Silva and Gao (2013) under constraints of all-units discounts and constraints of storage space for each DC. The objective is to minimize the total joint replenishment costs and the cost of locating the DCs in potential sites.

3.1. Assumptions

1. There is only one product.
2. DCs are replenished jointly in different locations.
3. Demand is constant and known.
4. The stock replenishment admits non-integer quantities of the items (principle of the divisibility of the variables).
5. Product price is related to the replenishment.
6. The stock replenishment is immediately.
7. Stock shortage is not allowed.
8. The waiting time of supply is zero.
9. Storage space of distributers is limited.

3.2. Model constraints

1. Constraints of storage space for DCs.
2. Constraints of all-units discount.

3.3. Parameters

j :Index of customer j(j=1,…, J).

i :Index of DCs or warehouses i(i=1,…,I).

o :Index of price break (o =1,…, O).

f_i: Fixed cost for opening of DC i.

c_{ij} :Cost of allocation DC i to customer j

S :A cost of replenishment for set, from DCs (independent from number of DCs in joint replenishment)

s_i :Variable cost for DC i in joint order;

h_i :Holding cost (maintenance) of a unit of the item in warehouse by unit of time at DC i.

d_j :Demand for the item by unit of time at customer j, constant and known.

D_i :Demand for the item by unit of time allocated to DC i.

R_{io} :Purchase quantity by DC i in price break o.

Q_{io} :The quantity of ordered by DC i in price break o.

u_{io} :The Upper bound quantity by DC i in price break o.

q_{io} :The Lower bound quantity by DC i in price break o.

p_{io}: Unit purchase cost by DC i in price break o under all-units quantity discounts.

w_i: Maximum storage capacity for DC i

t_i: Time between consecutive orders in DC i, called the reorder interval

k_i :An integer multiple of a basic cycle T for DC i.

3.4. Decision variable

$$X_{ij} = \begin{cases} 1 & \text{if customer } j \text{ is assigned to a DC at } i, \\ 0 & \text{otherwise.} \end{cases}$$

$$Y_i = \begin{cases} 1 & \text{if a DC is located at } i, \\ 0 & \text{otherwise.} \end{cases}$$

$$b_{io} = \begin{cases} 1 & \text{if quantity DC } i \text{ triggering the } o \text{ th price break}, \\ 0 & \text{otherwise.} \end{cases}$$

T :Basic cycle time of the replenishment.

3.5. Model formulation

$$\min \sum_{i=1}^{I}\sum_{j=1}^{J} c_{ij} X_{ij} + \sum_{i=1}^{I} f_i Y_i + \frac{S + \sum_{i=1}^{I} \frac{s_i}{k_i}.Y_i}{T} + \sum_{i=1}^{I} \frac{h_i k_i T \sum_{j=1}^{J} d_j X_{ij}}{2} + \sum_{i=1}^{I}\sum_{o=1}^{O} p_{io} R_{io} \quad (1)$$

S.t:

$$\sum_{i=1}^{I} X_{ij} = 1 \quad ; \quad \forall j \quad (2)$$

$$X_{ij} - Y_i \le 0 \quad ; \quad \forall i, j \quad (3)$$

$$D_i = \sum_{j=1}^{J} d_j X_{ij} \quad ; \quad \forall i \quad (4)$$

$$D_i = \sum_{o=1}^{O} R_{io} \quad ; \quad \forall i \quad (5)$$

$$q_{io} b_{io} \le R_{io} k_i T \le u_{io} b_{io} \quad ; \quad \forall i, o \quad (6)$$

$$Y_i = \sum_{o=1}^{O} b_{io} \quad ; \quad \forall i \quad (7)$$

$$\sum_{o=1}^{O} R_{io} k_i T \le w_i \quad ; \quad \forall i \quad (8)$$

$$X_{ij} \in \{0,1\} \quad ; \quad \forall i, j \quad (9)$$

$$Y_i \in \{0,1\} \quad ; \quad \forall i \quad (10)$$

$$b_{io} \in \{0,1\} \quad ; \quad \forall i, o \quad (11)$$

$$k_i \ge 0, \text{integer} \quad ; \quad \forall i \quad (12)$$

$$T \succ 0 \quad (13)$$

The purpose of (1) minimizes total location costs and total joint replenishment costs. Constraint (2) ensures that a demand center is allocated to one and only one DC. Constraint (3) ensures a demand center will be dedicated to an active DC. Constraint (4) indicates demand of the item by unit of time allocated to DC i. As shortage is not allowed, Constraint (5) ensures that all the customers' demand of the item from DC i is purchased at different price-break points. Constraints (6) shows that the order quantity of the item must be purchased between the discount rages. Constraint (7) states that DCs can buy goods only if located. Constraints (8) optimum order must be smaller than warehouse space. Constraints (9)-(13) are the decision variables of the problem.

Total location costs include cost of allocate customers to active centers and the construction cost of the facilities. The relevant costs associated with the problem of joint replenishment of stocks in accordance with the model assumptions are classified as ordering costs, holding costs and purchasing costs under discount. The ordering costs are divided a fixed part (main setup cost) (S) and a secondary cost (variable setup cost) (s_i) ordering fixed cost occurs when an order is independent of the number of DCs that participated in the replenishment. The holding cost per unit of time (h_i) results from storage of per unit of goods in a warehouse of DC i, when the goods are stored for consumption or commerce. Cost of each DC procurement occurs based on an all-units discount policy. However, as DC i is to be jointly replenished, we weight the quantity by its replenishment frequency as a function of T. Therefore, we will be able to identify the quantity to order from the expression $Q_i = k_i . D_i . T$ (Cha and Moon, 2005).

4. Solution

Joint replenishment- Location model examined in this paper is a non-linear and integer model. Arkin et al. (1989) showed that JRP is NP-hard in large scales and cannot be solved by exact methods. Therefore, taking into account the costs of purchase under the terms of all-units discount and add storage space limitation the problem would be difficult to solve. To solve joint replenishment problem (JRP) determining optimal limits upper and lower joint replenishment time $[T_{min}, T_{max}]$ is very important. Many innovative techniques have been developed for this purpose. The only article that considered JRP with all-units discount is a paper presented by Cha and Moon (2005). To solve this they proposed Quantity Discount RAND algorithm. In this algorithm upper and lower bound $[T_{min}, T_{max}]$ are computed by Eq. (14) and Eq. (15):

$$T_{\max} = \max\left[\sqrt{\frac{2\left(S + \sum_{i=1}^{n} s_i\right)}{\sum_{i=1}^{n} D_i h_i}}, \frac{\max\{u_{io}\}}{D_i}\right] \quad (14)$$

$$T_{\min} = \min\left(\sqrt{\frac{2 s_i}{D_i h_i}}\right) \quad (15)$$

On the other hand, demand of each DC is shown by D_i and according to Eq. (4) is dependent to decision variable X_{ij}. To use the Eq. (14) and Eq. (15), X_{ij} must be determined. To this end, we have proposed a two-stage method. First, by limiting the joint replenishment time between a very small amount and 2 the model is solved by GA and location of each DC (X_{ij}) is obtained. Then the optimal upper and lower bound $[T_{min}, T_{max}]$ are determined by Eq. (14) and Eq. (15). In the second stage, the model with the optimal upper and lower bound $[T_{min}, T_{max}]$ is resolved by GA and HSA.

4.1. Harmony search algorithm

Harmony search algorithm (HSA) was presented by Geem et al. (2001). The applicability of the algorithm for discrete and continuous optimization problems, a little arithmetic, simple concept, easy to implement and few parameters has made this algorithm as one of the most used optimization algorithms in recent years on various issues. This meta-heuristic algorithm compared with other methods has less mathematical requirements and can be implemented in various engineering problems by changing the parameters and operators. HSA is a meta-heuristic algorithm, which is inspired by the music and it is like improvisation of musicians. In this algorithm, the objective function is interpreted as an estimate of harmony and beauty of the performers in finding an appropriate form of coordination. The algorithm is inspired by the process of playing music and trying to find a wonderful harmony within it. Music and musicians began to involve in the process of looking for a better harmony. Musician, due to his former works, tries to play his best harmony or improve his existing harmony. Also, he can create new music which he had no experience about that. In the process of playing music without any previous experience, you can be playing any note within its authorized sounded distance the note with other notes produce a harmonic vector. If desired harmony is produced, it will be stored in the player's mind and increases the possibility of producing better harmonies in subsequent plays and exercises. Similarly, in engineering problems and process optimization, first we consider a possible value for each variable, and set of values for the variables forms a response vector. If the vector is a good answer to the question, the amount intended for the variables is saved in memory, and the possibility of finding a better response in the next solutions increases (Hajipour and et al., 2014).

4.1.1. Representation

The chromosome designed for the algorithm in this paper has five parts whose structure is shown in Fig. 1. The first part of the chromosome is a vector of DCs length. Each gene in this part gets values between zero or one; one for an active DC and zero otherwise. The second part of the chromosome is the same as the first, except that the genes take positive-integer values for the replenishment frequencies of the DCs (k_i). It should be noted that genes of this section that are corresponding to zero-value nodes of the first part of the chromosome, are zero. This ensures DCs that are not constructed will not be replenished. The third part is as same as the second part of the chromosome, with the difference that in this part of the chromosome nodes will accept values between one and price break points. The remarkable thing is that each of the genes of this part of the construction get values just after the center is constructed. The fourth part of the chromosome is a vector of number of demand centers length, and each genes take amounts between one and number of DCs. It should be noted with this action each of the demand centers will be allocated to a DC. In design of this part of chromosome, DCs which are not constructed, will not be selected for responding to the demands of demand centers. The fifth chromosome is the continuous variable of time between two replenishments which value is between high (M) and low (ε). M and ε first get values manually and then by using Eq. (14) and Eq. (15), the optimal values are determined.

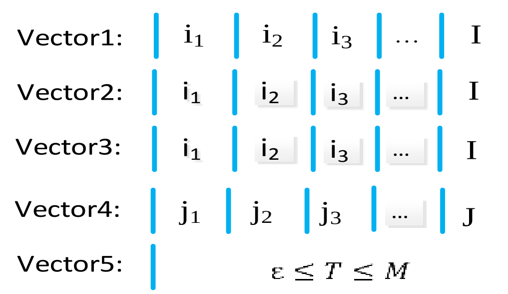

Fig. 1. A chromosome structure

In Fig. 2, an example of this chromosome for a problem with 5 potential DCs, 7 demand points and 3 price break points is shown.

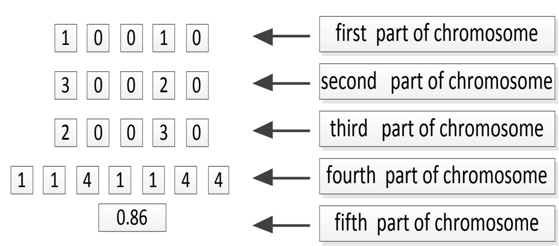

Fig. 2. An example of the chromosome structure

In this example, according to the first part of chromosome, DC 1 and 4 are constructed and according to the second part of the chromosome, the replenishment frequencies of the first DC is 3 and replenishment frequencies of the fourth DC is 2. With regard to the second part, DC 1 uses the offered price of the second price break point, and DC 4 purchases the item at the price offered in the third price break point. The last chromosome section shows that customer demands 4,2,1 and 5 are supplied by first DC and customer demands 3,6 and 7 are supplied by fourth DC. And in the fifth part the time between two replenishments is determined 0.86.

4.1.2. Evaluation

The generated chromosomes satisfy most of the constraints, occasionally Constraints (5), (6) and (8) can be violated. To deal with this type of violations, the penalty method is utilized, in which infeasible chromosomes are fined based on their degree of violation. Penalty functions reduce unjustified answers according to violation of the restrictions. The penalty function makes problems with constraints become problems with no constraints. The idea the penalty function is shown in Eq.16:

$$F(x)=\begin{cases} f(x) & ; \quad x\in Feasible\,Region \\ f(x)+p(x) & ; \quad x\notin Feasible\,Region \end{cases} \qquad (16)$$

where $P(x)$is the amount of fine. If a constraint is not violated $P(x)$ is zero, otherwise, it takes a big positive value. Moreover, due to severity involved in violating different constraints, it is necessary to normalize all the limitations before applying Eq. (16). For example, a limitation like $g_j(x) \leq b_j$ can be normalized using the Eq. 17.

$$p(x)=\omega\times Max\left\{\left(\frac{g(x)}{b}-1\right),0\right\} \qquad (17)$$

where ω is a large number and $\frac{g(x)}{b}-1$ is the amount of difference or unjustified of a constraint. This will scale-up all constraints and we can easily put them together and just one fine as penalty parameter of all constraints will be added to the objective function.

4.1.3. Improving process

Three improvement operations are involved in a harmony search algorithm described as follows.

HM considering operator: Using memory in HSA is similar to elitism in GA. This ensures that the best harmony won't be lost during the optimization process. This operator is How the display the solution and the process of evaluating the GA is similar to the HSA. To offspring production in GA we use Crossover and Mutation operator.

Crossover operator: in this paper to produce new offspring at each iteration of the algorithm, uniform crossover operator is used for the parts first to fourth and arithmetic controlled with a rate called harmony memory considering rate (HMCR). On the one hand, the low rate makes the algorithm converges very quickly because of the small number of elite harmonic improvisation is selected. On the other hand, too much of this rate leads the algorithm just using the existing harmony and the algorithm will converge to a weak point of the local optimization. Therefore, we calibrate it in the range [0.75-0.95].

Pitch adjustment rate (PAM): In the musical mode, rate adjustment step means little change in frequency. Similarly, in optimization process, the rate of adjustment step means to produce a few different solutions (neighbor). In fact, the solution space that is not searched by previous operators is likely to be searched by the algorithm. The operator uses a rate named PAR for adjustment control. This function is similar to the mutation operator of genetic algorithms. Thus large amount of PAR makes a variety of solutions to increase. As a result PAR is set in the range of [0.1 - 0.5] (Hajipour and et al., 2014).

To perform this operation, one (or more) chromosome vectors are randomly selected. Then, the switch (swap) operator will be used to implement adjustment (pitch) operator. In this strategy, we have two points of this vector randomly displaced. The operation is illustrated in Fig. 3.

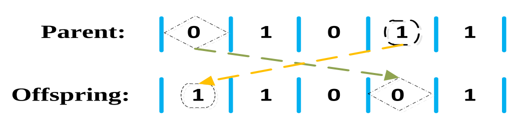

Fig. 3. An example of the pitch-adjusting operator

Randomization operator: same as adjustment step, the operator is also used to increase the variety of answers. However, the operator considers a wider variety answers of locally optimal solution is going to be a global optimum. Probability function of the random operator P_{rand} is:

$$P_{rand}=1-HMCR \qquad (18)$$

4.2. Genetic algorithm

crossover for the fifth. By utilization this type of operator it always produces offspring that are regulated and the creation of children without association with any member of the population is prevented. To implement a uniform crossover operator, you must have a random matrix (β) with values of zero and one. The dimensions of this matrix is equal to the size of the parent chromosome. Children will be created using the Eq. (19) and Eq. (20).

$$offspring1 = \beta \times parent1 + (1-\beta) \times parent2 \tag{19}$$

$$offspring2 = \beta \times parent2 + (1-\beta) \times parent1 \tag{20}$$

The arithmetic crossover operator is similar to the uniform crossover operator. Except that the β value is not a value between zero and one but a value between maximum and minimum of the continuous variable.

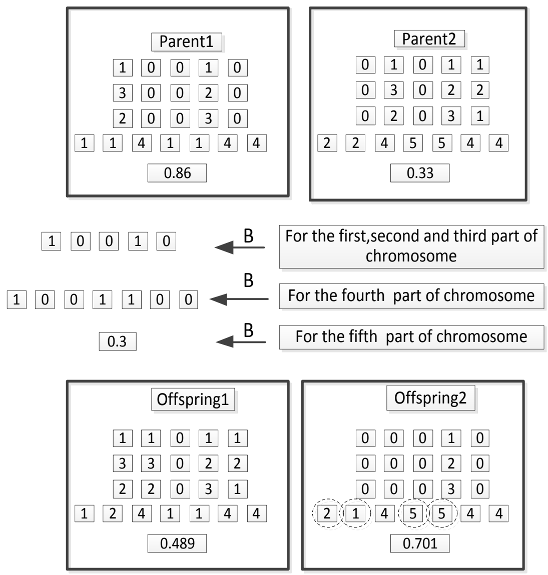

Fig .4. An illustration of the crossover operator

By applying this operator the demand point may be allocated to the DC that is not constructed. So an action should be taken to eliminate this unjustified condition. For this, places that are created in first part of child chromosome will be assumed as amount of new genes randomly placed. In this case the child chromosome is justified.

Mutation operator: mutation rate is an important concept in relation to the operator. Mutation rate is the percentage of the genes on each chromosome that are subject to change. If the mutation rate is very small, a large number of genes that could be helpful won't be testes and if the mutation rate is too large random there will be random differences and the child will lost similarities to parents. This will lose the historical memory of the algorithm. Thus the optimum value should be selected for mutation rate. Uniform mutation operator is used in this article. In this operation, first, a number of genes will be randomly selected from each chromosome, then the amount is changed randomly in the allowable limit. The number of selected genes from each

Table 2 and are tuned using the response surface methodology (RSM). In this regard, the 30 problems of different size of small, medium, and large are randomly

chromosome for uniform mutation operation is obtained by multiplication of mutation rate and number of genes in the chromosome. Normal mutation is used for the fifth part of the chromosome.

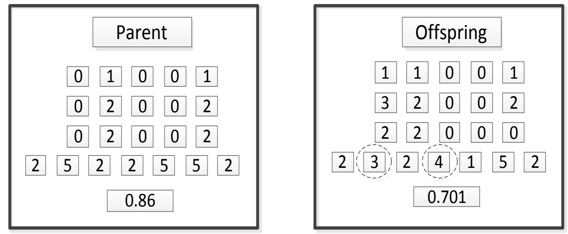

Fig. 5. An illustration of the mutate operator

5. Performance Evaluation

In this section, first the problem inputs are determined then 30 examples are solved in various aspects using GA. The parameters of algorithm will be set and after the parameter tune, examples are resolved. According to the obtained results, upper and lower bound $[T_{min}, T_{max}]$ are determined for each example. Then due to optimal $[T_{min}, T_{max}]$, the presented examples will be solved by using HSA and GA. Results before and after determining the optimum limits are evaluated according to the objective function value and the required computational time (Time).

In this paper the algorithms are coded in MATLAB 2012b and are implemented on a 1.8-GHz laptop with four GB RAM.

The inputs are generated based on what follows:

Typical values for the parameters of the model are determined with respect to the matters contained in the literature and based on uniform distribution. These values are shown in Table 1.

Table1
Parameters for the joint replenishment-location model

Parametr range			
d_j	U [80 800]	s_i	U [1 10]
f_i	U [400 800]	w_i	U [100 1000]
S	45	c_{ij}	U [10 45]
h_i	U [0 1]		

5.1. Parameter tuning

As the acquired results of the meta-heuristic algorithms are sensitive to parameter, a small change can affect the quality of the solution obtained. Therefore, one needs a fine tuning procedure for the parameters in order to find better solutions. These parameters are given in

generated to calibrate the parameters of both algorithms. The range of each parameter is shown in Table 3.

Table2
Parameters for GA and HSA

GA	Parameters	Population Size	Crossover ratio	Mutation ratio	Number of the iterations	
		$nPop$	P_c	P_m	$nItr$	
HS	Parameters	Population Size	Pitch adjusting rate	Outer loop size	Inner loop size	Harmony memory rate
		$nPop$	PAR	$out.Loop$	$In.Loop$	$HMCR$

Table3
Parameters' ranges along with their Levels

Optimization Algorithms	Algorithm Parameters	Parameter Range
GA	$nPop$	50-150
	P_c	0.5-0.99
	P_m	0.01-0.4
	$nItr$	800-1600
HS	$out.Loop$	800-1600
	PAR	0.1-0.5
	$In.Loop$	10-50
	$nPop$	50-200
	$HMCR$	0.75-0.95

Each sample is performed 3 times and the average of 3 runs is intended as output, each time their parameters (factors) are randomly changing based on described in permitted range in Table 2. The responses in each run are the objective function value and CPU time. The values of the related responses are first normalized by the linear norm. Then, a quadratic regression function for each measure is estimated using the MATLAB software to find significant relationships between the parameters and their response. At the end, the average estimation of the responses using the four estimated regression functions is taken to be maximized by the GAMS software, in order to find the optimal combinations of the parameters. The quadratic regression function consists of linear, interaction, and quadratic coefficients shown in Eq. (21).

$$\begin{aligned} E(y) &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 \\ &+ \beta_{11} X_1^2 + \beta_{22} X_2^2 + \beta_{33} X_3^2 + \beta_{44} X_4^2 \\ &+ \beta_{12} X_1 X_2 + \beta_{13} X_1 X_3 + \beta_{14} X_1 X_4 \\ &+ \beta_{23} X_2 X_3 + \beta_{24} X_2 X_4 + \beta_{34} X_3 X_4 \end{aligned} \tag{21}$$

Where $E(y)$ is the expected value of the response, β_0 is a constant that represents the intercept, $\beta_i, i = 1,2,3,4$ are linear coefficients, $\beta_{ii}, i = 1,2,3,4$ are the quadratic coefficients, β_{ij}, $i \neq j$ are coefficients of the interaction and X_i is the parameter of GA and HSA.

As an example, Table 4 contains the experimental results of employing GA. Based on the results provided in Table 4, the regression function estimated for the GA is:

$$\begin{aligned} E(y) &= 0.036 - 0.00827 nPop - 0.017 P_c \\ &-0.019 P_m + 0.004889 nItr - 0.016 nPop^2 \\ &+0.001167 P_c^2 + 0.01 P_m^2 - 0.013 nItr^2 \\ &-0.007133 nPop \times P_c - 0.0023 nPop \times P_m \\ &+0.006396 nPop \times nItr + 0.0038 P_c \times P_m \\ &-0.036 P_c \times nItr - 0.0022 P_m \times nItr \end{aligned} \tag{22}$$

HSA parameters can be calculated similar to GA. The results to calibrate the parameters of both algorithms are presented in Table 4.

Table 4
Computational results of applying GA

No.	Algorithm Parameters				Obtained Response	
	$nPop$	P_c	P_m	$nItr$	Best Solution	time
1	55	0.55	0.2	900	239935.556	25.599502
2	70	0.65	0.3	850	240586.8632	36.444389
3	75	0.53	0.28	1150	478827.3111	48.684753
4	130	0.9	0.35	900	239973.1923	90.473375
5	80	0.7	0.3	920	240454.397	72.806932
6	100	0.8	0.15	1100	239973.1599	60.385801
7	120	0.85	0.2	1000	240008.9694	72.806932
8	60	0.6	0.05	800	239973.6827	18.899531
9	65	0.55	0.25	860	239934.689	30.268056
10	150	0.99	0.06	820	239934.689	67.947217
11	120	0.72	0.2	1000	240016.6224	66.725678
12	85	0.63	0.33	1300	239934.689	68.054539
13	100	0.8	0.15	1100	239936.0829	60.439298
14	80	0.7	0.3	950	239934.8129	51.18934
15	95	0.64	0.36	1400	239935.5512	96.324997
16	140	0.95	0.05	800	203532.7815	58.691355
17	60	0.79	0.22	1550	239935.6454	57.807847
18	75	0.72	0.24	1600	255457.3235	71.476132
19	55	0.76	0.35	1500	239939.256	58.226897
20	110	0.77	0.22	1200	250355.8724	76.463927
21	100	0.9	0.1	1000	478991.4218	55.678054
22	105	0.5	0.158	1050	239934.7051	46.298375
23	150	0.8	0.06	1200	479497.7003	84.140858
24	90	0.6	0.3	1000	240177.0666	49.687906
25	105	0.77	0.23	1100	239934.689	68.134844
26	88	0.66	0.05	850	239935.5512	28.913946
27	88	0.88	0.09	1100	239973.8608	52.777878
28	110	0.59	0.11	1111	449435.7999	49.302571
29	150	0.77	0.29	950	255313.6317	92.139846
30	120	0.8	0.4	900	250966.4225	80.412876

Table 5
Parameter ranking of the algorithms

GA	Parameters	$nPop$	P_c	P_m	$nltr$	
	Rank	105	0.77	0.25	950	
HS	Parameters	$nNew$	PAR	HMS	$nltr$	$HMCR$
	Rank	40	0.1	200	1000	0.5

5.2. Computed results

To evaluate and compare the performance of GA before and after calculating $[T_{min}, T_{max}]$, and also to evaluate the performance of the GA and HSA, 30 examples of various aspects before and after calculating the optimal $[T_{min}, T_{max}]$ have been solved. These examples are presented in Table 6.

Table 6
Computational results of the solving methodologies

Problem No.	I	J	O	GA before calculating the upper and lower		GA after calculating the upper and lower		HS after calculating the upper and lower	
				Best Solution	time	Best Solution	time	Best Solution	time
1	2	5	3	8531.1241	98.15001	8436.9449	90.122436	8436.9877	38.837961
2	3	5	3	16912.1679	91.406163	16857.5678	91.13136	16750.8302	40.57735
3	2	5	2	72220.6771	88.561894	72120.6771	86.461894	72086.1941	38.275775
4	4	7	3	118835.6972	100.97112	118735.5973	90.42436	118737.7601	41.546342
5	3	8	3	19612.3216	99.302427	19502.6423	91.674115	15091.424	39.196794
6	3	8	3	13098.8306	90.57132	12498.8306	91.627572	12498.9683	41.133786
7	5	8	3	17085.6317	101.16799	17085.6317	92.122436	10792.3824	44.946944
8	2	8	3	60877.0239	96.015914	55310.2421	89.179007	55317.8619	38.135755
9	4	8	4	315308.2914	101.14301	247540.566	100.408542	247540.5763	46.226219
10	2	9	2	290433.7142	98.633152	290433.7142	97.733152	290434.0432	47.43381
11	4	10	4	71164.9091	111.19638	41073.9234	98.855436	37357.398	50.372374
12	5	10	4	286153.9187	114.6338	239998.4411	107.582128	240018.2148	48.40526
13	7	11	3	53243.5866	102.66991	52203.6658	102.669908	51035.0127	47.842904
14	3	12	3	15209.2627	98.412706	15209.2627	94.311706	15207.6358	41.998581
15	3	15	2	800871.8956	131.67696	700871.8956	102.669908	700873.3465	41.57001
16	4	16	4	253532.7915	102.47501	203532.7915	102.475006	203754.3503	45.242714
17	3	13	4	1930861.137	127.7645	40045.8095	94.038247	35543.1778	40.230372
18	6	19	3	676074.2947	130.65098	676074.2947	100.650982	676077.7176	49.418174
19	5	18	3	395537.9288	101.04671	395537.9288	101.046706	396319.1285	45.953621
20	8	20	4	265830.8798	158.93516	537736.6324	130.106343	317287.781	54.011636
21	8	21	4	663345.0509	114.32754	424899.96	125.514528	483974.4786	54.952946
22	5	21	5	30842.432	91.153474	30842.432	90.232436	30842.4549	50.071673
23	8	24	4	474666.8966	101.4644	513043.3959	134.656065	332812.9576	54.101276
24	7	22	3	86825.1125	119.77855	76825.1125	114.878552	52968.9104	48.620477
25	7	26	3	1201001.151	118.11223	1112315.123	114.125351	772128.1283	52.131121
26	5	25	5	1162946.345	115.66321	1068956.655	113.468343	760007.4181	42.749169
27	9	30	3	1112123.421	133.10041	1090211.581	132.926407	1099803.71	56.265722
28	10	35	4	993273.33	151.32405	988873.33	145.62405	873523.5003	62.028575
29	8	50	5	1562646.031	160.84142	1472421.322	132.910867	1462646.031	62.910867
30	9	40	7	1093253.122	151.32405	1082873.131	148.12316	893611.6121	59.128575

In order to compare algorithms used in solving the problem, we used an of variance (ANOVA) approach at 95% confidence level. A typical test of hypothesis on the equality of the means is stated in Eq. (22) and Eq. (23). In this equation $\mu_{z(GA1)}$ is the mean value of GA objective function before obtaining the optimal $[T_{min}, T_{max}]$ and $\mu_{z(GA2)}$ is the mean value of GA objective function after obtaining the optimal $[T_{min}, T_{max}]$.

$$\begin{cases} H_0 = \mu_{z(GA1)} = \mu_{z(GA2)} = \mu_{z(HS)} \\ H_1 = \mu_{z(GA1)} \neq \mu_{z(GA2)} \neq \mu_{z(HS)} \end{cases} \quad (23)$$

$$\begin{cases} H_0 = \mu_{t(GA1)} = \mu_{t(GA2)} = \mu_{t(HS)} \\ H_1 = \mu_{t(GA1)} \neq \mu_{t(GA2)} \neq \mu_{t(HS)} \end{cases} \quad (24)$$

The ANOVA results to compare the objective function value and the CPU time of the two algorithms are shown in Figs. 6 and 7 using the Minitab 16 software. The results show no significant difference between the two algorithms in the objective function value but on basis of CPU time, HSA is hugely better than GA in both cases.

```
Source  DF           SS           MS     F      P
Factor   2  2.44707E+11  1.22354E+11  0.59  0.557
Error   87  1.80449E+13  2.07413E+11
Total   89  1.82896E+13
                        Individual 95% CIs For Mean Based on
                        Pooled StDev
Level    N    Mean   StDev  -----+---------+---------+---------+----
GA1     30  468737  530252            (-------------*-------------)
GA2     30  387402  435634      (------------*-------------)
HS      30  342783  388966  (-------------*------------)
                            -----+---------+---------+---------+----
                               240000    360000    480000    600000
```

Fig. 6. ANOVA and the related interval plots for CS metric

```
Source  DF      SS     MS       F      P
Factor   2   78760  39380  142.00  0.000
Error   87   24128    277
Total   89  102888
                       Individual 95% CIs For Mean Based on
                       Pooled StDev

Level    N    Mean  StDev  ---------+---------+---------+---------+
GA1     30  113.08  21.32                                    (--*--)
GA2     30  106.93  18.10                                (--*--)
HS      30   47.48   7.05  (--*--)
                           ---------+---------+---------+---------+
                                   60        80       100       120
```

Fig. 7. ANOVA and the related interval plots for Time metric

In the second comparison, the technique for order preference by similarity to ideal solution (TOPSIS) is used

$$\begin{array}{c} \\ GA1 \\ GA2 \\ HS \end{array} \begin{array}{c} \begin{array}{cc} z & t \end{array} \\ \begin{pmatrix} 468736.9325 & 113.0824817 \\ 387402.3034 & 106.9250334 \\ 342782.6664 & 47.4772261 \end{pmatrix} \end{array}$$

Fig. 8. The mean value of the objective function and the results of the 30 examples

The result obtained from implementing the TOPSIS method shows that the coefficient close of the HSA is equal to 0.678697, algorithms GA2 is equal to 0.321303 and GA1 is equal to zero. Using this analysis, the HSA to GA is better than genetic algorithm in both conditions.

6. Conclusion

In this paper, a new model of JRP-location under all-units discount was offered. In this model number and location DCs, joint replenishment frequencies and optimum order quantity of each DC are defined as the total location costs and joint replenishment costs be minimized. Since in joint replenishment problem (JRP) determining optimal limits upper and lower joint replenishment time $[T_{min}, T_{max}]$ is very important, To solve this model, we have proposed a two-stage method. First, by limiting the joint replenishment time between a very small amount and 2 the model is solved by GA and location of each DC (X_{ij}) is obtained. Then the optimal upper and lower bound $[T_{min}, T_{max}]$ are determined by Quantity Discount RAND Algorithm. In the second stage, the model with the optimal upper and lower bound to compare the two algorithms in terms of the objective function value and CPU time, simultaneously.

$[T_{min}, T_{max}]$ is resolved by GA and HSA. To demonstrate the applicability of the proposed model and to measure the efficiency of the two solution algorithms, various test problems of different sizes were randomly generated. An of variance (ANOVA) approach at 95% confidence level and the technique for order preference by similarity to ideal solution (TOPSIS) is used to compare the two algorithms in terms of the objective function value and CPU time. While the statistical comparison approach showed no significant difference between the two algorithms at 95% confidence level, the results obtained using the TOPSIS method showed HSA the better algorithm.

For future research, the model can be extended for a multi-product problem, other meta-heuristic algorithms can be utilized to solve the proposed problem, the backordering costs can be considered for the joint replenishment part of the model. Also this model can be used for modeling of incremental discounts. Future research can also consider other terms such as incremental discounts and budget.

References

Arkin, E., Joneja, D., Roundy, R. (1989). Computational complexity of uncapacitated multi-echelon production planning problems. *Operations Research Letters*. 8: 61-66.

Cha, BC., Moon, IK. (2005). The joint replenishment problem with quantity discounts under constant demand. *Operational Research Spectrum*. 27: 569-581.

Chan, C.K., Li, L.K., Ng, C.T., Cheung, B.K. (2006). Scheduling of multi-buyer joint replenishments. *International Journal of Production Economics*. 102: 132-142.

Chan, C.K., Cheung, B.K., Langevin, A. (2003). Solving the multi-buyer joint replenishment problem with a modified genetic algorithm. Transportation Research Part B: Methodological. 37 (3): 291-299.

Fung, R., Ma, X., (2001). A new method for joint replenishment problems. *Journal of the Operational Research Society*. 52: 358–362.

Geem, Z.W., Kim, J.H., Loganathan, G.V. (2001). A new heuristic optimization algorithm: harmony search. Simulations. 76: 60–68.

Goyal, S.K., Deshmukh, S.G. (1993). The economic ordering quantity for jointly replenishment items. *International Journal of Production Research*. 31: 109 -116.

Hariga, M. (1994). Two new heuristic procedures for the joint replenishment problem. *Journal of the Operational Research Society*. 45: 463-471.

Hoque, M.A. (2006). An optimal solution technique for the joint replenishment problem with storage and transport capacities and budget constraint. *European Journal of Operational Research*. 175:1033-1042.

Hong, S.P., Kim, Y.H. (2009). A genetic algorithm for joint replenishment based on the exact inventory cost. *Computers & Operations Research*. 6 (1): 167-175.

Hajipour, V., Rahmati, S.H. A., Pasandideh, S. H. R., Niaki, S.T.A. (2014). A Multi-Objective Harmony Search Algorithm to Optimize Multi-Server Location-Allocation Problem in Congested Systems. *Computers & Industrial Engineering.* 72:187-197.

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., Goyal, S. (2008). A review of the joint replenishment problem literature: 1989–2005. *European Journal of Operational Research.* 186: 1–16.

Khouja, M., Michalewicz, Z., Satoskar, S. (2000). A comparison between genetic algorithms and the RAND method for solving the joint replenishment problem. Production Planning & Control: *The Management of Operations.* 11(6): 556-564.

Li Q. (2004). Solving the multi-buyer joint replenishment problem with the RAND method. *Computers & Industrial Engineering.* 46: 755-762.

Lu, T., Jia, S., Li, Y. (2010). A Modified RAND Algorithm for Multi-Buyer Joint Replenishment Problem with Resource Constraints. Information Science and Engineering, Hangzhou. China.

Lee, F.C., Yao, M.J. (2003). A global optimum search algorithm for the joint replenishment problem under power-of-two policy. *Computers and Operations Research*. 30: 1319-1333.

Olsen, A.L. (2005). An evolutionary algorithm to solve the joint replenishment problem using direct grouping. *Computers & Industrial Engineering*. 48: 223–235.

Porras, E., Dekker, R. (2006). An efficient optimal solution method for the joint replenishment problem with minimum order quantities. *European Journal of Operational Research*. 174: 1595-1615.

Qu, H., Wang, L., Zeng, Y.R. (2013). Modeling and optimization for the joint replenishment and delivery problem with heterogeneous items. Knowledge-Based Systems. 1-9.

Qu, H., Wang, L., Liu, R. (2014). A Contrastive Study of the Stochastic Location-Inventory Problem with Joint Replenishment and Independent Replenishment. *Expert Systems with Applications.*

Silva, F., Gao, L. (2013). A Joint Replenishment Inventory-Location Model. *Networks and Spatial Economics*. 13: 107-122.

Taleizadeh, A.A., Niaki, S.T.A., Aryanezhad, M.B., Tafti, A.F. (2010). A genetic algorithm to optimize multiproduct multiconstraint inventory control systems with stochastic replenishment intervals and discount. *Journal of advanced manufacturing Technology*. 51: 311-323.

van Eijs, M.J.G. (1993). A note on the joint replenishment problem under constant demand. *Journal of Operational Research Society*. 44: 185-191.

Viswanathan, S. (1996). A new optimal algorithm for the joint replenishment problem. *Journal of the Operational Research Society*. 47: 936-944.

Wang, L., Dun, C.X., Bi, W.J., Zeng, Y.R. (2012). An effective and efficient differential evolution algorithm for the integrated stochastic joint replenishment and delivery model. Knowledge-Based Systems. 36: 104-114.

Wang, L., Qu, H., Liu, S., Chen, C. (2014). Optimizing the joint replenishment and channel coordination problem under supply chain environment using a simple and effective differential evolution algorithm. Discrete Dynamics in Nature and Society. Article ID 709856, 1–12, doi: /10.1155/2014/709856.

This article can be cited: Abdollahi Sharbabaki, R., Pasandideh, S. H. (2020). Presenting a Joint Replenishment-location Model Under all-units Discount and Solving by Genetic Algorithm and Harmony Search Algorithm. *Journal of Optimization in Industrial Engineering*. 13 (1), 55-66.

http://www.qjie.ir/article_538179.html
DOI:10.22094/joie.2017.445.0