Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 173 (2006) 190–198
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Production, Manufacturing and Logistics

The joint replenishment problem with resource restriction

I.K. Moon [a,*], B.C. Cha [b]

[a] Department of Industrial Engineering, Pusan National University, Pusan 609-735, Republic of Korea
[b] Postal Technology Research Center, ETRI, Daejeon 305-700, Republic of Korea

Received 17 December 2003; accepted 16 November 2004
Available online 29 January 2005

Abstract

There are many resource restrictions in real production/inventory systems (for example, budget, storage, transportation capacity, etc.). But unlike other research areas, there is very little research to handle the joint replenishment problem (JRP) with resource restriction. The purpose of this paper is to develop two efficient algorithms for solving these problems. Firstly, we modify the existing RAND algorithm to be applicable to the JRP with resource restriction. Secondly, we develop a genetic algorithm for the JRP with resource restriction. Extensive computational experiments are performed to test the performances of the algorithms.

Keywords: Joint replenishment problem; Resource restriction; Genetic algorithm

1. Introduction

For an inventory system with multiple items, cost savings can be obtained when the replenishment of several items are coordinated. The joint replenishment problem (JRP) is the multi-item inventory problem of coordinating the replenishment of a group of items that may be jointly ordered from a single supplier. In this situation, the ordering cost has two components—a major common ordering cost S incurred whenever an order is placed and a minor ordering cost s_i incurred if item i is incurred in the order. In the deterministic joint replenishment problem, it is assumed that the major ordering cost is charged at a basic cycle time T and that the ordering cycle of each item is some integer multiple of this basic cycle.

Over the last few decades, the joint replenishment problem has received much attention. Arkin et al. [1] proved that the JRP is an NP-hard problem, i.e., the JRP is not solvable by polynomial-time algorithms. Goyal [4] proposed an enumeration approach, and he claimed that it always secures a global optimal

* Corresponding author. Tel.: +82 51 5102451; fax: +82 51 5127603.
E-mail address: ikmoon@pusan.ac.kr (I.K. Moon).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2004.11.020

solution. However, Van Eijs [12] has pointed out that the lower bound on an optimal cycle time used by Goyal [4] does not guarantee a global optimal solution and derived another algorithm that improves Goyal's algorithm [4]. Unlike these enumeration approaches, Silver [10,11] discussed the advantage and disadvantage of coordinating replenishments and presented a very simple non-iterative procedure to solve it. Kaspi and Rosenblatt [8] proposed an approach based on trying several values of the basic cycle time between a minimum and a maximum value. Then they ran the heuristic of Kaspi and Rosenblatt [7] for each value of the basic cycle time, which is a modified version of the algorithm of Silver [10]. They showed that their procedure (RAND) outperforms all available heuristics. Later, Goyal and Deshmukh [6] proposed an improvement of the lower bound used by Kaspi and Rosenblatt [8].

There are many resource restrictions in real production/inventory systems (for example, budget, storage, transportation capacity, etc.). But unlike other research areas, there is very little research to handle this problem. Goyal [5] introduced the JRP with one resource constraint and developed a heuristic algorithm using the Lagrangian multiplier. Recently Khouja et al. [9] applied the GA approach to the basic JRP and compared the performance of their GA algorithm with Kaspi and Rosenblatt's heuristic algorithm [8]. They mentioned that a major advantage of the GA algorithm is its ability to handle constrained problems and introduced the penalty technique for solving the constrained JRPs. However, the penalty technique is not suitable for the constrained JRPs, as we will explain later.

The purpose of this paper is to develop two algorithms for solving the constrained JRP. The following section introduces the constrained JRP and shows that how the modified RAND algorithm can be used to handle the JRP with a constraint. Section 3 develops the GA and shows that the penalty technique is not suitable for the constrained JRPs. Section 4 illustrates the procedures of two proposed algorithms with a numerical example. The next section gives the details of the computational experiments to compare the performance of three algorithms for 1600 randomly generated problems. Finally, we summarize the conclusions of the present work.

2. The constrained joint replenishment problem

To discuss the constrained joint replenishment problem, we introduce the following notation:

i	index of item, $i = 1, 2, \ldots, n$
D_i	demand rate of item i
S	major ordering cost
s_i	minor ordering cost of item i
h_i	inventory holding cost of item i, per unit per unit time
b_i	unit cost of item i
B	limit on capital that can be invested
T	basic cycle time (decision variable)
k_i	integer number that decides the replenishment schedule of item i (decision variable)

The total relevant costs per unit time to be minimized is

$$TC(T, k_i's) = \frac{S + \sum_{i=1}^{n} \frac{s_i}{k_i}}{T} + \sum_{i=1}^{n} \frac{D_i k_i T h_i}{2}, \tag{1}$$

subject to

$$\sum_{i=1}^{n} D_i k_i T b_i \leqslant B. \tag{2}$$

To solve this problem, Goyal [5] used

$$T^* = \min\left[\sqrt{2\left(S+\sum_{i=1}^{n} s_i\right)\Big/\sum_{i=1}^{n} D_i h_i},\ B\Big/\sum_{i=1}^{n} D_i b_i\right],$$

as the first value of T in his algorithm. For the unconstrained JRP, the first value of T greatly influences finding the optimal solution [8]. Using this idea, we modify the RAND algorithm and develop a new algorithm for solving the constrained JRP. The constrained RAND algorithm, we call the C-RAND from now on, is as follows. Note that we need to employ a Lagrangian multiplier in the proposed algorithm, while the RAND does not require it.

The Constrained RAND algorithm (C-RAND)

Step 1: Compute $T_{\max} = \sqrt{2(S+\sum_{i=1}^{n} s_i)/\sum_{i=1}^{n} D_i h_i}$ and $T_{\min} = \min\sqrt{2s_i/D_i h_i}$ for i.

Step 2: Divide the range $[T_{\min}, T_{\max}]$ into m different equally spaced values of $T(T_1, \ldots, T_j, \ldots, T_m)$. The value of m is to be decided by the decision maker. Set $j = 0$.

Step 3: Set $j = j + 1$, $r = 0$ and $z = 0$.

Step 4: Set $r = r + 1$. If $z = 0$, then $\lambda(r) = 0$.

Otherwise, $\lambda(r) = \left(\dfrac{S+\sum_{i=1}^{n} s_i/k_i^*(r-1)}{T_j^2} - \dfrac{\sum_{i=1}^{n} D_i h_i k_i^*(r-1)}{2}\right)\Big/\sum_{i=1}^{n} D_i b_i k_i^*(r-1).$

For T_j and each product i, compute $k_i^2(r) = 2s_i/(D_i h_i + 2\lambda(r) D_i b_i) T_j^2$.

Step 5: Find $k_i^*(r)$ for each i, where $k_i^*(r) = L$ if $L(L-1) < k_i^2(r) \leqslant L(L+1)$.

Step 6: Compute a new cycle time T_j according to

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

If $T_j = B/\sum_{i=1}^{n} D_i b_i k_i^*(r)$, then $z = 1$. Otherwise, $z = 0$.

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 for $(T_j, k_1^*(r), \ldots, k_n^*(r))$.
If $j = m$ then select $(T_j, k_1^*(r), \ldots, k_n^*(r))$ with the minimum TC.
Otherwise, go to Step 3.

3. Genetic algorithm

In this section we present a new genetic algorithm approach for the JRP with constraints. The main ideas of a genetic algorithm are introduced shortly and we will show how we apply the genetic algorithm to our problem. Introduced by Khouja et al. [9], GA is suitable for solving the JRP which has the important feature that it can be formulated as a problem having one continuous decision variable (basic cycle T) and a set of integer decision variables (n integer multiples k_i of a basic cycle T).

Genetic algorithms, which have been widely used in various areas for three decades, are stochastic search algorithms based on the mechanism of natural selection and natural genetics. Genetic algorithms, differing from conventional search techniques, start with an initial set of (random) solutions called a population. Each individual in the population is called a chromosome, representing a solution to the problem at hand.

The chromosomes evolve through successive iterations, called generations. During each generation, the chromosomes are evaluated, using some measures of fitness. Generally speaking, the genetic algorithm is applied to spaces, which are too large to be exhaustively searched. It is generally accepted that any genetic algorithm to solve a problem must have basic components, but have different characteristics depending on the problem under study (Aytug et al. [2]).

We explain our overall strategies including chromosome style as follows.

- Representation and initialization.
- Objective and fitness function.
- Reproduction, crossover and mutation.

3.1. Representation and initialization

The proper representation of a solution plays a key role in the development of a genetic algorithm. The basic cycle T and n integers (k_i's) have to be decided for solving the JRP. Khouja et al. [9] introduced a binary chromosome to represent n integers. They defined the smallest integer u_i that the values of 2^{u_i} is larger or equal to the upper bound k_i^{UB}. The total length of their binary chromosome was $\sum_{i=1}^{n} u_i$ bits. In our study, we use a random number representation for the following reasons:

(i) Our chromosome needs only n genes to represent n integers (k_i's).
(ii) It is very easy to decode our chromosome to a feasible solution.
(iii) Our GA is always searching for a suitable feasible region, not influenced by any crossover and mutation.

3.2. Objective and fitness function

Each chromosome in the population is evaluated by the following steps:

(i) Each chromosome is decoded to a feasible solution $(k_1, k_2, \ldots, k_n)$.
(ii) The optimal basic cycle T is determined for a given $(k_1, k_2, \ldots, k_n)$.
(iii) The total relevant cost TC is computed for a given $(T, k_1, k_2, \ldots, k_n)$.

Our decoding process for each gene of our chromosome is as follows:

$$k_i = k_i^{\mathrm{LB}} + \left\lfloor \left(k_i^{\mathrm{UB}} - k_i^{\mathrm{LB}} + 1\right) \times \mathrm{Gene}(i) \right\rfloor,$$

$\lfloor G \rfloor$ is the function which finds the integer number less than G. By defining the suitable range of k_i, we can reduce the search space. Khouja et al. [9] used $(k_i^{\mathrm{LB}} = 1,\ k_i^{\mathrm{UB}} = \lceil T_i^{\mathrm{IN}}/T_{\min} \rceil)$, where $T_i^{\mathrm{IN}} = \sqrt{2(S+s_i)/D_i h_i}$ is the individual optimal cycle time for product i which is obtained from an EOQ model. But we define tighter lower and upper bounds of k_i from the well-known optimality condition (Goyal [3]):

$$k_i(k_i - 1) \leqslant \frac{2s_i}{D_i h_i T^2} \leqslant k_i(k_i + 1).$$

That is to say, we can find the lower and upper bounds of k_i from the following two equations:

$$k_i^{\mathrm{LB}}\left(k_i^{\mathrm{LB}} - 1\right) \leqslant \frac{2s_i}{D_i h_i T_{\max}^2} \leqslant k_i^{\mathrm{LB}}\left(k_i^{\mathrm{LB}} + 1\right) \tag{3}$$

and

$$k_i^{\mathrm{UB}}\left(k_i^{\mathrm{UB}}-1\right) \leqslant \frac{2s_i}{D_i h_i T_{\min}^2} \leqslant k_i^{\mathrm{UB}}\left(k_i^{\mathrm{UB}}+1\right). \tag{4}$$

$T_{\max}$ and $T_{\min}$ were already defined in (step 1) of the C-RAND.

We have to decide the basic cycle time T to evaluate each chromosome. For a given particular set of k_i's, Eqs. (1) and (2) can be easily written as

$$TC(T) = \frac{C_1}{T} + C_2 T,$$

subject to

$$C_3 T \leqslant B,$$

where $C_1 = S + \sum_{i=1}^{n} \frac{s_i}{k_i}$, $C_2 = \sum_{i=1}^{n} \frac{D_i k_i h_i}{2}$, $C_3 = D_i k_i b_i$ are constants.

We take the first order derivative with respect to T, and set it to zero. We obtain

$$\frac{\partial TC(T)}{\partial T} = 0 \Rightarrow T^0 = \sqrt{\frac{C_1}{C_2}}.$$

And for a constraint, we obtain

$$T^1 = \frac{B}{C_3}.$$

Proposition 1. *For a given set of k_i's, the optimal basic cycle time T is $T^* = \min(T^0, T^1)$.*

Proof. If $T^0 \leqslant T^1$, $TC(T)$ is a convex function in the interval $[0, T^1]$. In this case, the optimal basic cycle time $T^* = T^0$. Otherwise $T^0 \geqslant T^1$, as $TC(T)$ is a decreasing function in the interval $[0, T^1]$. It is clear that the optimal basic cycle time $T^* = T^1$. This proposition proves that the penalty technique of Khouja et al.'s GA [9] cannot find the optimal solution in the constrained JRPs because they only used T^0 in their penalty function. □

3.3. Reproduction, crossover and mutation

Various evolutionary methods can be applied to this problem. We use $(\mu + \lambda)$ selection for selecting individuals for reproduction. With this strategy, μ parents and λ offspring compete for survival and the μ best out of offsprings and old parents are selected as parents of the next generation. And we produce offsprings with one-point crossover. Whenever an offspring is produced, mutation is applied with probability P_m. The operation of mutation replaces one gene of the chromosome chosen at random with new random number between $(0, 1)$.

4. Numerical example

To illustrate the procedures of our C-RAND and GA, we use the numerical example of Goyal [5]. The data for this example are given in Table 1. We also assume $S = \$200$ and $B = 25{,}000$.

To illustrate the C-RAND, we use $m = 5$. In step 1, we obtain $T_{\max} = 0.2188$ and $T_{\min} = 0.0949$. The iterative steps of the C-RAND are shown in Table 2.

Table 1
Data for the example

Item i	1	2	3	4	5	6
D_i	10,000	5,000	3,000	1,000	600	200
s_i	45	46	47	44	45	47
h_i	1	1	1	1	1	1
b_I	6.25	6.25	6.25	6.25	6.25	6.25

Table 2
The procedure of the C-RAND

T_j	Iteration r	$\lambda(r)$	$k_i(r)$ in step 5	T^0	T^1	New T_j	TC
$T_1 = 0.0949$	1	0	1, 2, 2, 3, 4, 7	0.1406	0.1220	0.1220	
	2	0.026315	1, 1, 1, 2, 3, 5	0.1836	0.1754	0.1754	
	3	0.007643	1, 1, 1, 2, 2, 4	0.1893	0.1818	0.1818	
	4	0.006735	1, 1, 1, 2, 2, 4	0.1893	0.1818	0.1818	\$4168.4
$T_2 = 0.1259$	1	0	1, 1, 1, 2, 3, 5	0.1836	0.1754	0.1754	
	2	0.007643	1, 1, 1, 2, 2, 4	0.1893	0.1818	0.1818	
	3	0.006735	1, 1, 1, 2, 2, 4	0.1893	0.1818	0.1818	\$4168.4
$T_3 = 0.1568$	1	0	1, 1, 1, 2, 3, 4	0.1850	0.1770	0.1770	
	2	0.007406	1, 1, 1, 2, 2, 4	0.1893	0.1818	0.1818	
	3	0.006735	1, 1, 1, 2, 2, 4	0.1893	0.1818	0.1818	\$4168.4
$T_4 = 0.1878$	1	0	1, 1, 1, 2, 2, 4	0.1893	0.1818	0.1818	
	2	0.006735	1, 1, 1, 2, 2, 4	0.1893	0.1818	0.1818	\$4168.4
$T_5 = 0.2188$	1	0	1, 1, 1, 1, 2, 3	0.2010	0.1923	0.1923	
	2	0.007395	1, 1, 1, 2, 2, 3	0.1911	0.1835	0.1835	
	3	0.006800	1, 1, 1, 2, 2, 4	0.1893	0.1818	0.1818	
	4	0.006735	1, 1, 1, 2, 2, 4	0.1893	0.1818	0.1818	\$4168.4

Fig. 1 shows the structure of our genetic algorithm for the same example.

For this example, the population size of 10 is used and the probability of crossover and mutation is set to 0.6 and 0.2, respectively. The termination condition is to stop if no improvement is made in 50 generations. Both representation and decoding solution of our best GA chromosome are shown in Table 3.

We can confirm that both the C-RAND and the GA obtain the same solution as Goyal's algorithm for this numerical example.

5. Computational experiments

In this section we compare the performances of three algorithms for a number of randomly generated JRPs with one constraint. Demand, minor ordering cost and holding cost are generated from uniform distribution on the ranges [100, 100 000], [0.5, 5.0] and [0.2, 3.0] respectively. Four different values of n (10, 20, 30 and 50) and four values of S (5, 10, 15 and 20) are considered. For the suitable constraint, B is generated from uniform distribution on the range $[n \times 200, n \times 800]$ and every b_i is considered as 1. For each combination of n and S, 100 problems are generated and solved using Goyal's algorithm [5], the C-RAND and the GA for a total of 1600 problems. A value of $m = 20$ is used in the C-RAND. In the GA, the population size of 50 is used for solving the small size problems ($n = 10$, 20 and 30). The population size of 100 is used for the large size problems ($n = 50$). The probability of crossover and mutation is set to 0.6 and

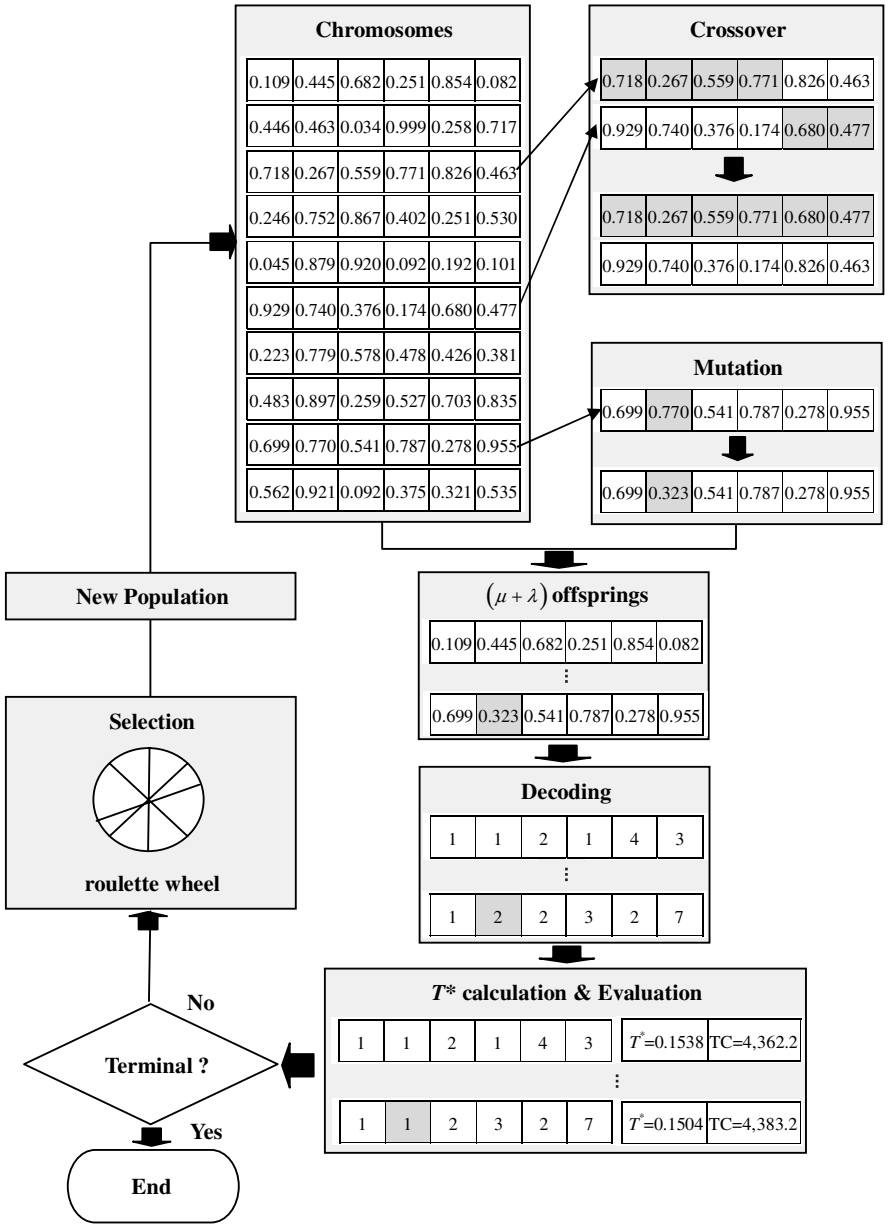

Fig. 1. The structure of the proposed genetic algorithm.

0.2, respectively. The termination condition is to stop if no improvement is made in 50 generations. A summary of computational results is shown in Table 4.

As shown in Table 4, the C-RAND is superior to other two algorithms. Especially, we can confirm that the C-RAND always outperforms the Goyal's algorithm [5] for large m. And for large value of S, we can see the performances of three algorithms are similar. The GA performs very well relative to the C-RAND. In addition, the GA can be easily expanded to the problems with several constraints. In order to solve the JRP with several constraints, both the C-RAND and Goyal's algorithm [5] must be changed by using more

Table 3
Representation and decoding solution of the best GA chromosome

Item	1	2	3	4	5	6
Best GA chromosome	0.446	0.463	0.092	0.426	0.252	0.381
k_i^{LB}	1	1	1	1	2	3
k_i^{UB}	1	2	2	3	4	7
Decoding solution (k_i)	1	1	1	2	2	4
Optimal T	0.1818					
TC	\$4168.4					

Table 4
Comparison of three algorithms (% improvement)

N	S	I	II	Best solution problems			GA better than Goyal		C-RAND better than Goyal		C-RAND better than GA	
				Goyal	GA	C-RAND	Maximum	Average	Maximum	Average	Maximum	Average
10	5	50	50	78	99	100	1.8736	0.0747	1.8736	0.0748	0.0061	0.0001
	10	42	58	86	100	100	1.1509	0.0288	1.1509	0.0288	0.0000	0.0000
	15	43	57	91	99	100	1.2648	0.0200	1.2648	0.0200	0.0026	0.0000
	20	36	64	91	99	98	0.1637	0.0063	0.1637	0.0052	0.0176	−0.0011
20	5	60	40	66	86	100	1.2669	0.0844	1.2669	0.1406	1.5142	0.0567
	10	60	40	71	90	99	1.0122	0.0269	1.0122	0.0366	0.3290	0.0098
	15	55	45	80	83	100	0.5995	−0.0017	0.5995	0.0138	0.6872	0.0156
	20	47	53	88	95	100	0.3539	0.0071	0.3539	0.0086	0.0997	0.0014
30	5	51	49	46	57	100	1.7119	0.0218	1.7119	0.1653	2.3667	0.1449
	10	56	44	74	70	99	0.5010	−0.0099	0.5010	0.0198	1.0502	0.0298
	15	59	41	79	73	100	0.2761	0.0011	0.2966	0.0159	0.3539	0.0148
	20	51	49	87	82	100	0.1418	−0.0032	0.1418	0.0034	0.4455	0.0066
50	5	61	39	34	51	96	2.1966	0.1839	2.1966	0.2755	1.2439	0.0921
	10	61	39	59	67	100	0.6394	−0.0069	0.6403	0.0471	0.9690	0.0542
	15	55	45	57	66	95	0.2635	−0.0105	0.9253	0.0323	1.0124	0.0430
	20	46	54	74	81	98	0.2139	0.0063	0.2139	0.0089	0.0982	0.0026
Maximum							2.1966		2.1966		2.3667	
Average		52	48	73	81	99		0.0268		0.0560		0.0294

I: The number of the problems that the optimal basic cycle time T is $T_0(T^* = T^0)$.
II: The number of the problems that the optimal basic cycle time T is $T_1(T^* = T^1)$.
In each combination of n and S, I + II = 100.

Lagrangian multipliers. But we can easily apply the GA to the multiple constraints case by only changing Proposition 1. As shown in Proposition 1, we can easily prove that the optimal basic cycle time T is $T^* = \min(T^0, T^1, T^2, \ldots, T^w)$ in the JRPs with w constraints.

6. Concluding remarks

Though resource restrictions are realistic in the production/inventory system, there have been very few studies in this area. This paper focuses on the development of the efficient algorithm for solving the constrained JRPs. We also introduce the GA approach to handle the constraints in the JRPs. Despite the

GA is inferior to the C-RAND, the GA approach is very meaningful for the constrained JRPs due to the several advantages, especially its extension ability.

Acknowledgments

The authors are very grateful to the anonymous referee who provided useful comments to improve our manuscript. This work was supported by grant No. R05-2004-000-10850-0 from the Basic Research Program of the Korea Science Engineering Foundation (KOSEF).

References

[1] E. Arkin, D. Joneja, R. Roundy, Computational complexity of uncapacitated multi-echelon production planning problems, Operations Research Letters 8 (1989) 61–66.
[2] H. Aytug, M. Khouja, F. Vergara, Use of genetic algorithms to solve production and operations management problems: A review, International Journal of Production Research 41 (2003) 3955–4009.
[3] S. Goyal, Determination of economic packaging frequency for items jointly replenished, Management Science 20 (1973) 232–238.
[4] S. Goyal, Determination of optimum packaging frequency of items jointly replenished, Management Science 23 (1974) 436–443.
[5] S. Goyal, Analysis of joint replenishment inventory systems with resource restriction, Operations Research Quarterly 26 (1975) 197–203.
[6] S. Goyal, S. Deshmukh, A note on the economic ordering quantity for jointly replenished items, International Journal of Production Research 31 (1993) 2959–2961.
[7] M. Kaspi, M. Rosenblatt, An improvement of Silver's algorithm for the joint replenishment problem, IIE Transactions 15 (1983) 264–269.
[8] M. Kaspi, M. Rosenblatt, On the economic ordering quantity for jointly replenished items, International Journal of Production Research 29 (1991) 107–114.
[9] M. Khouja, Z. Michalewicz, S. Satoskar, A comparison between genetic algorithms and the RAND method for solving the joint replenishment problem, Production Planning & Control 11 (2000) 556–564.
[10] E. Silver, Modifying the economic order quantity (EOQ) to handle coordinated replenishment of two or more items, Production & Inventory Management 16 (1975) 26–38.
[11] E. Silver, A simple method of determining order quantities in jointly replenishments under deterministic demand, Management Science 22 (1976) 1351–1361.
[12] M. Van Eijs, A note on the joint replenishment problem under constant demand, Journal of Operational Research Society 44 (1993) 185–191.