Improved Golden-Section Algorithm for the Multi-Item Replenishment Problem

S. Hernández*[1], I. Flores[2], J.A. Vázquez[1]

[1] Departamento de Ingeniería Industrial
Instituto Tecnológico de Celaya
Antonio García Cubas s/n, Celaya, México, C.P. 38010
*salvador.hernandez@itcelaya.edu.mx
[2] División de Estudios de Posgrado
Facultad de Ingeniería
Universidad Nacional Autónoma de México
Circuito Universitario s/n, Distrito Federal, México, C.P. 04510.

ABSTRACT
This paper presents a procedure for solving instances of the joint replenishment problem using the golden-section method. The algorithm includes an iterative method for obtaining a narrowing search range for the continuous variable in order to carry out less iterations. We studied the behavior of the algorithm experimentally and made comparisons with the heuristic technique known as RAND, solving randomly-generated problems. The results showed that the golden-section algorithm with the proposed improvements obtains the optimum solution for up to 100% of the problems solved, it is very stable when faced with the increase in the number of products in the problem and the runtime is notably competitive. The procedure is easy to implement and useful for professionals working in planning.

Keywords: line search methods, golden section, inventory management, joint replenishment

RESUMEN
Se muestra un método basado en sección dorada para resolver instancias del problema de reaprovisionamiento de productos múltiples. El algoritmo incluye un método iterativo para obtener un intervalo de búsqueda más pequeño. Se estudió el desempeño del algoritmo de manera experimental realizando las comparaciones con el algoritmo RAND, resolviendo instancias generadas aleatoriamente. Los resultados muestran que el algoritmo de sección dorada obtiene la solución óptima hasta en el 100% de las instancias resueltas, es estable frente al número de productos y el tiempo de ejecución es competitivo. El algoritmo es sencillo de implementar y muy útil para profesionistas dedicados a la planeación y control de inventarios.

1. Introduction

The problem of determining the frequency of production or packaging in systems of more than one product is known as the multi-item replenishment problem [1]. This problem is very important for inventory control and has been widely studied over the last few decades. Since the publication of Goyal's enumerative algorithm in 1974[1], a variety of heuristic procedures have been proposed for the multi-item replenishment problem (JRP): for example, the Silver method [2], based on the comparison of the costs of activating the purchase order, the costs of maintaining inventory and demand for the products; later, Kaspi and Rosenblatt proposed and studied a very efficient algorithm known as RAND in three consecutive articles [3, 4, 5].

Other contributions are the early version of Goyal's algorithm presented in 1973 [6]. Goyal and Deshmukh [7] performed an extensive series of tests with the RAND algorithm and the Silver´s algorithm setting the efficency of the first one, Hariga [8] proposed an heuristic procedure based on relaxation of the problem. Nisson et al [9]

proposed to group the items according to its frequency to perform the search; later in Nisson and Silver [10], there is a proposal of an implementation on a spreadsheet scheme, which is in fact the first implementation on a platform used by practitioners and managers.

We must mention briefly the applications of metaheuristics techniques: genetic algorithms were implemented by Khouja, Michalewicz and Satoskar [11] (using a wide search range for the variables) and Olsen [12] (with an strategy of direct grouping), but the comparisons against the RAND algorithm showed that the quality of the solution obtained with these techniques decrease rapidly with the size of the problem; hence, deeper research is needed in this area. Khouja and Goyal [13] reviewed the literature on this problem that was available up until 2005 including variants of the model with one constraint and models with stochastic demand.

The problem with heuristics is the deterioration in the quality of the solution: the hardest problems to solve are the ones where the value of the major activation cost (*S*) is low, while, at the same time, the size of the problem also has an effect. However, the RAND algorithm is characterized by being very robust and requiring very little computational effort and is used as a reference to compare the new proposals for heuristic algorithms[10]. In this paper we employ the golden-section method with some improvements and compare its behavior with the aforementioned RAND procedure. The advantage of the golden-section method lies in the fact that the solution obtained in an iteration does not depend on the result of the previous iteration.

Some applications of the golden-section algorithm can be found in Kabiriana and Ólafssonb [14] for optimization via simulation, Cai et al[15] in calculations for the determination of activation energy, Benavolia, Chiscib and Farinac [16] for noise filtration, and Tsai, Kolibal and Li [17] for the calculation of shape parameters in equations.

2. Optimization model

Before presenting the cost model, the following notation shall be defined:
TC: total cost.
i: product index.
n: number of products.
S: major activation cost, regardless of the number of products included in the order.
s_i: lower activation cost for product *i.*
D_i: demand for product *i.*
h_i: inventory carrying cost for product *i.*
T: base time cycle, continuous variable.
k_i: order frequency for product *i*, integer variable.
m: number of segments.
r: iteration.
$T_{\min}$: lower bound for the base time cycle.
$T_{\max}$: upper bound for the base time cycle.
T_a: improved lower bound for the base time cycle.
T_b: improved upper bound for the base time cycle.
$[a,b]$: uncertainty range.
U: uniform probability function.
Δ : stopping criteria.

The relevant costs are the activation cost and the inventory carrying cost; the optimization model is as follows:

$$\min \quad TC = \frac{1}{T}\left(S + \sum_{i=1}^{n} \frac{s_i}{k_i}\right) + \frac{1}{2} T \sum_{i=1}^{n} k_i D_i h_i \qquad (1)$$

$$T > 0 \qquad (2)$$

$$\text{subject to:} \quad k_i \geq 1, \text{ integers}, \forall\ i = 1,2,...,n. \qquad (3)$$

In the JRP, the order frequency k_i (an integer value) of each product and the base time cycle T^* (the time between two consecutive orders, continous variable) must be calculated, minimizing the total order and inventory carrying costs. The model is non-convex (Figure 1); however, for a given value of the frequencies the total cost is a convex function of the base time cycle *T* [18, 19]. The search range for the continuous variable [$T_{\min}$, $T_{\max}$] is determined using the following equations as proposed in [1] and corresponds to the strict cyclic policy:

$$T_{\min} = \min T_i = \min\left\{\sqrt{\frac{2s_i}{h_i D_i}}\right\} \qquad (4)$$

$$T_{\max} = \max\left\{\sqrt{\frac{2\left(S + \sum_{i=1}^{n} s_i\right)}{\sum_{i=1}^{n} D_i h_i}}\right\} \qquad (5)$$

3. The RAND heuristic

The RAND heuristic divides the search range [T_{min}, T_{max}] into m segments, then carries out a local search all along segment r: at a given value of T, the optimum order frequencies are calculated for each product using Equation (6), rounding up to the integer value, applying Equation (7)[1]; with the frequency values calculated, we obtain the new value of T; these steps are repeated and the solution converges [3]. As can be seen, the solution obtained in each iteration depends on the solution obtained in the previous iteration.

$$k_i = \frac{2s_i / D_i h_i}{T^2} \qquad (6)$$

$$\sqrt{k_i(k_i - 1)} \le k_i < \sqrt{k_i(k_i + 1)} \qquad (7)$$

4. Search range

In order to reduce the computational effort, a variety of methods have been proposed to determine a more narrow search range $[T_a, T_b]$, one example is the iterative method proposed by Viswanathan [18]. He observed that for moderate values of the major setup cost (S), the total cost TC monotonically decreases from T_{min} to a value T_a, and monotonically increases from a value T_b to T_{max} (Figure 1). The iterative procedure is executed only until the improvement in the bounds is above a certain predetermined value.

The new bounds reduce considerably the amount of operations needed, specially if an enumerative algorithm (like Goyal´s) is used.

By observing the shape of the objective function, we concluded that the search range $[T_a, T_b]$ can also be divided into segments (Figure 1). However, instead of implementing a local search, similar to the one used by the RAND algorithm, one viable solution is to apply a numerical method, such as the golden-section algorithm. The advantage of this type of procedures lies in the fact that the solution obtained in an iteration does not depend on the result of the previous iteration and also that at given frequencies the cost function is convex over the continuous variable T. In our case, we selected the golden-section algorithm because of its simplicity and speed of convergence.

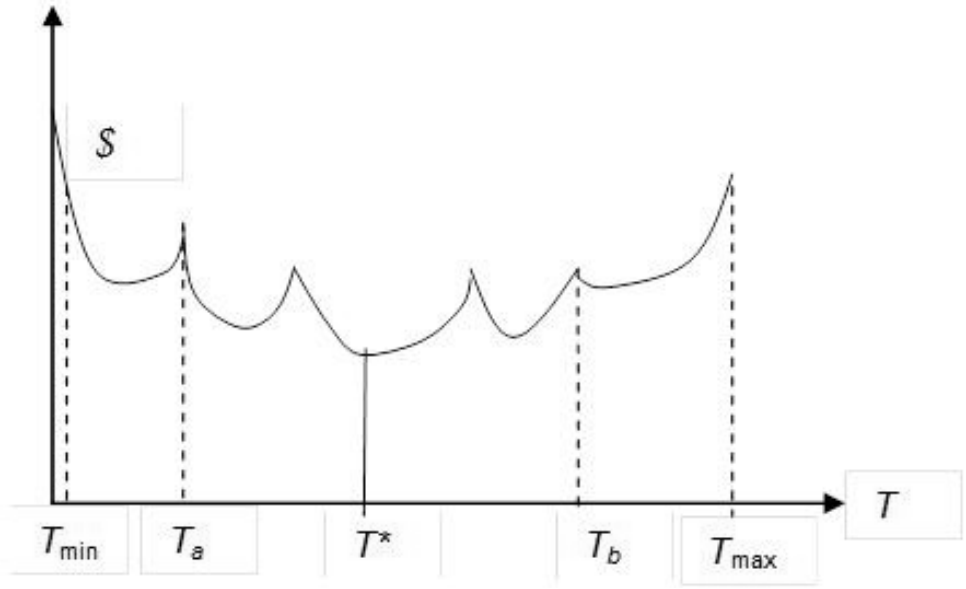

Figure 1. Approximate shape of the objective function.

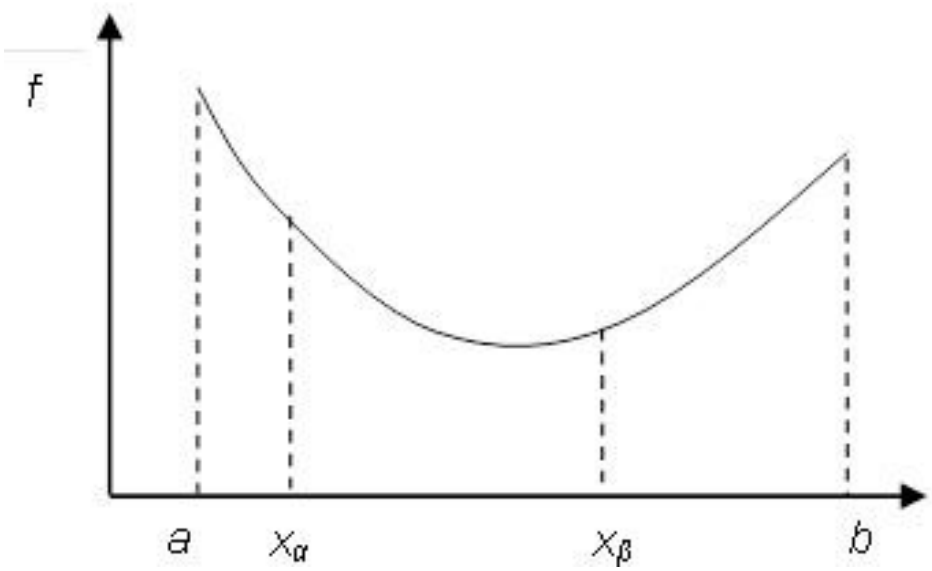

Figure 2. Quasiconvex function.

5. The golden-section method

The golden-section algorithm is a numerical procedure that calculates the optimum of a function within a range $[a,b]$ (called uncertainty range), it is not necessary to calculate the derivative of the function although the function needs to be quasiconvex in a certain region (Figure 2). The golden-section procedure assesses the objective function at points x_α and x_β that are to be found within the uncertainty range, these points are obtained with Equations (8) and (9):

$$x_\alpha = b - \alpha(b-a) \tag{8}$$

$$x_\beta = a + \alpha(b-a) \tag{9}$$

If $f(x_\alpha) > f(x_\beta)$, then the region to the left of x_α is eliminated and $a = x_\alpha$, thus obtaining the new uncertainty range, otherwise, if $f(x_\alpha) < f(x_\beta)$, then $b = x_\beta$ and the region to the right of x_β is eliminated. In each iteration, the range is reduced by a constant amount, $\alpha = 0.618034$, known as the golden section. It is worth mentioning that the size of the uncertainty range does not depend on the result of the *r*-th iteration, and only one operation should be done in the *r* +1 iteration [20].

6. Improved golden-section procedure

The heuristic procedure proposed in this report shall be denoted as an GS-V algorithm. The procedure calculates the initial search range $[T_{\min}, T_{\max}]$ using Equations (4) and (5), the improved range $[T_a, T_b]$ is then obtained, employing the Viswanathan's iterative procedure. The range is then divided into *m* segments; the local optimum is obtained on segment *r* (the base time cycle *T** and the k_i frequencies for each product that minimize the cost) employing the golden section. The entire procedure is given below:

Step 1. Calculate $[T_{\min}, T_{\max}]$ using Equations (4) and (5).

Step 2. Apply the Viswanathan's iterative procedure [18] to obtain the narrowest range

$[T_a, T_b]$, divide it into *m* segments of the same size.

Step 3. Calculate the optimum of segment *r* employing the golden-section method:

a. Set the uncertainty range $[T_{a,r}, T_{b,r}]$ and set Δ.

b. Calculate $T_{\alpha,r}$ and $T_{\beta,r}$ employing Equations (8) and (9), calculate the respective values for frequency $k_{i,\alpha}$ and $k_{i,\beta}$ employing Equations (6) and (7).

c. Calculate the respective total cost $TC(T_{\alpha,r})$ and $TC(T_{\alpha,r})$ using Equation (1).

d. If $TC(T_{\alpha,r}) < TC(T_{\beta,r})$ then $T_{b,r} = T_{\beta,r}$ and $TC_r = TC_\alpha$, otherwise $T_{a,r} = T_{\alpha,r}$, and $TC_r = TC_\beta$. Repeat steps b, c and d until $TC(T_{\alpha,r}) - TC(T_{\beta,r}) < \Delta$.

Do $TC^* = TC_r$.

Step 4. Stop the algorithm if $TC^* > TC_{r-1}$. Otherwise $r = r + 1$ and repeat Step 3.

We must point out that we adapted the golden-section method in this paper for the problem without relaxing the constraint that the frequencies must be integer values [19].

n	10, 20, 30, 50
S	5, 10, 15, 20, 30
h	*U* (0.5 – 5)
s	*U* (2 – 3)
D	*U* (100 – 100,000)

Table 1. Values of number of products(*n*), major activation cost (*S*) and ranges for randomly generating the inventory carrying parameter (*h*), lower activation cost (*s*) and demand (*D*).

7. Experimental procedure

The GS-V and RAND procedures were encoded in Fortran 94 and the tests were done on a computer with a 2 GHz Intel-Pentium Dual Core processor with 2 Gb RAM. No benchmark of instances exists (as for example the economic lot scheduling problem, ELSP), instead the problems must be randomly generated following the guidelines established in [3] and [10]: values of the major activation cost (*S*) and the number of products (*n*) are shown in Table 1. For each combination of *S* and *n*, 100 problems were generated, randomly generating the cost inventory parameter (*h*), lower activation cost (*s*) and demand (*D*). In total, 2000 problems were generated.

In the case of the GS-V algorithm, we carried out tests with *m*=10, 20, 30 and 50 segments and a value of $\Delta = 0.01$, while for the RAND algorithm tests were done with *m*=10 and 20 segments. As a means of comparison, we determined the number of times each algorithm returned the optimum in each *m* test. The optimum was calculated for each problem using Goyal's procedure [1].

The results are given in Tables 2 and 3, which are described as follows: the first column shows the number of products, the second column corresponds to the *S* parameter, the third column corresponds to the number of problems generated for the *n* and *S* combination, columns 4-5 show the percentage of problems in which the GS-V algorithm returned the optimum solution for each *m* test; the last two columns show, for each *m* test, the percentage of problems in which the RAND algorithm returned the optimum.

The results obtained with GS-V10 and GS-V20 are as follows: the GS-V10 algorithm returns the optimum solution in 97.25% of the problems solved , which is a lot lower than that would be achieved with RAND10 (98.05%) and RAND20 (99.3%) and in the test with GS-V20 , 99% was achieved (Table 2).

Table 2 also shows that the GS-V10 algorithm obtains for *n*=50 products the worst results: when *S*=5, the GS-V10 algorithm returns the optimum solution in 79% of the problems solved, when *S*=10, it returns the optimum solution in 90% and, finally, for the values of *S*=15 and 20, the algorithm returns the optimum solution in 96 and 99% of the problems.

n	*S*	# of problems	GS-V10	GS-V20	RAND10	RAND20
10	5	100	100%	100%	100%	100%
	10	100	100	100	100	100
	15	100	100	100	100	100
	20	100	100	100	100	100
	30	100	100	100	100	100
20	5	100	95% (0.987)	100%	99% (0.85)	100%
	10	100	100	100	97 (1.92)	98 (1.97)
	15	100	100	100	100	100
	20	100	100	100	100	100
	30	100	100	100	100	100
30	5	100	87% (1.29)	96% (0.657)	96% (1.17)	100%
	10	100	99 (0.37)	100	95 (0.856)	99 (0.29)
	15	100	100	100	100	100
	20	100	100	100	98 (0.835)	100
	30	100	100	100	98 (0.40)	100
50	5	100	79% (1.07)	88% (0.51)	90% (1.94)	98% (0.185)
	10	100	90 (0.20)	97 (0.20)	95 (0.788)	97 (0.14)
	15	100	96 (0.45)	99 (0.21)	96 (2.07)	97 (1.57)
	20	100	99 (0.63)	100	97 (1.43)	98 (0.645)
	30	100	100	100	99 (0.77)	99 (0.18)
		% Global	97.25%	99%	98.05%	99.30%
	Frequency	2000	1945	1980	1961	1986

Table 2. Test results with GS-V10 and GS-V20: % of optimums obtained per number of products.

The average difference in respect of the optimum solution (some authors call it the penalty cost and it is to be found in brackets in the same table) varies but in general behaves in the same way: it is large for low values of *S*. The tests using GS-V20 give better results as they achieve a global 99% for the number of times the optimum solution is obtained. However, this is slightly lower than results of the test using RAND20, in which a percentage of 99.35 is achieved (Table 2). No other test was done using the RAND procedure since (according to the authors) the improvements that can be obtained in penalty costs do not justify the additional computational effort [5]. The GS-V10 and GS-V20 algorithm behave favorably for high values of *S*, and its behavior deteriorates when the *S* parameter is reduced. In this respect, it is important to point out that the hardest problems are precisely the ones where the value of *S* is low, which has been reported on previous occasions [10].

n	*S*	# of problems	GS-V30	GS-V50	RAND10	RAND20
10	5	100	100%	100%	100%	100%
	10	100	100	100	100	100
	15	100	100	100	100	100
	20	100	100	100	100	100
	30	100	100	100	100	100
20	5	100	100%	100%	99% (0.85)	100%
	10	100	100	100	97 (1.92)	98 (1.97)
	15	100	100	100	100	100
	20	100	100	100	100	100
	30	100	100	100	100	100
30	5	100	100%	100%	96% (1.17)	100%
	10	100	100	100	95 (0.856)	99 (0.29)
	15	100	100	100	100	100
	20	100	100	100	98 (0.835)	100
	30	100	100	100	98 (0.40)	100
50	5	100	89% (0.63)	100%	90% (1.94)	98% (0.185)
	10	100	98 (0.35)	100	95 (0.788)	97 (0.14)
	15	100	100	100	96 (2.07)	97 (1.57)
	20	100	100	100	97 (1.43)	98 (0.645)
	30	100	100	100	99 (0.77)	99 (0.18)
		% Global	99.35%	100%	98.05%	99.30%
	Frequency	2000	1987	2000	1961	1986

Table 3. Test results with GS-V30 and GS-V50: % of optimums obtained per number of products

In the test with GS-V30, we obtained a small increase in the global result achieving 99.35% (Table 3); finally, in the test with GS-V50 the procedure returns the optimum solution in 100% of the problems solved (Table 3). Table 3 also shows that for n=50 and S=5, the GS-V30 algorithm returns the optimum solution in 89% of the problems and when S= 10 it returned 98%. Finally, in the GS-V50 test, we obtained the optimum in 100% of the problems.

The GS-V algorithm is very stable, in other words the difficulty in returning a quality solution is not significantly affected when the size of n is increased, by comparison, the RAND algorithm is very obviously affected. As can be seen in Figure 3, the optimum solution is obtained in all the problems solved in the test with GS-V50 and, as shown below, a shorter runtime is required.

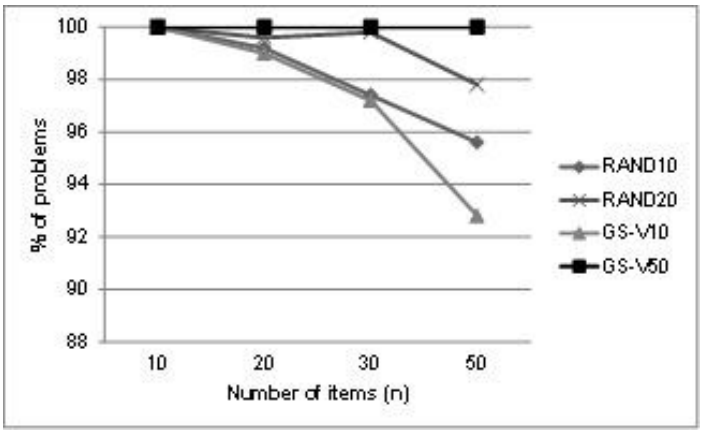

Figure 3. Percentage de optimums returned by both algorithms.

Table 4 shows the available runtimes for each test and for each value of n. We observe that the time is shorter for all the tests with GS-V than that required by the RAND algorithm.

	Test					
n	GS-V10	GS-V20	GS-V30	GS-V50	RAND10	RAND20
10	0.54	0.54	0.56	0.66	0.54	0.68
20	0.62	0.62	0.68	0.78	0.78	1.6
30	0.72	0.8	0.9	0.98	1.08	2.5
50	0.98	1.12	1.24	1.5	1.84	3.04

Table 4. Available runtime (miliseconds).

Figure 4 compares runtimes for GS-V10, GS-V50 tests (where the optimum is obtained in all the problems), RAND10 and RAND20. It is apparent that more time is required in the tests using the RAND algorithm.

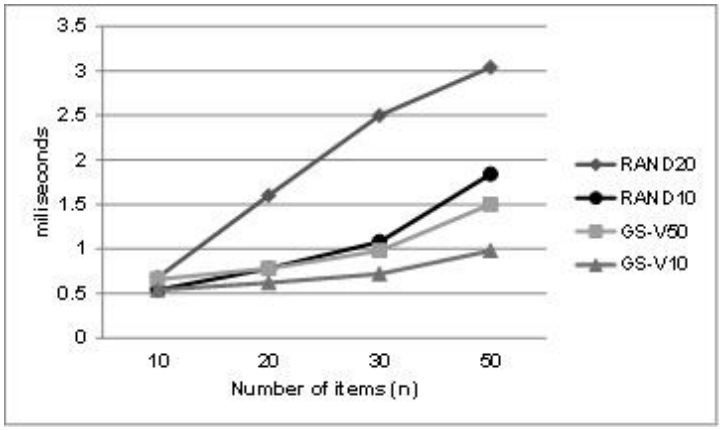

Figure 4. RAND10, RAND20, GS-V10 and GS-V50 runtimes.

Table 5 details the runtime per number of products: for a value of *n*=10 the runtime for the test using GS-V50 is slightly less than the time reported with RAND20; for *n*=20 products the difference is 48.75%; for *n*=30 products, the difference in the times is 39.20%, finally for *n*= 50 products, the difference is 49.34%.

n	GS-V50 (milisec.)	RAND20 (milisec.)	$100\left[\frac{time_{GS-V}}{time_{RAND20}}\right]$
10	0.66	0.68	97.06%
20	0.78	1.6	48.75%
30	0.98	2.5	39.20%
50	1.5	3.04	49.34%

Table 5. Percentage of difference in the runtimes: GS-V50 vs RAND20

8. Conclusions

This article proposes an application of the GS method to solve instances of the JRP. Unlike other procedures, no exhaustive enumeration is employed and the result of one iteration is not affected by the result of a previous iteration. The algorithm was adapted to solve the unrelaxed problem and also includes an iterative method for obtaining narrower search range for the continuous variable. The results for the tests with 2000 randomly generated problems show that the GS-V algorithm can even return the optimum solution for up to 100% of the problems and is very stable when the number of products n is increased. The average required runtime is less in the tests using the GS-V algorithm, even if the number of segments (*m*) is increased or the number of products (*n*) increased. The algorithm can be easily adapted and will be very useful to professionals in charge of planning inventories; also, it has applications for the constrained problem or the ELSP problem.

References

[1] Goyal S.K., Determination of optimum packaging frequency for items jointly replenished, Management Science, Vol. 21, No. 4, 1998, pp. 436-443.

[2] Silver, E., A simple method of determining order quantities in joint replenishments under deterministic demand. Management Science, Vol. 22, No. 12, 1976, pp. 1351-1361.

[3] Kaspi M., Rosenblatt M.J., An improvement of Silver's algorithm for the joint replenishment problem. IIE Transactions, Vol. 15, No. 3, 1983, pp.264–267.

[4] Kaspi M, Rosenblatt M.J., The effectiveness of heuristic algorithms for multi-item inventory systems with joint replenishment costs. International Journal of Production Research, Vol. 23, No. 1, 1985, pp. 109-116.

[5] Kaspi M., Rosenblatt M. J., On the economic ordering quantity for jointly replenished items. International Journal of Production Research, Vol. 29, No. 1, 1991, pp. 107–114.

[6] Goyal S.K., Determination of economic packaging frequency of items jointly replenished. Management Science, Vol. 20, No. 2, 1973, pp. 232-235.

[7] Goyal S.K. and Deshmukh S.G., A note on: the economic ordering quantity for jointly replenished items. International Journal of Production Research, Vol. 31, No. 12, 1993, pp.2959-2961.

[8] Hariga M., Two new heuristic procedures for the joint replenishment problem. Journal of the Operational Research Society, Vol. 45, No. 4, 1994, pp. 463–471.

[9] Nisson A., Segersted A., van der Sluis E., A new iterative heuristic to solve the joint replenishment problem using a spreadsheet technique. International Journal of Production Economics, No. 108, No. 1-2, 2007, pp. 399 - 405.

[10] Nisson A., Silver E., A simple improvement on Silver's heuristic for the joint replenishment problem. Journal of the Operational Research Society, Vol. 59, No. 10, 2008, pp.1415-1421.

[11] Khouja M., Michalewicz Z., Satoskar S.,A comparison between genetic algorithms and the RAND method for solving the joint replenishment problem. Production Planning and Control, Vol.11, No. 6, 2000, pp.556-564.

[12] Olsen A.L., An evolutionary algorithm to solve the joint replenishment problem using direct grouping. Computers & Industrial Engineering, Vol.48, No. 2, 2005, pp. 223–235.

[13] Khouja M., Goyal S., A review of the joint replenishment problem literature: 1989–2005. European Journal of Operational Research, No. 186, No. 1, 2008, pp. 1- 16.

[14] Kabiriana A, Ólafssonb S, Continuous optimization via simulation using Golden Region search European Journal of Operational Research. Vol.208, No.1, 2011, pp. 19–27.

[15] Cai J., Han D., Chen Ch., Chen S., Application of the golden section search algorithm in the nonlinear isoconversional calculations to the determination of the activation energy from nonisothermal kinetic conversion data. Solid State Sciences, Vol. 12, No. 5, 2010, pp. 829-833.

[16] Benavolia A., Chiscib L., Farinac A., Fibonacci sequence, golden section, Kalman filter and optimal control. Signal Processing, Vol. 89, No. 8, 2009, pp. 1483–1488.

[17] Tsai C.H, Kolibal J, Li M., The golden section search algorithm for finding a good shape parameter for meshless collocation methods. Engineering Analysis with Boundary Elements, Vol. 34, No. 8, 2010, pp. 738-746.

[18] Viswanathan S., A new optimal algorithm for the joint replenishment problem. Journal of the Operational Research Society, Vol. 47, No. 7, 1996, pp. 936 - 934.

[19] Frenk J.B.G., Kleijn M.J., Dekker R., An efficient algorithm for a generalized joint replenishment problem. European Journal of Operational Research, No.118, No. 2, 1999, pp. 413- 428.

[20] Bazaraa M., Sherali H., Shetty, C.M., Nonlinear programming. John Wiley and Sons, 2006, pp. 343 – 351.