Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Developing a multi-objective, multi-item inventory model and three algorithms for its solution

Ommolbanin YOUSEFI[†1], Mirbahadorgholi ARYANEZHAD[1], Seyed Jafar SADJADI[1], Arash SHAHIN[2]

([1]*Department of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran*)

([2]*Department of Management, University of Isfahan, Isfahan, Iran*)

[†]E-mail: oyousefi@iust.ac.ir

Received Dec. 28, 2011; Revision accepted June 13, 2012; Crosschecked July 6, 2012

Abstract: We develop a multi-objective model in a multi-product inventory system. The proposed model is a joint replenishment problem (JRP) that has two objective functions. The first one is minimization of total ordering and inventory holding costs, which is the same objective function as the classic JRP. To increase the applicability of the proposed model, we suppose that transportation cost is independent of time, is not a part of holding cost, and is calculated based on the maximum of stored inventory, as is the case in many real inventory problems. Thus, the second objective function is minimization of total transportation cost. To solve this problem three efficient algorithms are proposed. First, the RAND algorithm, called the best heuristic algorithm for solving the JRP, is modified to be applicable for the proposed problem. A multi-objective genetic algorithm (MOGA) is developed as the second algorithm to solve the problem. Finally, the model is solved by a new algorithm that is a combination of the RAND algorithm and MOGA. The performances of these algorithms are then compared with those of the previous approaches and with each other, and the findings imply their ability in finding Pareto optimal solutions to 3200 randomly produced problems.

Key words: Joint replenishment problem, Multi-objective genetic algorithm, RAND algorithm
doi:10.1631/jzus.C1100384 **Document code:** A **CLC number:** TP301.6

1 Introduction

In this research the joint replenishment problem (JRP) has been studied. The JRP is a multi-item inventory problem in which products have common characteristics; for example, products may use the same transportation devices, or be placed in different packages from a batch production, or be replenished from a common supplier. What characterizes the JRP is that all products have a common ordering or setup cost consisting of two main components: (1) a major ordering cost which is independent of the number of different products in the order and is incurred when at least one product is ordered, and (2) a minor ordering cost which depends on the number of different products in the order and is carried out for each product that exists in the order. Because of this ordering cost structure, using group replenishment will lead to cost savings compared to independent replenishment. The purpose of the JRP is to plan a coordinated replenishment program that minimizes the total cost of ordering and inventory holding. Order quantity and order cycle time of each item are two replenishment schedules in the JRP, which is an NP-hard problem (Khouja *et al*., 2000). Kaspi and Rosenblatt (1983) developed the best-known algorithm for solving the JRP, the RAND algorithm. This algorithm is based on computing m equally spaced values of the cycle time (T) within the upper and lower bounds and then improving the objective function, at each value T. Khouja *et al*. (2000), in a comparison between the genetic algorithm (GA) and the RAND algorithm, showed that GAs can provide better solutions than the RAND for some problems. Some researchers developed the classic JRP under special conditions. Li

(2004) solved a multi-buyer JRP and used the RAND method to solve it. Khouja *et al*. (2005) developed the classic JRP with continuous unit cost decrease. Siajadi *et al*. (2005) used the JRP as a sub-problem and developed a model for a vendor who jointly replenishes *n* parts which produce a finished product. Chan *et al.* (2006) developed a solution approach for optimizing the inventory decisions of many buyers who use the JRP in ordering their inventory from the supplier. Moon and Cha (2006) suggested the classic JRP with resource restriction and then developed the RAND algorithm to solve it. Olsen (2008) solved the classic JRP with interdependence of minor ordering costs by using the evolutionary algorithm (EA). Wang and Cheng (2008) developed a JRP with uncertain demands and inaccurate unit holding cost. Porras and Dekker (2008) presented a new algorithm with polynomial time for the JRP, where a correction is made for empty replenishments. Hsu (2009) used the JRP for a central factory and its satellite factories. Wang *et al.* (2012) proposed a new model with using both direct and indirect grouping for interdependence of minor ordering cost and then a new differential evolution (DE) algorithm to solve it.

In this paper, a special case of the classic JRP is developed in which more than one objective function must be optimized. In most inventory models, all of the requirements such as cost and service levels are summarized into a single objective, but in many real inventory systems decision makers tend to optimize more than one objective function. In multi-objective inventory models in the literature, some important objectives are as follows: minimization of workload and inventory investment and maximization of customer service level (Gardner and Dannenbring, 1979), minimization of wastage cost and maximization of profit (Padmanabhan and Vrat, 1990), minimization of expected annual total relevant cost and minimization of expected annual frequency of stock out (Wee *et al*., 2009), minimization of expected annual total relevant cost and minimization of expected annual frequency of stock out and minimization of the expected annual number of items stocked (Tsou, 2009), maximization of total average profit and minimization of total average wastage (Xu and Liu, 2008; Xu and Zhao, 2010), minimization of total cost and maximization of the yield rate and fixing the replenishment to a desired number (Kang and Lee, 2010).

Unlike other research areas, there is very little research on the JRP with more than one objective function. The model developed in this research has two objective functions and is solved by three proposed algorithms. The main concepts of JRP, the RAND algorithm, multi-objective optimization problems (MOPs), and multi-objective genetic algorithms (MOGAs), are described. To solve the proposed problem, we introduce the developed model (the bi-objective joint replenishment problem), state the modification of the RAND (named B-RAND), describe the use of MOGA, and describe a hybrid of B-RAND and MOGA (named HRG). The three algorithms are used to solve 3200 randomly generated problems.

2 Problem definition

2.1 Joint replenishment problem

As mentioned in the previous section, the JRP is used in an inventory system where multiple items must be jointly coordinated for ordering from the same supplier. In the JRP there are n items such that the ith item (i=1, 2, …, n) has demand D_i per unit time and holding cost h_i per unit time. In this inventory system, each time an order is made, a joint replenishment cost or major ordering cost, S, is incurred regardless of the combination of items in the order. Also, an item-specific order cost, s_i, is incurred for each ith item in the replenishment order. Order quantity of each item, Q_i, and order cycle time of each item, T_i, are two replenishment schedules in the JRP. The objective of JRP is to minimize the total cost of setup (TCS) and total cost of holding (TCH) of inventory per unit time. There are two strategies for solving the JRP, a direct grouping strategy (DGS) and an indirect grouping strategy (IGS). Under DGS, products are partitioned into a predetermined number of sets and the products within each set have the same cycle time. Under IGS a replenishment is made at regular time intervals and each product has an integer multiple of the regular time interval, so in this strategy the decision variables are basic cycle time, T, and an integer number that states the replenishment schedule of item i, k_i, such that

$$T_i = k_i T, \quad Q_i = D_i T_i, \quad \forall i = 1, 2, ..., n. \tag{1}$$

Thus, the JRP model is as follows:

$$\min \mathrm{TC}(T, K_{i,s}) = \mathrm{TCH} + \mathrm{TCS} = \frac{T}{2}\sum_{i=1}^{n} k_i D_i h_i + \frac{1}{T}\left(S + \sum_{i=1}^{n} s_i / k_i\right) \quad (2)$$

subject to

$$T \geq 0,\ k_i \geq 0,\ k_i \text{ is integer},\ \forall i = 1, 2, \ldots, n. \quad (3)$$

2.2 RAND algorithm

As mentioned in the first section, the RAND algorithm is the most popular heuristic for solving JRP. It is based on computing m equally spaced values of the basic cycle time within its upper and lower bounds ($T_{\max}$, $T_{\min}$), obtained from Eqs. (4) and (5) and then applying Silver's improved algorithm at each value of T (Khouja *et al*., 2000).

This algorithm is as follows:

Step 1: Compute $T_{\max}$ and $T_{\min}$ according to the following equations:

$$T_{\min} = \min_{1 \leq i \leq n} \sqrt{s_i / (h_i D_i)}, \quad (4)$$

$$T_{\max} = \left[2\left(S + \sum_{i=1}^{n} s_i \right) \Big/ \sum_{i=1}^{n} D_i h_i \right]^{1/2}. \quad (5)$$

Step 2: Divide the range $[T_{\min}, T_{\max}]$ into m equally spaced values of T (T_j, j=1, 2, ..., m). The value of m must be decided by the decision maker. Set j=0.

Step 3: Set j=j+1 and r=1.

Step 4: Set r=r+1 for T_j and each product i.

$$k_{i,r}^2 = 2 s_i T_j^2 \big/ (D_i h_i). \quad (6)$$

Step 5: Find $k_i^*(r)$ for each i, where

$$k_i^*(r) = L, \ \text{if } L(L-1) < k_i^2(r) \leq L(L+1). \quad (7)$$

Step 6: Compute a new cycle time T_j according to

$$T_j = \left[\left(S + \sum_{i=1}^{n} s_i / k_i^*(r) \right) \Big/ \sum_{i=1}^{n} D_i h_i k_i^*(r) \right]^{1/2}. \quad (8)$$

Step 7: If r=1 or $k_i^*(r) \neq k_i^*(r-1)$ for any i, then go to Step 4; otherwise, compute TC from Eq. (2) for $(T_j, k_1^*(r), k_2^*(r), \ldots, k_n^*(r))$.

Step 8: If j=m, then select $(T_j, k_1^*(r), k_2^*(r), \ldots, k_n^*(r))$ with minimum TC; otherwise, go to Step 3.

2.3 Multi-objective optimization problems

Multi-objective optimization problems (MOPs) usually belong to the subset of complex and nonlinear systems in real engineering cases. The goal of these problems is finding a vector of feasible decision variables to reach a vector of acceptable values for all objective functions (Djeffal and Bendib, 2011). In many MOPs, objectives conflict with each other, so finding a perfect solution that simultaneously optimizes all objective functions is rarely possible. One practical approach to solving MOPs is determination of Pareto optimal solutions (POSs). A POS is a solution that dominates when no solution in the feasible region performs better in at least one objective and equivalently or better in the other objectives. Since finding all POSs is not practically feasible, the primary goal of multi-objective algorithms is introducing a representative subset of the POS set. There are two main approaches for this purpose: (1) ideal approach and (2) preference based approach. In the ideal approach the focus is first finding a set of POSs and then selecting one solution from the POSs by taking more information from the decision maker. Multi-objective evolutionary optimization algorithms (MOEAs) are based on this approach. In the preference approach, the goal is to find one, instead of all, of the POSs. Classical multi-objective optimization algorithms use this approach (Singh and Khare, 2011).

2.4 Multi-objective genetic algorithm

There are several ways to solve MOPSs based on their selection mechanism. These approaches can be classified into three groups: (1) aggregation function approaches; (2) population-based approaches, and (3) Pareto-based approaches.

Aggregation function approaches linearly or nonlinearly combine all the objective functions of the problem into a single function. In population-based approaches, the diversity of the search is obtained by using the population of an evolutionary algorithm (EA). The classical example of these approaches is the vector evaluated genetic algorithm (VEGA),

which consists of a simple GA with a modified selection mechanism. In this algorithm, for a problem with k objectives, k sub-populations are generated. These sub-populations are then added together to obtain a new population, and then crossover and mutation are performed by applying the GA. In this regard, one of the new practical approaches is the particle swarm optimization (PSO) algorithm, which can be used to solve difficult multi-dimensional optimization problems in various fields (Ozkaya and Gunes, 2012; Sahoo *et al*., 2012). Pareto-based approaches are based on MOEAs that incorporate the concept of Pareto optimality in their selection mechanism. In these methods multiple optimal solutions can be found in a single simulation run, instead of by applying the method many times (Qu and Suganthan, 2010; Sundar *et al*., 2010). A GA with a single objective can be modified to find a set of non-dominated solutions in a single run. The ability of GA to simultaneously search different regions of a solution space makes it possible to find a diverse set of solutions for difficult problems (Fung *et al*., 2012).

The multi-objective genetic algorithm (MOGA) used in this research is one of the most famous approaches in the Pareto-based approaches introduced by Fonesca and Fleming (1993). An MOEA is an extension of an evolutionary algorithm in which the following two mechanisms must be stated: (1) selection mechanism (such that the dominated solution has less chance relative to non-dominated solutions); (2) diversity maintenance mechanism (such that as much of the Pareto optimal set as possible can be evaluated in the population).

For the first mechanism in MOGA, the rank of each solution in the population is equal to the number of other solutions that dominate it. All of the non-dominated solutions have the same rank, and thus they have the same probability of being selected for the next generation. For the second mechanism, MOGA uses a fitness sharing approach so that solutions uniformly distributed over the Pareto front can be obtained. A niche-formation for reducing fitness of solutions in densely populated areas is used for fitness sharing. A similarity threshold (σ_{share}) by a relatively simple method determines the radius of each niche (Konak *et al*., 2006).

3 New model: bi-objective joint replenishment problem

In this section we introduce the notations and formulation used in the proposed model. All assumptions, decision variables, input parameters, and objective functions are stated.

3.1 Assumptions

The assumptions of our bi-objective JRP model are similar to those of the classical JRP, which are as follows:

1. Parameters are deterministic and known.
2. Shortages are not allowed.
3. Quantity discounts are not permitted.
4. Holding cost is linear and is calculated based on average of the stored inventory.
5. The IGS strategy is used.

And in our model, also:

6. Transportation cost is based on the maximum of the stored inventory and is not a part of holding cost.

3.2 Input parameters

i: the index of items, i=1, 2, …, n.

D_i: the demand rate of item i.

S: the major ordering cost.

s_i: the minor ordering cost of item i.

h_i: the inventory holding cost of item i per unit time.

tr_i: inventory transportation cost of item i.

3.3 Decision variables

T: the basic cycle time.

k_i: the integer number that states the replenishment schedule of item i.

3.4 Bi-objective JRP formulation

Our proposed model has two objective functions. The first objective is the same objective as the JRP, that is, minimization of the total cost of setup (TCS) and total cost of holding (TCH) of inventory per unit time. We represent this objective function with F_1. On the other hand, in many inventory problems in real cases, transportation cost is large and independent of time and thus is not a part of holding or setup cost, so for more applicability of our model, we propose that transportation cost is based on the maximum of stored

inventory. Note that holding cost in the first objective is based on average of stored inventory. Thus, the second objective function of our model is minimization of the total transportation cost of inventory that we represent with F_2.

The proposed model is as follows:

$$\min F_1(T,K_{i,s}) = \mathrm{TCH} + \mathrm{TCS} = \frac{T}{2}\sum_{i=1}^{n} k_i D_i h_i + \frac{1}{T}\left(S + \sum_{i=1}^{n} s_i/k_i\right) \quad (9)$$

$$\min F_2(T,K_{i,s}) = T\sum_{i=1}^{n} \mathrm{tr}_i k_i D_i \quad (10)$$

subject to

$$T\geq 0,\ k_i\geq 1,\ k_i \text{ is integer},\ \forall i=1, 2, \ldots, n. \quad (11)$$

4 Three new algorithms for solving the proposed model

4.1 Modified RAND algorithm

In this subsection, the RAND method is modified for solving the proposed model.

4.1.1 Models and mathematical relations

To modify the RAND algorithm for solving the JRP with two objective functions, we need to use the l^p-metric method with p=1. It is assumed that w_1, w_2 are the weights of the two objectives. Solving the problem for a given weight factor $W=\{w_1, w_2\}$ yields a single solution. To obtain a Pareto optimal set, we must solve the problem multiple times with different weight combinations randomly generated at each run. Using the l^p-metric method, the two objective functions of the main model are combined into one objective function as follows:

$$\min F_3(T,K_{i,s}) = w_1\frac{F_{1\max}-F_1}{F_{1\max}-F_{1\min}} + w_2\frac{F_{2\max}-F_2}{F_{2\max}-F_{2\min}}. \quad (12)$$

If we assume

$$w_1' = \frac{-w_1}{F_{1\max}-F_{1\min}}, \quad (13)$$

and

$$w_2' = \frac{-w_2}{F_{2\max}-F_{2\min}}, \quad (14)$$

then the modified model is as follows:

$$\min F_3(T,K_{i,s}) = w_1'\left(F_1(T,K_{i,s})\right) + w_2'\left(F_2(T,K_{i,s})\right) = \frac{w_1'T}{2}\sum_{i=1}^{n} k_i D_i h_i + \frac{w_1'}{T}\left(S + \sum_{i=1}^{n} s_i/k_i\right) + w_2'T\sum_{i=1}^{n} \mathrm{tr}_i k_i D_i \quad (15)$$

subject to

$$T\geq 0,\ k_i\geq 1,\ k_i \text{ is integer},\ \forall i=1, 2, \ldots, n.$$

For a fixed value of k_i ($\forall i$=1, 2, …, n), $\dfrac{\partial F_3(T,K_{i,s})}{\partial T}$ =0 yields

$$T(K_{i,s}) = \left[\frac{2w_1'\left(S + \sum_{i=1}^{n} s_i/k_i\right)}{\sum_{i=1}^{n} D_i k_i (w_1' h_i + 2w_2' \mathrm{tr}_i)}\right]^{1/2}. \quad (16)$$

Since $T(K_{i,s})$ is decreasing in $(k_1, k_2, ..., k_n)\in\mathbb{N}^n$, if we assume k_i=1 ($\forall i$=1, 2, …, n), we obtain

$$T_{\max} = \left[\frac{2w_1'\left(S + \sum_{i=1}^{n} s_i\right)}{\sum_{i=1}^{n} D_i (w_1' h_i + 2w_2' \mathrm{tr}_i)}\right]^{1/2}. \quad (17)$$

On the other hand, for a fixed value of T, $\dfrac{\partial F_3(T,K_{i,s})}{\partial K_i} = 0$ ($\forall i$=1, 2, …, n) gives

$$k_i^2 = \frac{2w_1' s_i}{T^2 (w_1' h_i + 2w_2' \mathrm{tr}_i) D_i}. \quad (18)$$

Thus,

$$T_{\min} = \min_{1\leq i\leq n}\left[\frac{w_1' s_i}{(w_1' h_i + 2w_2' \mathrm{tr}_i) D_i}\right]^{1/2}. \quad (19)$$

By using $T_{\max}$ and $T_{\min}$ given by Eqs. (17) and (19) respectively, we modify the RAND method and develop a new algorithm for solving the multi-objective JRP. In Section 4.1.2 the bi-objective RAND algorithm, which will be referred to as B-RAND from this point forward, is described.

4.1.2 B-RAND algorithm

Step 0: t=1 (t is the counter of iteration).

Step 1: Generate 100 random feasible solutions and calculate $F_{1\min}$, $F_{1\max}$, $F_{2\min}$, $F_{2\max}$.

Step 2: Generate random w_1, w_2 within [0, 1].

Step 3: Calculate w_1', w_2' using Eqs. (13) and (14), respectively.

Step 4: Compute $T_{\max}$ and $T_{\min}$ using Eqs. (17) and (19), respectively.

Step 5: Divide the range $[T_{\min}, T_{\max}]$ into m equally spaced value of T (T_j, j=1, 2, …, m). The value of m is to be decided by the decision maker. Set j=0.

Step 6: Set j=j+1 and r=1.

Step 7: Set r=r+1 for T_j, and for each product i compute $k_i^2(r)$ using Eq. (18).

Step 8: Find $k_i^*(r)$ for each i, where $k_i^*(r)=L$ if $L(L-1)<k_i^2(r)\le L(L+1)$.

Step 9: Compute a new cycle time T_j for $k_i^*(r)$ using Eq. (16).

Step 10: If r=1 or $k_i^*(r)\ne k_i^*(r-1)$ for any i, then go to Step 4; otherwise, compute $F_3(T, K_{i,s})$ for (T_j, $k_1^*(r)$, $k_2^*(r)$, …, $k_n^*(r)$).

Step 11: If j=m then select (T_j, $k_1^*(r)$, $k_2^*(r)$, …, $k_n^*(r)$) with minimum $F_3(T, K_{i,s})$ and add it to a pool of the best solutions, and select $F_{1\min}$, $F_{1\max}$, $F_{2\min}$, $F_{2\max}$ for the next iteration; otherwise, go to Step 6.

Step 12: t=t+1. If t<M (M is the number of iterations), go to Step 2; otherwise, select the Pareto optimal set from the pool of the best solutions. The value of M must be decided by the decision maker.

4.2 Multi-objective genetic algorithm

In this subsection, an MOGA is developed for solving the proposed model. In the following subsections, the basic components of the proposed algorithm are presented.

4.2.1 Variable bounds

As mentioned earlier, the basic cycle time (T) and integer multiplier of the basic cycle time for each product ($k_{i,s}$) are decision variables in our model. The upper and lower bounds of T are as follows (Khouja *et al*., 2000):

$$\begin{cases} T_{\min}=\min\limits_{1\le i\le n}\sqrt{2s_i/(h_iD_i)}, \\ T_{\max}=\left[2\left(S+\sum\limits_{i=1}^{n}s_i\right)\Big/\sum\limits_{i=1}^{n}D_ih_i\right]^{1/2}, \end{cases} \tag{20}$$

$$k_{i\min}=1,\quad \forall i=1,2,\ldots,n, \tag{21}$$

$$k_{i\max}=\lceil T_{i\text{in}}/T_{\min}\rceil, \tag{22}$$

such that

$$T_{i\text{in}}=\left(2(S+s_i)/h_i\right)^{1/2}. \tag{23}$$

4.2.2 Representation

In our proposed MOGA, a solution or individual consists of n+1 genes among which the first n genes represent the values of $K_{i,s}$ and the last gene shows the basic cycle time (T).

4.2.3 Initial population

A random number generator introduces the population such that for each chromosome, the algorithm produces n integer values for $K_{i,s}$ between $k_{i\min}$ and $k_{i\max}$ and one value for T between $T_{\min}$ and $T_{\max}$.

4.2.4 Fitness function

The fitness value of each solution in the population is calculated by using the following procedure (Konak *et al*., 2006):

Step 1: Assign a rank $r(x, t)$ to each solution $x\in p_t$:

$$r(x, t)=1+\text{nq}(x, t) \tag{24}$$

such that nq(x, t) is the number of solutions that dominate solution x at generation t.

Step 2: Assign a fitness value to each solution based on the solution's rank:

$$f(x,t)=N-\sum_{k=1}^{r(x,t)}n_k-0.5\left(n_r(x,t)-1\right), \tag{25}$$

where n_k is the number of solutions with rank k and N is the population size.

Step 3: Calculate the nich count nc(x, t) of each solution $x\in p_t$:

$$\text{nc}(x,t)=\sum_{y\in p,r(y,t)=r(x,t)}\max\left(\frac{\sigma_{\text{share}}-dF(x,y)}{\sigma_{\text{share}}},0\right), \tag{26}$$

where $dF(x, y)$, a value between 0 and 1, is the Euclidean distance between every solution pair of x and y in the normalized objective space, calculated by

$$dF(x,y)=\sqrt{\left(\frac{F_1(x)-F_1(y)}{F_{1\max}-F_{1\min}}\right)^2+\left(\frac{F_2(x)-F_2(y)}{F_{2\max}-F_{2\min}}\right)^2}, \tag{27}$$

where $F_{k\max}$ and $F_{k\min}$ (k=1, 2) are the maximum and minimum values of the objective function F_k

observed so far during the search, respectively.

Step 4: Calculate the shared value of each solution $x \in p_t$:

$$f^{(1)}(x,t) = f(x,t) / \mathrm{nc}(x,t). \tag{28}$$

Step 5: Normalize the fitness values using the shared fitness values:

$$f^{(2)}(x,t) = \frac{f^{(1)}(x,t) n_r(x,t)}{\sum_{y \in p_t} f^{(1)}(x,t)} f(x,t). \tag{29}$$

4.2.5 Reproduction

The selection method in our proposed algorithm is based on a roulette-wheel selection mechanism in which the chance of the ith solution with fitness value $f_i^{(2)}$ for selection in the next generation is $f_i^{(2)} \Big/ \sum_{i \in N} f_i^{(2)}$.

4.2.6 Crossover and mutation

A random-point crossover operator is used for producing offspring from the parents selected from the previous generation. It acts with probability p_c (probability of crossover) and selects a random number of genes from two parents with the same locations and exchanges them. Using a mutation operator, the value of each gene changes to a random value between its lower and upper bounds. This operator acts with low probability p_m (probability of mutation) after crossover.

4.2.7 Stopping condition

Our proposed algorithm will be stopped when the pool of POSs does not change in 50 generations.

4.2.8 Procedure of MOGA

Step 1: Start with a random initial population p_0, and set t=0.

Step 2: If the stopping criterion is satisfied, return p_t.

Step 3: Evaluate fitness of the population ($f^{(2)}(x, t)$).

Step 4: Use a roulette-wheel selection method based on $f^{(2)}$ to select parents for the mating pool.

Step 5: Apply crossover and mutation on the mating pool until offspring population Q_t of size N is filled.

Step 6: Set $P_{t+1}=Q_t$.

Step 7: Update the pool of POSs.

Step 8: Set $t=t+1$, and go to Step 2.

4.3 Hybrid of B-RAND and MOGA (HRG)

The combination of different algorithms for solving combinatorial optimization problems is usually powerful. Thus, the hybrid of B-RAND and MOGA for the proposed model, referred to as the HRG algorithm, is used. The main algorithm is MOGA for this hybridization; thus, the main concepts are the same as described in Section 4.2 except for the mechanism of generating the initial population. In our proposed hybrid method, we first solve the problem by B-RAND and use its final solutions as the initial solution to MOGA and then solve the problem again by MOGA.

5 Computational study

We divided the testing of the performances of the proposed algorithms into two parts. In the first part we compared our algorithms with the algorithms in the literature. Because there is no JRP with more than one objective function, we supposed the value to be zero for the second objective function (F_2). The value of the first objective function was reached by each of the three proposed methods. The final solutions were then compared with the RAND method (Kaspi and Rosenblatt, 1991). In the second part the performances of the three algorithms for providing POSs were compared with each other. In each part, 1600 problems were randomly generated from the same distribution which is used to test the RAND performance (Kaspi and Rosenblatt, 1991). Each generated problem has six parameters (Table 1). Parameter values of the three algorithms are shown in Table 2. The values of MOGA are the same as those used to test GA (Khouja *et al*., 2000). All algorithms were coded in Microsoft Visual Basic 6.5. For observing the performance and output of the three algorithms in each group of N and S, one randomly generated problem was solved and the results, i.e., the POSs, are shown in Figs. 1a–1d. Table 3 shows the results of the first comparison and Table 4 and Figs. 2a–2f show the results of the second comparison. In Tables 3 and 4, for each row (altogether 16 different combinations of n and S) 100 randomized problems were produced.

Table 1 Parameter values of instances

Parameter	Value(s) or range
Demand, D_i	[100, 100 000]
Minor ordering cost, s_i	[0.5, 5]
Holding cost, h_i	[0.2, 3]
Number of products, n	10, 20, 30, 50
Major ordering cost, S	5, 10, 15, 20
Transportation cost of item, tr_i	1

Table 2 Parameter values of the algorithms

Parameter	Value	Algorithms
Number of spaced values of the basic cycle time, m	10	B-RAND and HRG
Population size	100	MOGA and HRG
Probability of crossover	0.6	MOGA and HRG
Probability of mutation	0.2	MOGA and HRG
Similarity threshold, σ_{share}	1	MOGA and HRG

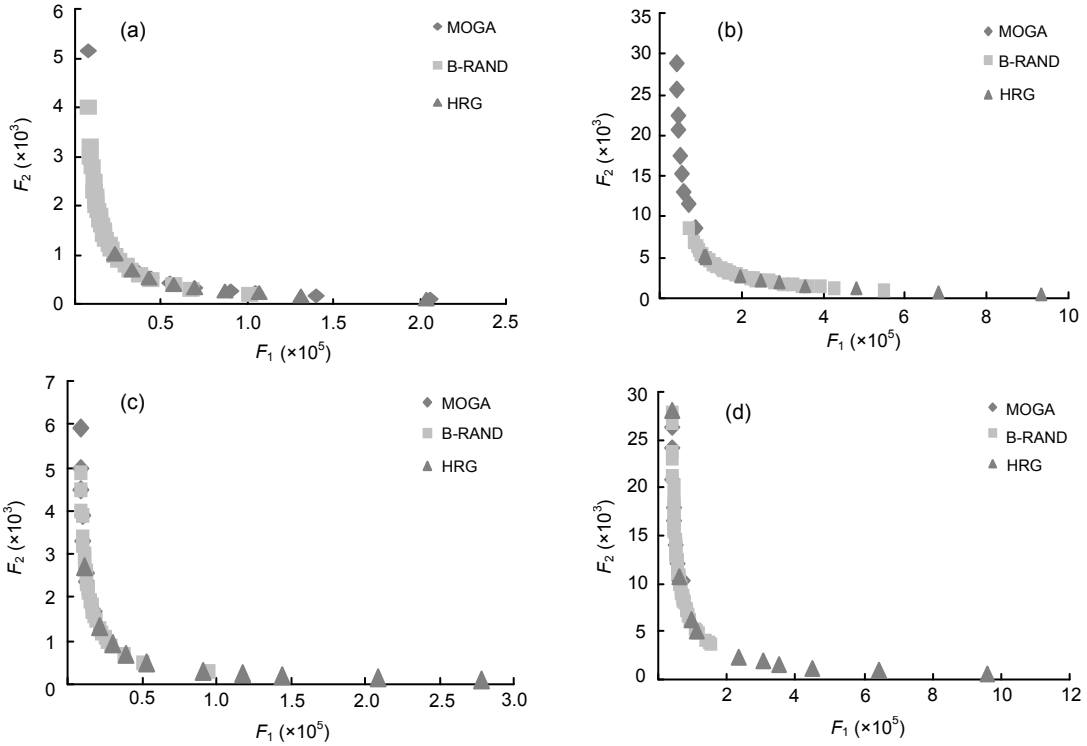

Fig. 1 Pareto optimal solutions of a randomized problem with n=10, S=5 (a), n=50, S=20 (b), n=10, S=20 (c), and n=50, S=5 (d)

n: number of products; S: major ordering cost. F_1: the first objective function; F_2: the second objective function

6 Results and discussion

To test the performances of our proposed three algorithms, we show the results of the 3200 problems randomly generated from the same distribution. As mentioned in the previous section, we have solved a randomly generated problem in each of four groups of N and S to compare the performance and output of the algorithms. As shown in Figs. 1a–1d, MOGA provided POSs with better F_1, HRG provided POSs with better F_2, and RAND obtained POSs with F_1 and F_2 between MOGA and HRG. On the other hand, the number of POSs obtained using B-RAND was larger than those obtained using HRG and MOGA. As shown in these figures, aggregation of the POSs of the three algorithms can be a useful set for selecting the most utilized solutions by the decision maker..

As stated earlier, in the first part of our testing, the value for F_2 was supposed to be zero and the results have been compared with the RAND method (Kaspi and Rosenblatt, 1991). The results are shown in Table 3. Previous tests (except for n=50) on RAND delivered the optimal solution for 83.4% of problems. Our proposed algorithms performed well compared to RAND. For 1088 (68%) of the 1600 randomly produced problems, B-RAND reached a solution with the same or better objective function (F_1) than MOGA or HRG. For these 1088 problems, the maximum

Table 3 Comparison between B-RAND, MOGA, and HRG solutions for F_1

n	S	Percentage of RAND optimum* (%)	Problem percentage (%)			Maximum improvement (%)			Average improvement (%)		
			Type I	Type II	Type III	Type I	Type II	Type III	Type I	Type II	Type III
10	5	95.2	56	29	15	27.16	9.29	9.29	8.44	2.65	2.55
	10	98.1	64.2	24.5	11.3	23.90	19.10	8.60	12.90	2.70	1.60
	15	99.5	70.4	20.4	9.3	38.36	21.48	2.53	13.50	4.10	0.87
	20	99.8	67.3	20.0	12.7	33.10	11.50	1.70	11.40	2.60	0.90
20	5	76.2	53	32	16	13.08	3.17	7.79	4.03	0.71	1.86
	10	84.8	79	12	9	19.07	0.12	12.29	5.05	0.05	4.59
	15	88.2	65	19	16	15.96	0.84	0.31	4.27	0.33	0.15
	20	91.1	63	20	17	17.97	0.30	0.27	4.05	0.09	0.10
30	5	53.5	53	39	8	10.33	1.81	0.51	2.83	0.69	0.25
	10	65.5	72	22	6	13.33	0.93	0.71	2.69	0.40	0.44
	15	73.8	76	12	12	10.76	0.43	1.49	2.77	0.14	0.43
	20	75.3	69	19	11	14.90	0.60	4.90	2.20	0.10	2.10
50	5		81	13	6	22.89	2.17	0.77	3.64	1.00	0.38
	10		76	9	15	13.56	0.84	0.70	3.40	0.49	0.22
	15		67	21	13	23.49	1.42	1.42	4.32	0.38	0.32
	20		84	6	9	20.74	0.02	2.45	4.53	0.01	0.88
Maximum			84.38	38.89	17.07	38.36	21.48	12.29	13.50	4.10	4.59
Average		83.4	68.5	19.9	11.6	19.91	4.63	3.48	5.63	1.03	1.10

* m=10. m: number of spaced values of the basic cycle time; n: number of products; S: major ordering cost. For each pair of (n, S), 100 randomized problems were produced. Type I problems: F_1(RAND)≤F_1(GA) and F_1(RAND)≤F_1(HRG); Type II problems: F_1(GA)≤F_1(RAND) and F_1(GA)≤F_1(HRG); Type III problems: F_1(HRG)≤F_1(RAND) and F_1(HRG)≤F_1(GA)

percentage of saving in the objective function provided by B-RAND was 38.36% and the average percentage of saving was 5.63%. B-RAND and HRG did not outperform MOGA for 320 (20%) problems, and the maximum and average percentages of saving in total cost (F_1) provided by MOGA were 21.48% and 1.03%, respectively. HRG was better than or equal to the other algorithms for 192 (12%) problems and HRG provided 12.29% and 1.10% of the maximum and average percentages of saving in the total cost, respectively.

Table 4 shows the comparison among the performances of the three algorithms, summarizing results of solving the 1600 randomly generated problems. By solving each problem some POSs were obtained. Obviously, a better algorithm has a lower value of AVG(F_j)/min(AVG(F_j)) and AVG(time)/min(AVG(time)) and an upper value of VAR(F_j)/min(VAR(F_j)). These values based on rows of Table 4 have been plotted in Figs. 2a–2f.

Fig. 2a shows that, for F_1, B-RAND provided minimum values and MOGA and HRG obtained almost the same performance. For larger values of S and n, HRG was superior to MOGA. Fig. 2b shows that, for F_2 MOGA and HRG had almost the same and the best performance and B-RAND had better performance for larger problems. Fig. 2c shows that B-RAND had the minimum time for solving problems. For larger problems HRG was faster than MOGA, but for smaller problems MOGA was faster than HRG. Figs. 2d and 2e show that HRG gave POSs with the maximum diversity and that POSs of B-RAND had the minimum diversity. Fig. 2f shows that MOGA and HRG had the same number of POSs, but lower than B-RAND.

7 Conclusions

In this paper, we model a bi-objective joint replenishment problem. Shortcomings of previous models on the JRP under special conditions and on multi-objective inventory models are reviewed, and a new model is developed. Our proposed model has two objective functions. The first objective function is the same as the classical JRP objective function, and the second is minimization of transportation cost. As our proposed model is NP-hard, three new algorithms, including a modified RAND algorithm (B-RAND), a multi-objective genetic algorithm (MOGA), and a

Table 4 Comparison of the performances of our proposed three algorithms

n	S	AVG(F_1)/min(AVG(F_1))			AVG(F_2)/min(AVG(F_2))			AVG(time)/min(AVG(time))		
		MOGA	B-RAND	HRG	MOGA	B-RAND	HRG	MOGA	B-RAND	HRG
10	5	1.1	1	1.4	1	6.87	1.09	4.88	1	5.92
	10	1.1	1	1.6	1	8.67	1.00	4.85	1	4.77
	15	1.4	1	1.5	1	7.48	1.02	4.52	1	5.35
	20	1.4	1	2.0	1.12	6.45	1.00	3.87	1	4.87
20	5	1.2	1	1.6	1.2	4.27	1.00	5.87	1	4.43
	10	1.5	1	1.7	1	5.56	1.08	3.19	1	5.13
	15	1.2	1	1.4	1	4.76	1.00	3.07	1	4.00
	20	1.3	1	1.6	1	6.68	1.02	3.36	1	4.29
30	5	1.3	1	1.3	1.03	3.15	1.00	3.11	1	4.01
	10	1.3	1	1.4	1	5.50	1.08	2.90	1	5.38
	15	1.3	1	1.7	1	4.23	1.01	2.69	1	3.75
	20	1.5	1	1.6	1.05	3.61	1.00	2.86	1	3.79
50	5	1.7	1	1.6	1	3.17	1.19	5.05	1	3.01
	10	1.3	1	1.5	1	4.34	1.32	6.03	1	3.15
	15	1.6	1	1.3	1	4.82	1.16	5.83	1	3.15
	20	1.5	1	1.4	1	6.28	1.33	6.25	1	3.25
Average		1.4	1	1.5	1	5.40	1.10	4.27	1	4.26

n	S	VAR(F_1)/min(VAR(F_1))			VAR(F_2)/min(VAR(F_2))			Average number of POSs		
		MOGA	B-RAND	HRG	MOGA	B-RAND	HRG	MOGA	B-RAND	HRG
10	5	1	2	147 411	3.94	1	14.55	10	30.85	10
	10	1	66 228	1 178 865	1	6.20	15.96	10	32.80	10
	15	1	136	252 178	6.19	1	9.50	10	32.00	10
	20	1	454 162	82 143	3.13	1.41	1.00	9.95	33.30	10
20	5	1	91 807	332 348	5.54	1	14.74	9.95	36.25	9.95
	10	1	862	285 392	7.97	1	19.66	9.85	37.15	9.95
	15	1	228	2 002 203	1.40	1	15.87	10	39.05	9.85
	20	1	16 150	682 650	1	2.95	11.32	10	38.75	10
30	5	1	58	962 626	1.90	1	11.47	9.9	40.20	9.95
	10	1	869	630 357	6.24	1	12.57	10	38.80	9.95
	15	1	1 368 084	545 522	1	2.32	1.60	9.95	41.80	10
	20	1	610 275	289 850	1	1.29	1.74	10	41.15	9.95
50	5	1	15 932	1 286 692	1	7.88	13.04	9.8	43.75	9.95
	10	1	1 473 002	522 452	1	1.62	3.97	9.95	44.15	9.85
	15	1	19 421	328 268	3.46	1	19.62	9.95	44.30	10
	20	1	317 236	1 774 754	1.64	1	3.97	10	43.45	9.95
Average		1	277 153	706 482	2.96	2.04	10.66	9.96	38.61	9.96

For each pair of (n, S), 100 randomized problems were produced. n: number of products; S: major ordering cost

hybrid of the two previous algorithms (HRG) are proposed. To validate these algorithms, some test problems are designed and solved. For comparing the performances of the proposed algorithms with those of the previous approaches, it is supposed that the second objective function is equal to zero. The abilities of algorithms in finding good solutions are shown. We also compare the performances of our proposed three algorithms with each other, and the results show that B-RAND obtains POSs with better F_1 but better F_2 is obtained using MOGA and HRG. More diversified solutions are obtained from HRG

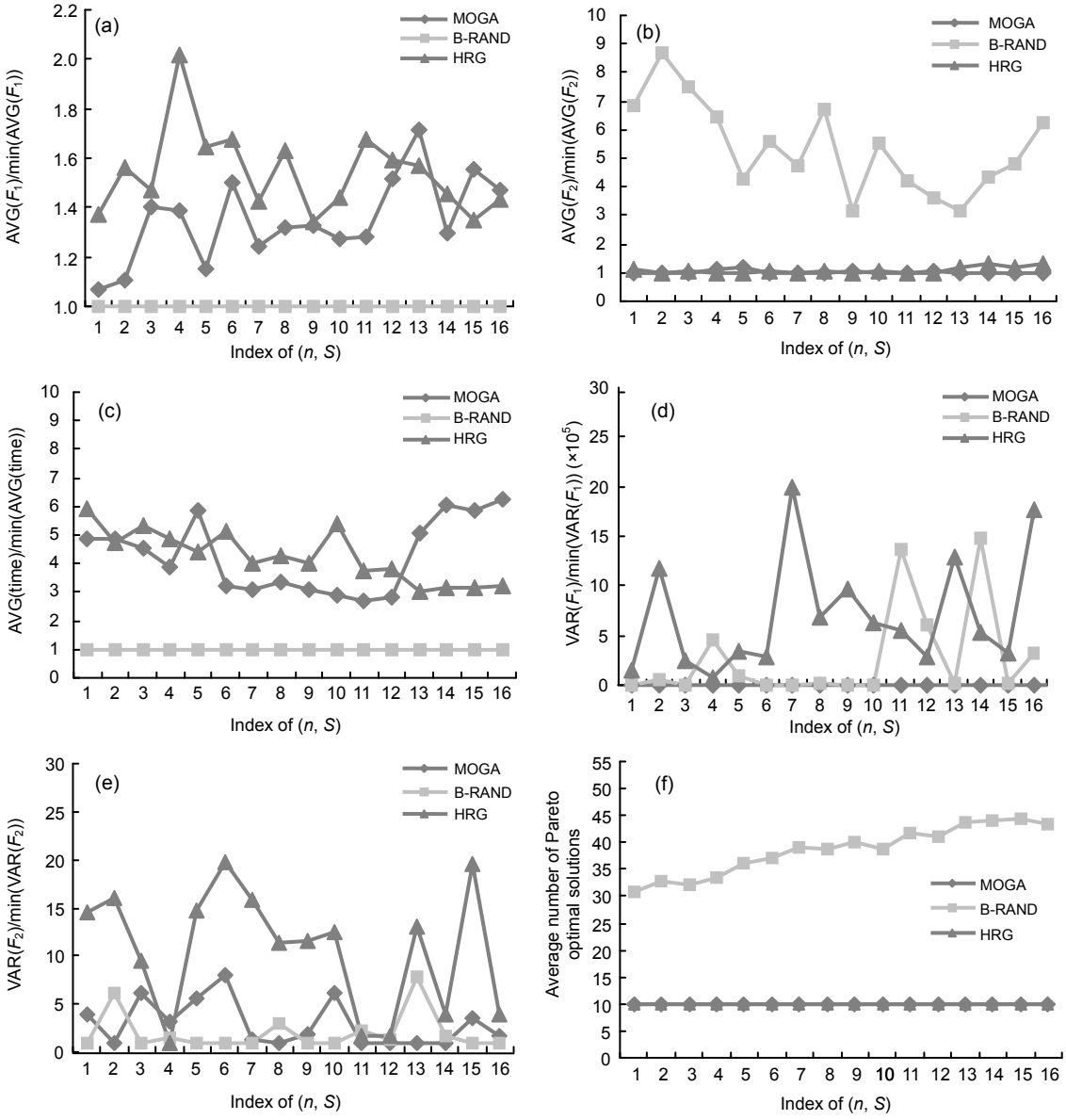

Fig. 2 Comparison of the values of AVG(F_1)/min(AVG(F_1)) (a), AVG(F_2)/min(AVG(F_2)) (b), AVG(time)/min(AVG(time)) (c), VAR(F_1)/min(VAR(F_1)) (d), VAR(F_2)/min(VAR(F_2)) (e), and the average number of Pareto optimal solutions (f)
The x-axis represents the index of the pair of (n, S) in Tables 3 and 4, for example, index 1 for n=10, S=5, index 2 for n=10, S=10, and index 16 for n=50, S=20

and MOGA, which means that MOGA and HRG find more POSs than B-RAND. The proposed model has no constraint; a constraint such as space or budget, however, can be used to increase its applicability. Moreover, it seems that using stochastic or dynamic demand in place of deterministic demand can be another area of future research. In our model the IGS strategy has been used. For future research, the DGS strategy can be used. Considering more than one buyer and modeling a multi-buyer multi-objective JRP can be another work for future research. In addition, to improve the convergence or computational time of solution algorithms, using better solution methods such as population-based methods with more capability can provide a good opportunity for future study. As an example, the PSO algorithm modified by Ozkaya and Gunes (2012) can be used for solving multi-objective optimization problems.

References

Chan, C.K., Li, L.Y., Ng, C.T., Cheung, B.K., Langevin, A., 2006. Scheduling of multi-buyer joint replenishments. *Int. J. Prod. Econ.*, **102**(1):132-142. [doi:10.1016/j.ijpe.2005.02.005]

Djeffal, F., Bendib, T., 2011. Multi-objective genetic algorithms based approach to optimize the electrical performances of the gate stack double gate (GSDG) MOSFET. *Microelectron. J.*, **42**(5):661-666. [doi:10.1016/j.mejo.2011.03.003]

Fonesca, C.M., Fleming, P.J., 1993. Multi Objective Genetic Algorithm. IEE Colloquium on Genetic Algorithms for Control System Engineering, Digest No. 1993/20.

Fung, K.Y., Kwong, C.K., Siu, K.W.M., Yu, K.M., 2012. A multi-objective genetic algorithm approach to rule mining for affective product design. *Exp. Syst. Appl.*, **39**(8):7411-7419. [doi:10.1016/j.eswa.2012.01.065]

Gardner, E.S., Dannenbring, D.G., 1979. Using optimal policy surfaces to analyze aggregate inventory trade offs. *Manag. Sci.*, **25**(8):709-720. [doi:10.1287/mnsc.25.8.709]

Hsu, S.L., 2009. Optimal joint replenishment decisions for a central factory with multiple satellite factories. *Exp. Syst. Appl.*, **36**(2):2494-2502. [doi:10.1016/j.eswa.2008.01.069]

Kang, H., Lee, A., 2010. Inventory replenishment model using fuzzy multiple objective programming: a study of a high-tech company in Taiwan. *Appl. Soft Comput.*, **10**(4):1108-1118. [doi:10.1016/j.asoc.2009.11.035]

Kaspi, M., Rosenblatt, M.J., 1983. An improvement of Silver's algorithm for the joint replenishment problem. *IIE Trans.*, **15**(3):264-269. [doi:10.1080/05695558308974644]

Kaspi, M., Rosenblatt, M.J., 1991. On the economic ordering quantity for jointly replenished items. *Int. J. Prod. Res.*, **29**(1):107-114. [doi:10.1080/00207549108930051]

Khouja, M., Michalewicz, M., Satoskar, S., 2000. A comparison between genetic algorithms and the RAND method for solving the joint replenishment problem. *Prod. Plan. Control*, **11**(6):556-564. [doi:10.1080/095372800414115]

Khouja, M., Park, S., Saydam, C., 2005. Joint replenishment problem under continuous unit cost change. *Int. J. Prod. Res.*, **43**(2):311-326. [doi:10.1080/00207540420002703 68]

Konak, A., Coit, D.W., Smith, A.E., 2006. Multi-objective optimization using genetic algorithms: a tutorial. *Rel. Eng. Syst. Safety*, **91**(9):992-1007. [doi:10.1016/j.ress.2005.11.018]

Li, Q., 2004. Solving the multi-buyer joint replenishment problem with the RAND method. *Comput. Ind. Eng.*, **46**(4):755-762. [doi:10.1016/j.cie.2004.05.008]

Moon, K., Cha, B.C., 2006. The joint replenishment problem with resource restriction. *Eur. J. Oper. Res.*, **173**(1):190-198. [doi:10.1016/j.ejor.2004.11.020]

Olsen, A., 2008. Inventory replenishment with interdependent ordering: an evolutionary algorithm solution. *Int. J. Prod. Econ.*, **113**(1):359-369. [doi:10.1016/j.ijpe.2007.09.004]

Ozkaya, U., Gunes, F., 2012. A modified particle swarm optimization algorithm and its application to the multi-objective FET modeling problem. *Turk. J. Electr. Eng. Comput. Sci.*, **20**(2):263-271. [doi:10.3906/elk-1102-1032]

Padmanabhan, G., Vrat, P., 1990. Analysis of multi-item inventory systems under resource constrains: a non-linear goal programming approach. *Eng. Costs Prod. Econ.*, **20**(2):121-127. [doi:10.1016/0167-188X(90)90096-Z]

Porras, E., Dekker, R., 2008. A solution method for the joint replenishment problem with correction factor. *Int. J. Prod. Econ.*, **113**(2):834-851. [doi:10.1016/j.ijpe.2007.11.008]

Qu, B.Y., Suganthan, P.N., 2010. Multi-objective differential evolution with diversity enhancement. *J. Zhejiang Univ.-Sci. C (Comput. & Electron.)*, **11**(7):538-543. [doi:10.1631/jzus.C0910481]

Sahoo, N.C., Ganguly, S., Das, D., 2012. Multi-objective planning of electrical distribution systems incorporating sectionalizing switches and tie-lines using particle swarm optimization. *Swarm Evol. Comput.*, **3**:15-32. [doi:10.1016/j.swevo.2011.11.002]

Siajadi, H., Ibrahim, R.N., Lochert, P.B., Chan, W.M., 2005. Joint replenishment policy in inventory production systems. *Prod. Plan. Control*, **16**(3):255-262. [doi:10.1080/09537280500033213]

Singh, D.P., Khare, A., 2011. Different aspects of evolutionary algorithms, multi-objective optimization algorithms and application domain. *Int. J. Adv. Network. Appl.*, **2**(4):770-775.

Sundar, D., Umadevi, B., Alagarsamy, K., 2010. Multi-objective genetic algorithm for the optimized resource usage and the prioritization of the constraints in the software project planning. *Int. J. Comput. Appl.*, **3**(3):1-4. [doi:10.5120/718-1010]

Tsou, C., 2009. Evolutionary Pareto optimized for continuous review stochastic inventory system. *Eur. J. Oper. Res.*, **195**(2):364-371. [doi:10.1016/j.ejor.2008.02.039]

Wang, L., He, J., Wu, D., Zeng, Y.R., 2012. A novel differential evolution algorithm for joint replenishment problem under interdependence and its application. *Int. J. Prod. Econ.*, **135**(1):190-198. [doi:10.1016/j.ijpe.2011.06.015]

Wang, Y.C., Cheng, W.T., 2008. A sensitivity analysis of solving joint replenishment problems using the RAND method under inaccurate holding cost estimates and demand forecasts. *Comput. Ind. Eng.*, **55**(1):243-252. [doi:10.1016/j.cie.2007.12.010]

Wee, H.M., Lo, C.C., Hsu, P.H., 2009. A multi-objective joint replenishment inventory model of deteriorated items in a fuzzy environment. *Eur. J. Oper. Res.*, **197**(2):620-631. [doi:10.1016/j.ejor.2006.08.067]

Xu, J., Liu, Y., 2008. Multi-objective decision making model under fuzzy random environment and its application to inventory problems. *Inf. Sci.*, **178**(14):2899-2914. [doi:10.1016/j.ins.2008.03.003]

Xu, J., Zhao, L., 2010. A multi-objective decision-making model under fuzzy rough coefficient and its application to inventory problem. *Inf. Sci.*, **180**(5):679-696. [doi:10.1016/j.ins.2009.11.002]