Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2014) 22: 1338 – 1353

doi:10.3906/elk-1205-22

Solving a new bi-objective joint replenishment inventory model with modified RAND and genetic algorithms

Ommolbanin YOUSEFI*, **Seyed Jafar SADJADI**
Department of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran

Received: 12.05.2012 • **Accepted:** 25.01.2013 • **Published Online:** 15.08.2014 • **Printed:** 12.09.2014

Abstract: There are many cases in real inventory systems where more than one objective must be optimized. The main purpose of this research is to develop a multiobjective joint replenishment problem (JRP), where one objective is the minimization of the total inventory investment and another is the minimization of the total inventory ordering and holding costs. To solve the suggested model, 3 algorithms are proposed. In the first algorithm, the existing RAND method, called the best heuristic for solving the JRP, is modified and a new heuristic algorithm is developed to be applicable to the JRP with 2 objectives. The second algorithm is a multiobjective genetic algorithm that has shown good performance for solving the JRP. Finally, a third algorithm is developed, using a combination of the 2 previous ones. The performances of these algorithms are then compared. Running the programs shows good performance in solving the 9200 randomly produced problems.

Key words: Joint replenishment problem, multiobjective, modified RAND, genetic algorithm

1. Introduction

The main objective of most inventory models is the minimization of the total cost. In multiproduct inventory problems, the total cost consists of the set up or ordering cost and the holding cost. The joint replenishment problem (JRP) is a multiitem inventory model in which products are ordered from the same supplier, use the same transportation devices, or are placed in different packages from a batch production [1].

The cost of placing an order to a supplier consists of 2 main components:

1. The major ordering cost, independent of the number of different products in the order, which is carried out when each order is placed.

2. The minor ordering cost, which depends on the number of different products in the order, where a different number for each product is used.

Because of the major ordering cost, using a group replenishment or group ordering may lead to substantial cost savings [2,3]. Arkin [4] showed that the JRP is a nondeterministic polynomial-time hard (NP-hard) problem. There are 2 major strategies for solving the JRP: a direct grouping strategy (DGS) and an indirect grouping strategy (IGS). Under the DGS, products are partitioned into a predetermined number of sets; the products within each set have the same cycle time. In the IGS, replenishment is made at regular time intervals, named

*Correspondence: oyousefi@iust.ac.ir

the basic cycle time. Each product has a replenishment quantity sufficient to last for exactly an integer multiple of the basic cycle time. Products within each group have the same integer multipliers [5].

In this research, a special case of the classic JRP is developed. The proposed model consists of 2 objective functions. Most inventory models aggregate several cost concepts and other requirements such as the service level into a single objective, but in the multiobjective inventory model, the decision maker (DM) tries to optimize 2 or more objectives simultaneously with or without constraint(s) [6,7]. Starr and Miller [8] proposed an inventory model with 2 minimization objectives: cycle stock investment and the workload. Gardner and Dannenbring [9] used minimization of the workload, minimization of the inventory investment, and maximization of customer service as 3 objectives of their inventory model. Padmanabhan and Vart [10] used minimization of the wastage cost and maximization of the profit as 2 objectives. Wee et al. [7] solved an inventory model with 2 minimization objectives: the expected annual total relevant cost and expected annual frequency of sock out. Tsou [6] formulated an inventory model consisting of minimization of the expected annual total cost of the relevant expected annual frequency of stock-out occasions and expected annual number of items stocked out. Xu et al. [11,12] used maximization of the total average profit and minimization of the total average wastage cost as 2 objectives in their inventory models. Kang and Lee [1] developed an inventory model with 3 objectives: minimization of the total cost, maximization of the yield rate, and fixing of the replenishments to a desired number. The model developed in this paper optimizes 2 objective functions, which were introduced in Section 1. Sections 2 and 3 describe the main concepts of the RAND algorithm and multiobjective optimization problems, respectively. Section 4 states the developed model (bi-objective joint replenishment model). Section 5 describes 3 new algorithms for solving the proposed model, consisting of a modified RAND (M-RAND) algorithm, a multiobjective genetic algorithm (GA), and a hybrid of the M-RAND and GA (RG). Section 6 states the results of the computational experiments to compare the performances of the 3 algorithms for some randomly generated problems. A conclusion of this research follows in Section 7.

2. The RAND algorithm

In the JRP, the order quantity of each item, Q_i, and the order cycle time of each item, T_i, are 2 replenishment schedules. Under the IGS strategy, the decision variables are the basic cycle time, T, and an integer number that states the replenishment schedule of the ith item, k_i, such that:

$$T_i = k_i T \quad \text{and} \quad Q_i = D_i T_i \qquad \forall i = 1, 2, \ldots, n \tag{1}$$

Thus, the JRP model is as follows:

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

$$\text{subject to}: T \geq 0, k_i \geq 1, k_i \in N \ \ \forall i = 1, 2, \ldots, n \tag{3}$$

Here, n is the number of items, D_i is the demand of the ith item per unit time, h_i is the holding cost of the ith item per unit time, S is the major ordering cost and s_i is the minor ordering cost of the ith item, TCH is the total cost of the inventory holding, and TCS is the total cost of set up or ordering of the inventory. Silver [13] developed a heuristic algorithm for solving the JRP, which Goyal and Belton [14] and then Kaspi and Rosenblatt [15] further improved. This new heuristic algorithm was then called the RAND (based on a

randomly procedure) method. RAND is the most popular heuristic method for solving the JRP. It is based on computing m equally spaced values of the basic cycle time within its upper and lower bounds obtained from Eqs. (4) and (5) and then applying Silver's improved algorithm at each T [2].

$$T_{max}=\left[2\left(S+\sum_{i=1}^{n}s_i\right)\Big/\sum_{i=1}^{n}D_ih_i\right]^{1/2} \tag{4}$$

$$T_{\min}=\sqrt{\frac{s_i}{h_iD_i}} \tag{5}$$

- **Procedure of the RAND algorithm**

Step 1: Compute T_{max} and T_{min} from Eqs. (4) and (5), respectively.

Step 2: Divide the range [T_{min}, T_{max}] into m different equally spaced values of T (T_1, T_2, ..., T_j, ..., T_m). The value of m is to be decided by the DM.

Set j = 0.

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

Step 4: Set r = r + 1 for Tj and for each product i compute:

$$k_{i,r}^2=\frac{2\,s_iT_j^2}{D_ih_i} \tag{6}$$

Step 5: Set r = r + 1 for T_j and for each product i, find $k_i^*(r)$ for each i, where:

$$k_i^*(r)=L \quad if \;\; L(L-1)<k_i^2(r)\le L(L+1) \tag{7}$$

Step 6: Compute a new cycle time T_j according to:

$$T_j=\left[\left(S+\sum_{i=1}^{n}S_i/k_i^*(r)\right)\Big/\sum_{i=1}^{n}D_ih_ik_i^*(r)\right]^{1/2} \tag{8}$$

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

Step 8: If j = m, then select $(T_j,k_1^*(r),\ldots,k_n^*(r))$ with the minimum $TC(T,K_{i,s})$ (from Eq. (2)); otherwise, go to step 3.

3. Multiobjective optimization problems

Multiobjective optimization problems usually belong to complex and nonlinear systems in real engineering cases and contain several objectives that require optimization. The goal of these problems is finding a vector of feasible decision variables to reach a vector of acceptable values for all of the objective functions. For a multiobjective problem, with several and possibly conflicting objectives, there is usually no single or perfect optimal solution and a set of Pareto optimal solutions (POSs) must be obtained [16]. A POS is a solution dominated with no solution in the feasible region. A dominated solution is a solution that performs better

in at least one objective and equivalently or better in the other objectives. In this research, we assume that the DM may need a single solution or may want to find a set of POSs. The main goal of a multiobjective optimization algorithm is to identify the Pareto optimal set. There are 2 general approaches for solving a multiobjective optimization problem. In the first approach, the preference approach, one POS, instead of multiple POSs, is found. The classical multiobjective optimization algorithms are based on this approach. The second general approach is the ideal approach, in which a representative set of POSs must be found. Evolutionary multiobjective optimization algorithms use this approach. These algorithms can be classified into 3 groups: 1) aggregating function approaches that combine all of the objective functions into a single function, 2) population-based approaches that use an evolutionary algorithm for obtaining POSs, and 3) Pareto-based approaches that are based on multiobjective evolutionary algorithms and use the Pareto optimality concept in their selection mechanism [17].

4. Developing a new model

This section introduces the notation and formulation used in our bi-objective JRP model. All of the assumptions, input parameters, and decision variables are stated.

4.1. Assumptions

The assumptions of the proposed model are similar to those of the classic JRP, as follows:
1. Parameters are known and deterministic.
2. Shortages are not permitted.
3. Quantity discounts are not allowed.
4. The holding cost is linear.
5. The IGS strategy is used.

4.2. Input parameters

i: Index of the items; i = 1, 2, 3, .., n.
D_i: The demand rate of item i.
S: The major ordering cost.
s_i: The minor ordering cost of the item.
h_i: The inventory holding cost of item i per unit time.
c_i: The unit cost of item i.

4.3. Decision variables

T: The basic cycle time.
k_i: The integer number that decides the replenishment schedule of item i.

4.4. Objective functions

TC: The total cost of ordering and holding of inventory per unit time.
TII: The total inventory investment per unit time.

4.5. The multiobjective JRP formulation

The 2 objectives of the proposed model are:

$$1)\, minimize\, TC\,(T,K_{i,s}) = TCH + TCS$$

$$= \frac{T}{2}\sum_{i=1}^{n} k_i D_i h_i + \frac{\left(S + \sum_{i=1}^{n} \frac{s_i}{k_i}\right)}{T} \tag{9}$$

$$2)\, minimize\, TII\,(T,K_{i,s}) = T\sum_{i=1}^{n} c_i k_i D_i \tag{10}$$

$$subject\, to: T \geq 0\,,\; k_i \geq 1\,,\; k_i \in N \;\; \forall\, i = 1\,,2\,,\ldots,n \tag{11}$$

5. Developing 3 new algorithms for solving the proposed model

5.1. Modifying the RAND algorithm (M-RAND)

To solve the proposed model, according to Eqs. (9) to (11), it is assumed that w_1 and w_2 are the weights of the 2 objective functions, such that $w_2 = 1 - w_1$. Using the $\imath^p$ metric method with $p = 1$, the following objective function must be optimized:

$$TCF\,(T, K_{i,s}) = w_1\left(\frac{TC_{\max} - TC}{TC_{\max} - TC_{\min}}\right) + w_2\left(\frac{TII_{\max} - TII}{TII_{\max} - TII_{\min}}\right) \tag{12}$$

It is supposed that

$$w_1^{'} = \frac{-w_1}{TC_{\max} - TC_{\min}} \tag{13}$$

$$w_2^{'} = \frac{-w_2}{TII_{\max} - TII_{\min}} \tag{14}$$

such that $TC_{\max}TC_{\min}$, $TII_{\max}TII_{\min}$ can be obtained from the RAND algorithm.

Thus, the final model is as follows:

$$minimize\, TCF^{'}\,(T, K_{i,s}) = w_1^{'}\,(TC) + w_2^{'}\,(TII)$$

$$= \frac{w_1^{'}T}{2}\sum_{i=1}^{n} k_i D_i h_i + \frac{w_1^{'}\left(S + \sum_{i=1}^{n} \frac{s_i}{k_i}\right)}{T} + w_2^{'}T\sum_{i=1}^{n} c_i k_i D_i \tag{15}$$

$$\text{subject to:} T \geq 0\,,\; k_i \geq 1,\,,\, k_i \in N\, \forall\, i = 1,2,\ldots,n \tag{16}$$

For a fixed value of $k_i\,(\forall\, i = 1\,,2\,,\ldots,n)$,

$$\frac{\partial TCF^{'}}{\partial T} = 0,\ \text{yields:}$$

$$T(K_{i,s}) = \left[\frac{2\,w_1^{'}\left(S + \sum_{i=1}^{n} \frac{s_i}{k_i}\right)}{\sum_{i=1}^{n} D_i k_i \left(w_1^{'} h_i + 2\,w_2^{'} c_i\right)} \right]^{\frac{1}{2}} \tag{17}$$

Since $T(K_{i,s})$ is decreasing in $(k_1 k_2, \ldots, k_n) \in N^n$ if it is assumed that $k_i = 1\,(\forall\, i = 1\,,\, 2\,, \ldots, n)$, the upper bound of T is obtained as follows:

$$T_{\max} = \left[\frac{2\,w_1^{'}\left(S + \sum_{i=1}^{n} s_i\right)}{\sum_{i=1}^{n} D_i \left(w_1^{'} h_i + 2\,w_2^{'} c_i\right)} \right]^{\frac{1}{2}} \tag{18}$$

On the other hand, for a fixed value of T,

$\frac{\partial TCF^{'}}{\partial K_i} = 0\,(\forall\, i = 1\,,\, 2\,, \ldots, n)$, gives:

$$k_i^2 = \frac{2\,w_1^{'} s_i}{T^2 (w_1^{'} h_i + 2\,w_2^{'} c_i) D_i} \tag{19}$$

Thus, we can show that $(k_1(T), k_2(T), \ldots, k_n(T)) \in N^n$ must satisfy:

$$\frac{2\,w_1^{'} s_i}{k_i(T)\,(k_i\,(T) + 1\,)(w_1^{'} h_i + 2\;w_2^{'} c_i) D_i} \leq T^2 \leq \frac{2\,w_1^{'} s_i}{k_i(T)\,(k_i\,(T) - 1\,)(w_1^{'} h_i + 2\,w_2^{'} c_i) D_i}$$

$$\forall i = 1\,,\, 2\,, .., n \tag{20}$$

If $k_i = 1(\,\forall\, i = 1\,, 2\,, \ldots, n)$, Eq. (20) gives the following lower bound of T:

$$T_{\min} = [\frac{w_1^{'} s_i}{(w_1^{'} h_i + 2\,w_2^{'} c_i) D_i}]^{\frac{1}{2}} \tag{21}$$

Using $T_{\max}$ and $T_{\min}$ from Eqs. (18) and (21), respectively, we modify the RAND method and develop a new algorithm for solving the bi-objective JRP. Figure 1 shows the modified RAND algorithm that we call M-RAND from now on.

It is clear that for obtaining the POSs, we can solve the proposed model many times with different values of the objective weights using M-RAND and then obtain the POSs.

5.2. Developing a GA

In this section, a multiobjective GA for solving the proposed model is introduced. Two cases are possible, which are described in the following.

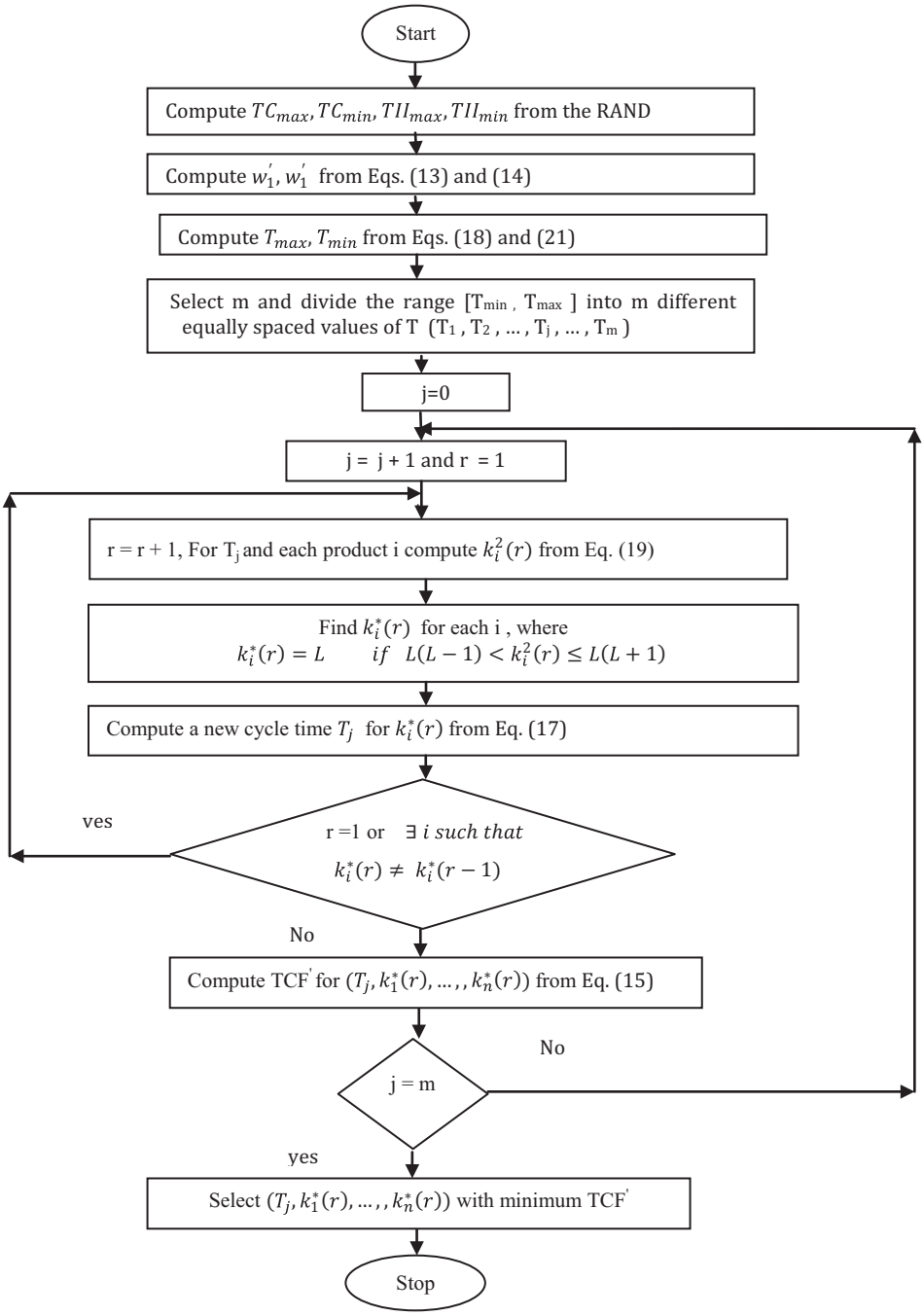

Figure 1. The M-RAND algorithm.

5.2.1. Case 1: Obtaining a single optimal solution

In this case, we suppose that the DM needs a single optimal solution instead of the set of POSs and propose his/her preferences, i.e. the weights of the objective functions. For developing a GA, we use the preference approach, and we aggregate 2 objective functions into 1 objective and develop a single objective GA. For designing a GA, some basic components are considered as follows:

- **Variable bounds**

As mentioned earlier, under the IGS strategy, decision variables are the basic cycle time (T) and integer multiplier of the basic cycle time for each product $(k_{i,s})$. The upper and lower bounds of T can be obtained from Eqs. (18) and (21). The lower bound of each $(k_{i,s})$ is obviously $k_i\,(min) = 1\ (\forall\, i = 1, 2, \ldots, n)$. For obtaining the upper bound of $(k_{i,s})$, the individual solution from the economic order quantity formula $(T_i\,(in))$ and $\mathrm{T}_{\min}$ can be used [2]:

$$T_i\,(in) = \left(\frac{2\ (S+s_i)}{h_i}\right)^{\frac{1}{2}} \tag{22}$$

$$k_i\,(max) = \left\lceil \frac{T_i\,(in)}{T_{min}} \right\rceil \tag{23}$$

Here, $\lceil . \rceil$ denotes the upper-entire function.

- **Representation (solution encoding)**

The good representation of a solution is an important aspect of the development of a GA [2]. In our proposed GA, a chromosome or a solution consists of n genes that represent n integers $(k_{i,s})$

- **Initial population**

The population is introduced using a random number generator, which produces n integer values between $k_i\,(min)$ and $k_i\,(max)$.

- **Fitness value**

The fitness value is the same as the final objective function of the model. For evaluating each chromosome in the population, at first the optimal basic cycle time (T) is determined for each chromosome $(\mathrm{k}_1, \mathrm{k}_2, \ldots, \mathrm{k}_n)$ using Eq. (17), and then the total relevant objective function is computed for a given $(\mathrm{T}, \mathrm{k}_1, \mathrm{k}_2, \ldots, \mathrm{k}_n)$ using Eq. (15).

In this equation, $TC_{\max} TC_{\min}$, $TII_{\max}$, $TII_{\min}$ are obtained from the running of the GA with separate fitness functions equal to each of these objective functions.

- **Reproduction (selection strategies)**

A roulette-wheel selection mechanism is used for selecting individuals for reproduction. In this method, the probability of the ith chromosome with fitness value $TCF_i^{'}$ for selection as a parent is $\frac{TCF_i^{'}}{\sum_i TCF_i^{'}}$.

- **Crossover and mutation**

Offspring are produced from parents that are selected from the previous generation with random-point crossover. This operator acts with probability p_c. It selects a random number of chromosomes from 2 parents with the same locations and exchanges them.

After the production of offspring, the mutation operator exchanges each chromosome with relatively low probability p_m. The value of each chromosome changes to a random value between $k_i\,(min)$ and $k_i\,(max)$

6. Replacement (survive selection)

After generating new offspring, the roulette-wheel mechanism is used for selecting a fixed number of parents and offspring for the next generation. This fixed number is equal to a predefined population size. In this mechanism, solutions with higher fitness functions have a higher chance for selection for the next generation.

- **Termination condition**

The stopping criterion of the proposed algorithm is when no improvements are obtained in 50 generations.

6.1. Case 2: Obtaining a subset of POSs

In this case, we decide to obtain a subset of POSs. For this purpose, we develop the strength Pareto evolutionary algorithm (SPEA-II). This algorithm is one of the most powerful types of multiobjective GAs, which use an external archive to save nondominated solutions obtained so far in the search [18,19]. For developing the SPEA-II algorithm, we use variable bounds, an initial population, and crossover, mutation, and termination conditions that are the same as in the first case. The representation and the procedure of this algorithm are as follows:

- **Representation**

In our proposed SPEA-II, an individual consists of n + 1 genes, among which the first n genes show the values of $K_{i,s}$ and the last gene represents the basic cycle time (T).

- **Procedure of SPEA-II [18,19]**

N_E: The maximum size of the nondominated archive, E.

N_P: The population size.

K: Parameter for density calculation $k = \sqrt{N_E + N_P}$.

Step 1: Initialization: randomly generate an initial solution P_0 and set $E_0 = \infty$.

Step 2: Fitness assignment: calculate fitness values of individuals in $P_t \cup E_t$ as follows:

Step 2-1: $r(x,t) = \sum_{\substack{y \in P_t \cup E_t \\ y \succ x}} s(y,t)$, where s(y,t) is the number of solutions in $P_t \cup E_t$ dominated by solution y.

Step 2-2: Calculate the density as $m(x,t) = (\sigma_x^k + 1)^{-1}$, where σ_x^k is the distance between solution x and the kth nearest neighbor.

Step 2-3: Assign a fitness value as f(x,t) = r(x,t) + m(x,t).

Step 3: Environmental selection: copy all nondominated solutions in $P_t \cup E_t$ to E_{t+1}. Two cases are possible:

Case 1: If $|E_{t+1}| > N_E$, then truncate $|E_{t+1}| - N_E$ solutions by iteratively removing solutions with the maximum σ_x^k distances. Break any tie by examining σ^l for l = k- 1,..., 1 sequentially.

Case 2: If $|E_{t+1}| \leq N_E$, copy the best $N_E - |E_{t+1}|$ dominated solutions according to their fitness values from $P_t \cup E_t$ to E_{t+1}.

Step 4: Termination: if the stopping criterion is satisfied, stop and return nondominated solutions in E_{t+1} .

Step 5: Mating selection: perform binary tournament selection with replacement on E_{t+1} in order to fill the mating pool.

Step 6: Variation: apply crossover and mutation operators to the mating pool to create N_p offspring solutions, copy offspring to $P_t + 1$, t = t + 1, and go to step 2.

6.2. Developing an algorithm from the hybrid RAND and GA (RG method)

The results found for various combinatorial optimization problems have shown that the combinations of different algorithms are very powerful. Thus, the hybrid of the M-RAND method as a good heuristic approach and the GA or SPEA-II as a good metaheuristic approach for solving a multiobjective JRP is used. For this hybridization, the main algorithm for case 1 is the GA and for case 2 is the SPEA-II, where the main components are the same as described in Section 5.2, except for the mechanism of generating the initial population.

For generating an initial population with a predefined population size in the GA method, n integer values between k_i (min) and k_i (max) should be produced. However, in this hybrid method, at first, the M-RAND method runs and introduces the best solutions, and then the roulette-wheel mechanism is used for introducing n random numbers for each individual.

The numbers that exist in the RAND solutions have more chance of selection at this stage. Thus, the initial solutions with high probability are similar to the optimal RAND solutions.

7. Results and discussion

To test the performances of the proposed algorithms, some problems are randomly generated. Each instance consists of 6 parameters, summarized in Table 1. Moreover, the parameters of each algorithm are shown in Table 2 [2]. We coded all of the algorithms in Microsoft Visual Basic 6.5. The performances of the proposed algorithms are analyzed in 3 parts.

Table 1. Instance parameter values.

Parameter	Range
Demand D_i	[100, 100,000]
Minor ordering cost s_i	[0.5, 5]
Holding cost	[0.2, 3]
Number of products	10, 20, 30, 50
Major ordering cost S	5, 10, 15, 20
Cost of item c_i	1

Table 2. Parameter values of the algorithms.

Parameter	Value	Algorithm
Number of spaced values of T(m)	10	M-RAND and RG
Population size	100	GA and SPEA-II and RG
Probability of crossover	0.6	GA and SPEA-II and RG
Probability of mutation	0.02	GA and SPEA-II and RG
Nondominated archive size	12	SPEA-II and RG
w_1 (weight of TC)	0.3, 0.7, 0.5, 1	M-RAND, GA, and RG

7.1. First comparison part

In the first part of the comparisons, we suppose that the DM needs a single optimal solution instead of the set of POSs and proposes the weights of the objective functions. Thus, in this section, a GA according to case 1 of Section 5.2 is developed. Next, the performances of the algorithms for obtaining a single optimal solution are observed and compared. In this section, we compare the results of the 3 algorithms with each other and with the result of the RAND and GA methods in the literature [2].

For each value of w_1 and w_2, 1600 randomly produced problems are solved by the 3 algorithms. Table 3 shows the results for $w_1 = 0.7$, for instance. In Table 3, for each value of S and n, 100 problems are solved. Columns Max-i and Avg-i are the maximum and average of improvement, i.e. $100 \times \frac{(\text{Better solution} - \text{The worst solution})}{\text{Better solution}}$), respectively. As shown in Table 3, for example, for n = 20 and S = 5, for 4 problems of 100 randomly produced problems, the GA provides a better solution than the M-RAND and RG, with an average improvement of 0.13%, and for 74 problems, the RG provides a better solution than the M-RAND and GA, with an average improvement of 0.16%. Moreover, for 22 problems, the RG and GA provide a solution that is the same, but better than the M-RAND solution, with an average improvement of 0.23%. From Table 3 we can also see that, generally, the RG in 71.75%, GA in 26.75%, and GA in 1.5% of problems provides a better solution than the other(s). In conclusion, it can be seen that the RG is superior to the other algorithms, especially for larger values of S and n. For smaller values of S and n, the RG and GA show relatively similar performance, but both outperform the M-RAND. A summary of the computational results for 4 values of w_1 and w_2 are shown in Table 4, where it is seen that the RG is superior to the other 2 algorithms, and for 80% (57.71% + 15.1% + 7.4%) of the problems, it provides the best solution. It can also be confirmed that the RG has better performance for small values of w_1. For $w_1 = 1$, the classic JRP is solved and, as shown in 64% of the problems, all 3 algorithms have similar performances. In 22% of the problems, the M-RAND and RG provide the same solution and outperform the GA. Khouja [2] showed that RAND has an impressive performance and identifies the optimal solution for approximately 83% of problems. Thus, it can be said that the proposed algorithms deliver the optimal solution for at least (64% + 22%) × 83% = approximately 71% of problems.

7.2. Second comparison part

In this section, we compare the performances of the algorithms with each other for obtaining a set of POSs. For this aim, according to case 2 of Section 5.2, the SPEA-II algorithm was developed. For each value of 16 groups of S and n, 100 problems are randomly generated. For instance, 4 randomly generated problems with n = 10, S = 5; n = 10, S = 20; n = 50, S = 5; and n = 50, S = 20 are solved to show the performances and outputs of the algorithms. The results, i.e. POSs, are shown in Figures 2–5, where it is seen that the M-RAND gives a higher number of POSs than the RG and GA. However, the diversity of solutions in the RG and SPEA-II is larger than in the M-RAND. Obviously, we can aggregate the POSs of the 3 algorithms and ask the DM to choose the most utilized solutions from these aggregated solutions. The results of solving the 1600 randomly generated problems by the proposed algorithms are summarized in Table 5. For each value of S and n, i.e. for each row of Table 5, 100 randomized problems are produced and solved by each algorithm. In each row, by solving each problem, some POSs are obtained. First, for each problem of all 100 problems and with each algorithm, these values are calculated: Avg (TCI), Avg (TII), and Avg (time). Next, for each problem, the minimums of the above values between the 3 algorithms are calculated, and then the columns of Table 5 are calculated.

Table 3. Results of the 3 algorithms for $w_1 = 0.7$.

n	S	GA[1]			RG[2]			GA = RG > M-RAND[3]		
		%	Max-i	Avg-i	%	Max-i	Avg-i	%	Max-i	Avg-i
10	5	1	0.14	0.14	41	0.22	0.14	58	0.58	0.23
10	10	3	0.14	0.11	34	0.13	0.09	63	0.35	0.18
10	15	6	0.13	0.09	23	0.13	0.08	71	0.23	0.14
10	20	2	0.13	0.11	27	0.12	0.08	71	0.22	0.12
20	5	4	0.15	0.13	74	0.33	0.16	22	0.33	0.23
20	10	1	0.19	0.19	79	0.20	0.13	20	0.29	0.17
20	15	1	0.07	0.07	83	0.17	0.11	16	0.21	0.14
20	20	0	0.00	0.00	82	0.17	0.10	18	0.16	0.13
30	5	0	0.00	0.00	84	0.24	0.15	16	0.42	0.26
30	10	0	0.00	0.00	96	0.22	0.14	4	0.21	0.17
30	15	0	0.00	0.00	99	0.21	0.13	1	0.16	0.16
30	20	0	0.00	0.00	95	0.22	0.12	5	0.15	0.11
50	5	5	0.17	0.17	53	0.23	0.17	42	0.37	0.28
50	10	0	0.00	0.00	82	0.21	0.14	18	0.24	0.20
50	15	1	0.09	0.09	96	0.19	0.13	3	0.21	0.18
50	20	0	0.00	0.00	100	0.18	0.12	0	0.00	0.00
Max		6	0.19	0.19	100	0.33	0.17	71	0.58	0.28
Avg		1.5	0.07	0.07	71.75	0.20	0.12	26.75	0.26	0.17

[1]The GA is better than the M-RAND and RG.
[2]The RG is better than the M-RAND and GA.
[3]The GA is equal to RG and both are better than M-RAND.

Table 4. Comparison of the 3 algorithms.

		$w_1 = 1$		$w_1 = 0.7$		$w_1 = 0.5$		$w_1 = 0.3$		Total	
		Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg
M-RAND = GA = RG[1]	%	96	64	0	0	17	3	17	3	96	17
GA[2]	%	32	8.1	6.0	1.5	0	0	0	0	32.0	2.4
	Max-i	0.1	0	0.2	0.1	0	0	0	0	0.2	0
	Avg-i	0	0	0.2	0.1	0	0	0	0	0.2	0
RG[3]	%	8.0	1.9	100.0	71.8	99.0	78.6	99.0	78.6	100.0	57.7
	Max-i	0.1	0	0.3	0.2	0.2	0.1	0.2	0.1	0.3	0.1
	Avg-i	0.1	0	0.2	0.1	0.1	0.1	0.1	0.1	0.2	0.1
GA = RG > M-RAND[4]	%	23.0	4.8	71.0	26.8	58.0	14.3	58.0	14.3	71.0	15.1
	Max-i	0.3	0.1	0.6	0.3	0.2	0.1	0.2	0.1	0.6	0.1
	Avg-i	0.1	0	0.3	0.2	0.1	0	0.1	0.0	0.3	0.1
M-RAND = RG > GA[5]	%	45.0	21.2	0	0	9.0	4.2	9.0	4.2	45.0	7.4
	Max-i	0.2	0.1	0	0	0.2	0.1	0.2	0.1	0.2	0.1
	Avg-i	0.1	0.0	0	0	0.1	0.1	0.1	0.1	0.1	0

[1]The 3 algorithms give the same solution.
[2]The GA is better than the M-RAND and RG.
[3]The RG is better than the M-RAND and GA.
[4]The GA and RG are better than the M-RAND.
[5]The M-RAND and RG are better than the GA.

As can be seen from columns 3–5 of Table 5, for TCI, the M-RAND provides minimum values and the SPEA-II and RG show relatively the same performance. Columns 6–8 show that for TII, the SPEA-II and M-RAND have relatively the same performance, and the RG outperforms them. From columns 9–11, it is

observed that the M-RAND has minimum time but the RG and SPEA-II have relatively the same performance, and for larger problems, the RG is faster than the SPEA-II. Columns 12–14 show that the number of POSs in the RG and SPEA-II is equal but the number of POSs in the M-RAND is larger than those in the GA and SPEA-II.

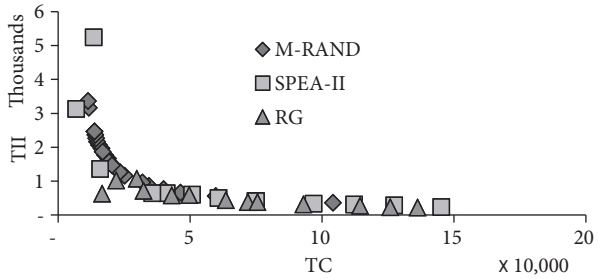

Figure 2. POSs of a randomized problem with n = 10, S = 5.

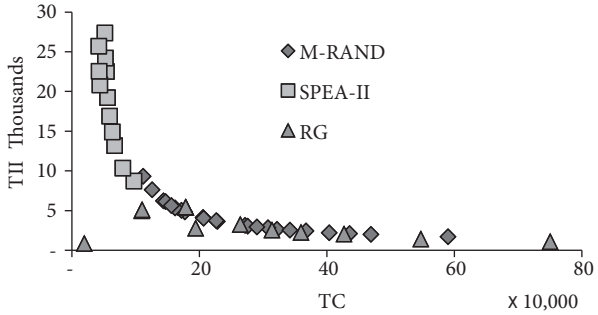

Figure 3. POSs of a randomized problem with n = 10, S = 20.

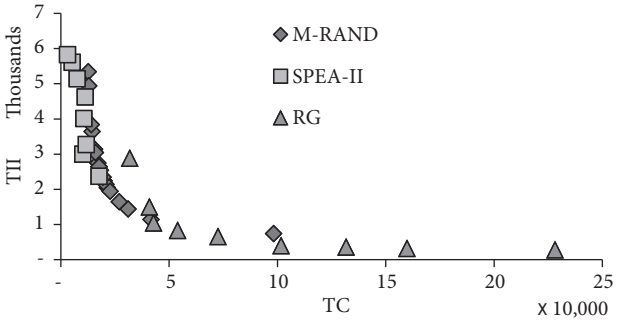

Figure 4. POSs of a randomized problem with n = 50, S = 5.

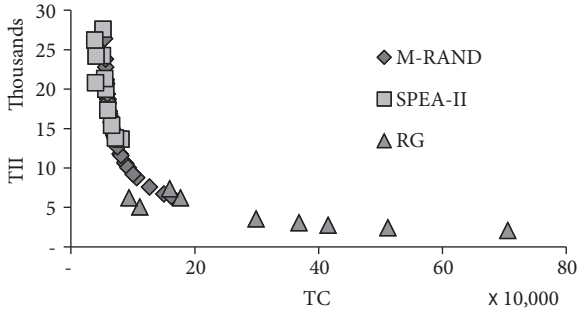

Figure 5. POSs of a randomized problem with n = 50, S = 20.

7.3. Third comparison part

In this part, we make a comparison of our results with the existing results in the literature, because all of the JRP models have only one objective function. We suppose that the value of the second objective function, i.e. TII, is equal to 0. Next, the obtained model is solved by the 3 algorithms and the results are then compared with the RAND method [15]. Table 6 shows the results, where it is seen in column 3, that previous tests (except for n = 50, for which no results exist in the literature) show that the RAND method delivers the optimal solution for approximately 83% of problems. As columns 4 and 6 show, the M-RAND reaches a solution with the same or better objective function (TCI) than the GA or RG for 51.2% of problems and the average percentage saving is 8.5%. Columns 6 and 7 show that the SPEA-II is not outperformed by the M-RAND and RG for 28.9% of problems, and that the average percentage saving in TCI provided by the GA is 10.1%. As shown in columns 8 and 9, the RG performs better than or equal to the other algorithms for 19.9% of problems. In these problems, the RG provides 4.6% of the average percentage saving in TCI. In conclusion, we can say that our proposed algorithms perform well relative to the RAND method.

Table 5. Comparison of the performances of the 3 algorithms.

n	S	Avg (TCI) / Min (Avg (TCI))			Avg (TCD) / Min (Avg (TCD))			Avg (mean time) / Min (Avg (mean time))			Average number of POSs		
		SPEA-II	M-RAND	RG	SPEA-II	M-RAND	RG	SPEA-II	M-RAND	RG	SPEA-II	M-RAND	RG
10	5	1.28	1	1.11	1.18	1.78	1	5.5	1	6.6	10.65	30.85	10.65
	10	1.26	1	1.83	1.00	1.61	1.09	5.1	1	5.0	10.28	32.8	11.28
	15	1.49	1	2.39	1.50	1.67	1	4.7	1	5.5	10.18	32	10.18
	20	1.61	1	1.55	1.20	1.73	1	4.6	1	5.6	11.64	33.3	11.69
20	5	1.43	1	1.20	1.00	1.38	1	6.6	1	5.2	10.68	36.25	10.68
	10	1.13	1	1.30	1.00	1.50	1.04	3.9	1	5.9	10.61	37.15	10.71
	15	1.41	1	1.74	1.11	1.39	1	3.8	1	4.8	10.75	39.05	10.60
	20	3.49	1	1.80	1.22	2.11	1	3.5	1	4.4	11.10	38.75	10.10
30	5	1.37	1	1.56	1.70	1.26	1	3.7	1	4.6	10.47	40.2	10.52
	10	1.93	1	1.61	2.17	1.08	1	3.2	1	5.7	10.29	38.8	10.24
	15	1.17	1	1.11	1.71	1.18	1	2.9	1	4.0	10.19	41.8	10.24
	20	1.26	1	1.44	1.40	1.42	1	2.9	1	3.8	10.04	41.15	9.99
50	5	1.02	1	1.16	2.30	1.25	1	5.9	1	3.9	10.67	43.75	10.82
	10	1.23	1	1.29	1.01	1.16	1	6.7	1	3.8	10.64	44.15	10.54
	15	1.42	1	1.64	1.09	1.18	1	5.9	1	3.2	9.97	44.3	10.02
	20	1.18	1	1.15	1.66	1.19	1	6.6	1	3.6	10.35	43.45	10.30
Avg		1.48	1.00	1.49	1.39	1.43	1.01	4.72	1.00	4.71	10.53	38.61	10.53

Table 6. Comparison of the performances of the 3 algorithms with the RAND.

n	S	% RAND optimal m = 10	The best algorithm					
			M-RAND		SPEA-II		RG	
			% number	Aver. % improv.	% number	Aver. % improv.	% number	Aver. % improv.
10	5	95.2	50.6	11.5	39.7	18.8	9.6	2.7
	10	98.1	39.0	20.0	31.9	4.3	29.1	2.1
	15	99.5	68.7	12.5	26.3	9.6	4.9	0.6
	20	99.8	52.3	8.6	28.5	9.6	19.2	8.1
20	5	76.2	31.8	6.5	40.9	8.3	27.3	1.3
	10	84.8	55.5	8.5	19.7	6.7	24.8	6.3
	15	88.2	46.2	4.5	25.0	4.5	28.8	10.0
	20	91.1	62.3	8.2	22.4	10.8	15.3	3.8
30	5	53.5	49.1	5.6	38.4	6.0	12.6	3.1
	10	65.5	40.8	6.0	25.7	5.7	33.4	1.2
	15	73.8	56.9	6.6	22.0	7.7	21.2	5.8
	20	75.3	61.2	3.3	26.6	29.5	12.2	9.7
Maximum		99.8	68.7	20.0	40.9	29.5	33.4	10.0
Average		83.4	51.2	8.5	28.9	10.1	19.9	4.6

8. Conclusion

In this paper, a multiobjective JRP was developed. The shortcomings of previous models in the literature, problems of multiobjective inventory models, were reviewed and a new improved model was presented. As our proposed model is NP-hard, 3 new algorithms, the M-RAND, multiobjective GA, and RG, were developed to solve it. To validate these algorithms, some test problems were designed and solved. The comparison of the results showed that the proposed algorithms are able to find near optimal solutions for the problems. The proposed model has no constraints, such as space or investment, which can be added to increase its applicability. Moreover, it seems that using direct grouping replaces the indirect grouping that is used in this research, which could be another area of work for future researches. Moreover, multiobjective models of the stochastic joint replenishment problem or dynamic-demand joint replenishment problem can be developed and solved for future researches. Finally, the use of other solving methods, such as population-based methods with more convergence speed or less computational time, can provide a good opportunity for future study. As an example, the PSO algorithm modified by Özkaya and Güneş [20] can be used for solving the multiobjective JRP.

References

[1] H. Kang, A. Lee, "Inventory replenishment model using fuzzy multiple objective programming: a study of a high-tech company in Taiwan", Applied Soft Computing, Vol. 10, pp. 1108–1118, 2010.

[2] M. Khouja, M. Michalewicz, S. Satoskar, "A comparison between genetic algorithms and the RAND method for solving the joint replenishment problem", Production Planning and Control, Vol. 11, pp. 556–564, 2000.

[3] K. Moon, B.C. Cha, "The joint replenishment problem with resource restriction", European Journal of Operational Research, Vol. 173, pp. 190–198, 2006.

[4] E. Arkin, D. Joneja, R. Roundy, "Computational complexity of un capacitated multi echelon production planning problems", Operations Research Letters, Vol. 8, pp. 61–66, 1989.

[5] M. Khouja, S. Goyal, "A review of the joint replenishment problem literature: 1989-2005", European Journal of Operational Research, Vol. 186, pp. 1–16, 2008.

[6] C.S. Tsou, "Evolutionary Pareto optimizers for continuous review stochastic inventory system", European Journal of Operational Research, Vol. 195, pp. 364–371, 2009.

[7] H.M. Wee, C.C. Lo, P.H. Hsu, "A multi-objective joint replenishment inventory model of deteriorated items in a fuzzy environment", European Journal of Operational Research, Vol. 197, pp. 620–631, 2009.

[8] M.K Starr, D.W. Miller, Inventory Control: Theory and Practice, Englewood Cliffs, NJ, USA, Prentice Hall, 1962.

[9] E.S. Gardner Jr., D.G. Dannenbring, "Using optimal policy surfaces to analyze aggregate inventory tradeoffs", Management Science, Vol. 25, pp. 709–720, 1979.

[10] G. Padmanabhan, P. Vart, "Analysis of multi-item inventory systems under resource constrains: a non-linear goal programming approach", Engineering Cost and Production Economics, Vol. 20, pp. 121–127, 1990.

[11] J. Xu, L. Zhao, "A multi-objective decision-making model under fuzzy rough coefficient and its application to inventory problem", Information Science, Vol. 180, pp. 679–696, 2010.

[12] J. Xu, Y. Liu, "Multi-objective decision making model under fuzzy random environment and its application to inventory problems", Information Science, Vol. 178, pp. 2899–2914, 2008.

[13] E. Silver, "A simple method of determining order quantities in joint replenishments under deterministic demand", Management Science, Vol. 22, pp. 1351–1361, 1976.

[14] S.K. Goyal, A.S. Belton, "On a simple method of determining order quantities in joint replenishments under deterministic demand", Management Science, Vol. 25, pp. 604–610, 1979.

[15] M. Kaspi, M.J. Rosenblatt, "An improvement of Silver's algorithm for the joint replenishment problem", IIE Transactions, Vol. 15, pp. 264–269, 1983.

[16] F. Djeffal, T Bendib, "Multi-objective genetic algorithms based approach to optimize the electrical performances of the gate stack double gate (GSDG) MOSFET", Microelectronics Journal, Vol. 42, pp. 661–666, 2011.

[17] D.P. Singh, A. Khare, "Different aspects of evolutionary algorithms, multi-objective optimization algorithms and application domain", International Journal of Advanced Networking and Applications, Vol. 2, pp. 770–775, 2011.

[18] A. Konak, D.W. Coit, A.E. Smith, "Multi-objective optimization using genetic algorithms: a tutorial", Reliability Engineering & System Safety, Vol. 91, pp. 992–1007, 2006.

[19] E. Zitzler, M. Laumanns, L. Thiele, "SPEA2: improving the strength Pareto evolutionary algorithm", in Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems (K.C. Giannakoglou, D.T. Tsahalis, J. Periaux, T. Fogarty, editors), Zurich, Switzerland, Swiss Federal Institute of Technology, pp. 95–100, 2001.

[20] U. Özkaya, F. Güneş, "A modified particle swarm optimization algorithm and its application to the multi objective FET modeling problem", Turkish Journal of Electrical Engineering & Computer Sciences, Vol. 20, pp. 263–271, 2012.