On the efficiency of optimal algorithms for the joint replenishment problem: a comparative study

Eric Porras*, Rommert Dekker
Econometric Institute, Tinbergen Institute, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

Econometric Institute Report EI 2004-33

Abstract
In this paper we proposed an efficient algorithm to solve the joint replenishment problem to optimality. We perform a computational study to compare the performance of the proposed algorithm with the best one reported in Viswanathan [6]. The study reveals that for large minor set-up costs and moderate major set-up cost, our algorithm outperforms the latter.
Keywords: inventory; joint replenishment; deterministic demand

1. Introduction

The joint replenishment problem (JRP) has been extensively studied in the literature under cyclic strategies (also called general integer (GI) policies). This problem considers M items that can be jointly replenished against a major set up cost S. It is assumed that each item j ($j = 1,\ldots,M$) included in a single order causes a minor set-up cost s_j and has an associated holding cost h_j. Demands for the items occur at constant rates D_j and no backorders are allowed. Items are ordered every k_jT units of time, where T is called the basic cycle time and k_j is an integer multiple of T.

Although a number of heuristics are available to solve the JRP, optimal solutions are few. Most of these methods are based on an algorithm first proposed by Goyal [3], which is based on enumeration of the total cost function between a lower and an upper bound of T. As pointed out by van Eijs [2] and Andres and Emmons [1], the algorithm by Goyal [3] can guarantee optimal solutions only for strict-cyclic policies, in which at least one family item is included in every replenishment occasion. Van Eijs [2] proposed a modified version of Goyal's algorithm for cyclic strategies, where an explicit formula is introduced to obtain the intervals over which the total cost is enumerated. The pitfall of the algorithms by Goyal and van Eijs is that for large number of items and relatively high minor set-up costs, they require a large number of enumerations. Viswanathan [5] and Wildeman et al. [7] proposed the use of tighter bounds for the basic cycle time. In this paper, we incorporate the bounds suggested by Wildeman in an algorithm similar to van Eijs [2]. Viswanathan [6] presented a comparative study of the performance of different methods until 2002. However, he did not cite the work by Wildeman et al. [7]. Thus, our objective is to perform a similar study to compare our method with the best one reported by Viswanathan [6].

* Corresponding author: porrasmusalem@few.eur.nl

2. JRP formulation

We consider the following formulation for the JRP:

$$(P)\ \text{Min}\ TC(T,\mathbf{k}) = \frac{S}{T} + \sum_{j=1}^{M}\left(\frac{s_j}{k_j T} + \frac{1}{2}h_j k_j D_j T\right)$$

s.t. $T > 0$

$k_j \geq 1$ integers for $j = 1, 2, \ldots, M$

where $\mathbf{k}$ is the vector of the k_j's.

Note that the function $TC(T,\mathbf{k})$ is not jointly convex with respect to T and $\mathbf{k}$. However, for a fixed vector $\mathbf{k}$ the function $TC(T)$ is convex in T, with optimal T given by:

$$T^*(k_1,\ldots,k_M) = \sqrt{\frac{2\left(S + \sum_{j=1}^{M}\frac{s_j}{k_j}\right)}{\sum_{j=1}^{M} h_j D_j k_j}} \qquad (2)$$

Substituting (2) back in (1) we get the optimal TC for a fixed $\mathbf{k}$:

$$TC(k_1,\ldots,k_M) = \sqrt{2\left(S + \sum_{j=1}^{M}\frac{s_j}{k_j}\right)\left(\sum_{j=1}^{M} h_j D_j k_j\right)} \qquad (3)$$

2.1 *Solution method for problem* (*P*)

We develop a solution method similar to van Eijs [2] but based on a formulation of problem (*P*) given by Wildeman et al. [7]. This formulation gives a better insight into the behaviour of the function *TC* and has not been used by other authors. Wildeman et al. [7] gives the following equivalent formulation of problem (*P*):

$$(P)\ \text{Min}\ TC(T) = \frac{S}{T} + \sum_{j=1}^{M} z_j(T),\ \text{s.t.}\ T > 0$$

where the functions $z_j(T)$ given by:

$$z_j(T) = \min_{k_j}\left\{\frac{s_j}{k_j T} + \frac{1}{2}h_j D_j k_j T\right\} : k_j \geq 1 \text{ integers for } j = 1,\ldots, M.$$

For given T, an optimal value of $\mathbf{k}$ is given in Wildeman et al. [7] by:

$$k_j(T) = \left\lceil -\frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{8b_j}{T^2}} \right\rceil \text{ for } j = 1,\ldots, M. \qquad (4)$$

where $b_j = \frac{s_j}{h_j D_j}$.

Although formula (4) can be easily derived from the inequalities given by Goyal [3] for the optimal solution, it was not provided before in the literature to show the correctness of Goyal's algorithm. We refer to the Appendix for the details on this analysis. Thus, from (4) it follows that the vector **k** remains unchanged for T inside some interval $[T^{(i)}, T^{(i-1)})$, where $T^{(i)}$ can be obtained from:

$$T^{(i)} = \max_j \left\{ \sqrt{\frac{2b_j}{k_j^{(i-1)}(k_j^{(i-1)}+1)}} \right\} \qquad (5)$$

We call this vector $\mathbf{k}^{(i-1)}$ and $T^*_{(i-1)}$ the optimal T associated with it, as given by equation (2). Note that $T^*_{(i-1)}$ does not necessarily belongs to the interval $[T^{(i)}, T^{(i-1)})$. However, as stated in the following theorem, the overall optimal solution for TC has an associated optimal T, say T_{opt}, equal to some $T^*_{(i-1)}$. This result was not provided in previous papers to show that Goyal's algorithm and modified versions of it (including the one presented in this paper) provide indeed the optimal solution for the JRP. Moreover, this result may not hold for extension of the JRP, e.g. when a correction for empty replenishments is made in the total cost function (see Porras and Dekker [4]).

Theorem 1

Let $\mathbf{k}_{\text{opt}}$ be the vector of k_j^* values that minimize the function $TC(T,\mathbf{k})$ among all possible T values as given by equation (4). Let $[T^l_{opt}, T^u_{opt}]$ be the interval associated with $\mathbf{k}_{\text{opt}}$. Then $T_{opt} = T^*(k_1^*, k_2^*, ..., k_M^*) \in [T^l_{opt}, T^u_{opt}]$.

Proof

From (2) it follows that $T^*(k_1,\ldots,k_M)$ is monotone decreasing in **k**. Now let $\mathbf{k}^{(-1)}$ be the adjacent locally optimal vector to $\mathbf{k}_{\text{opt}}$ for $T > T^u_{opt}$ and suppose that $T^*(k_1^*, k_2^*, ..., k_M^*) > T^u_{opt}$. By the convexity of $TC(T)$ it follows that TC is decreasing in $[T^l_{opt}, T^u_{opt}]$, which implies that TC is increasing for $T > T^u_{opt}$. Again by the convexity of $TC(T)$, this would imply that $T^*(\mathbf{k}^{(-1)}) < T^*(k_1^*, k_2^*, ..., k_M^*)$, which is a contradiction by the monotonicity of $T^*(k_1,\ldots,k_M)$. It follows that $T^*(k_1^*, k_2^*, ..., k_M^*) < T^u_{opt}$. Proceed in a similar way to show that $T^*(k_1^*, k_2^*, ..., k_M^*) > T^l_{opt}$, implying $T_{opt} = T^*(k_1^*, k_2^*, ..., k_M^*)$. ð

The previous analysis allows us to make a partition on the set $I = (0,\infty)$ of T values, with optimal solutions given by (4). Note that we do not need a full enumeration on **k**, since we only consider the vectors **k** that minimize the total cost for a given T. Therefore we only need to evaluate a finite number of intervals between a lower and an upper bound on T, say T_{low} and T_{upp}, respectively. We can obtain the local minima of TC with formula (3) inside each interval of such a partition, and compute the best solution among all intervals.

According to the previous discussion, the difference in performance for the various methods to solve the JRP, depends on their ability to get tighter bounds for the basic cycle time. In the next section we discuss the bounds used in Viswanathan's algorithm (referred to as **Visw**), and the bounds suggested by Wildeman et al. [7], which were implemented in the algorithm proposed in this paper (referred to as **Porras-Wild**).

2.2 Bounds on the basic cycle time

As shown by van Eijs [2], an upper bound on T can be obtained from:

$$T_{upp}^{(1)} = \sqrt{\frac{2(S + \sum_j s_j)}{\sum_j h_j D_j}}$$

This upper bound was first proposed by Goyal [3]. Note that for a large number of items and relatively high minor set-up costs, the previous upper bound can be very large. This increases considerably the computational effort to find the optimal TC. In **Visw**, a tighter upper bound, say $T_{upp}^{(V)}$, is obtained in the following way: starting in $T_{upp}^{(1)}$, find the first $T_{(i-1)}^{*}$ that lies inside the interval $[T^{(i)}, T^{(i-1)})$. The function TC will be monotone increasing between the overall optimal T and $T_{(i-1)}^{*}$ (Viswanathan [5]).

Van Eijs [2] proposed a lower bound on T for cyclic policies as follows:

$$T_{low,VE} = 2S / TC^{U}$$

where TC^{U} is an upper bound on the total cost $TC(T,\mathbf{k})$.

This lower bound can be improved further by inserting in the last equation the best value of TC found so far in each step of the optimization algorithm. However, even with this improvement technique, except for large values of the major set up cost and moderate minor set-up costs, the resulting lower bound can be very small. Therefore the computation effort to find the optimal solution increases considerably. Starting with $T_{low,VE}$, **Visw** finds a tighter lower bound on T, say $T_{low}^{(V)}$, by using a similar procedure as the one described for $T_{upp}^{(V)}$ (Viswanathan [5]).

To avoid a large number of enumerations to get the improved lower and upper bounds, **Visw** stops the search if before reaching the best possible lower (upper) bound, the ratio (T_{upp}/T_{low}) is below a predetermined value. Viswanathan [6] did not apply Wildeman bounds.

As we show in the computational results of section 3, **Visw** performs very well for a number of problem configurations. However, for moderate major set-up costs and relatively high minor set-up cost (magnitude order of 1 to 1 until 1 to 5), **Visw** can be outperformed by a faster algorithm that incorporates the bounds on T proposed by Wildeman et al. [7]. These lower and upper bounds on T are suitable for GI policies and a general cost structure. Wildeman et al. [7] proceeded in the following way:

First let $f_j(k_jT) = \frac{s_j}{k_jT} + \frac{1}{2}h_jD_jk_jT$ for $j = 1,\ldots, M$.

It is easy to verify that the function $f_j(k_jT)$ is strictly convex in T with a minimum for $T = x_j^*/k_j$, with

$$x_j^* = \sqrt{\frac{2s_j}{h_jD_j}}$$

Now a lower bound T_{low} is obtained from:

$$T_{low} = \frac{S}{TC(T(R),\mathbf{k}^{T(R)}) - \sum_{j=1}^{M}(\frac{s_j}{x_j^*} + \frac{1}{2}h_jD_jx_j^*)} \qquad (6)$$

where $T(R)$ is the optimal basic cycle time for the relaxation (R) of problem (P):

(R) min $TC^{(R)}(T,\mathbf{k})$

s.t. $T > 0$

$k_j \geq 1 \qquad j = 1, 2,\ldots, M$

where $TC^{(R)}(T,\mathbf{k})$ is defined in a similar way as $TC(T,\mathbf{k})$.

Note: A procedure to determine $T(R)$ can be found in Wildeman et al. [7].

The above procedure basically finds a locally optimal solution for the original function TC in $T=T(R)$, and then by bisection on the interval $(0,T(R)]$ a lower bound is obtained by using equation (6). It can be shown that the function $TC^{(R)}$ is convex in T (Wildeman et al. [7]) and therefore an upper bound on T, say T_{upp}, can be obtained by the same bisection procedure on the interval $[T(R), T_{upp}^{(1)}]$, whenever $T_{upp}^{(1)} > T_{upp}$.

2.3 Algorithm to solve (P) (**Porras-Wild**)

Step 0. Initialization
Evaluate Wildeman bounds T_{low} and T_{upp}.
Set $\mathbf{k}^{(0)} = \mathbf{k}(T_{upp})$ using equation (4).
Set $TC_{\min}^{(0)} = \infty$, $T^{(0)} = 8$ and $n = 1$.

Step 1. For $\mathbf{k}^{(n-1)}$ determine $T^{(n)}$ from:

$$T^{(n)} = \max_j \left\{ \sqrt{\frac{2b_j}{k_j^{(n-1)}(k_j^{(n-1)}+1)}} \right\} \text{ and set:}$$

$$J^{(n)} = \left\{ j : \max_{j=1,\ldots,M} \left\{ \sqrt{\frac{2b_j}{k_j^{(n-1)}(k_j^{(n-1)}+1)}} \right\} \right\}$$

Evaluate: $T_{n-1}^{*} = \sqrt{\dfrac{2\left(S + \sum_j \dfrac{s_j}{k_j^{(n-1)}}\right)}{\sum_j h_j D_j k_j^{(n-1)}}}$

Set: $TC_{\min}^{(n)} = \begin{cases} \min\{ TC_{\min}^{(n-1)}, TC(T_{n-1}^{*}, \mathbf{k}^{(n-1)}) \} & \text{if } T_{n-1}^{*} \in \left[T^{(n)}, T^{(n-1)} \right] \\ \infty & \text{otherwise} \end{cases}$

Obtain the elements of the new vector $\mathbf{k}^{(n)}$, according to:

$$k_j^{(n)} = \begin{cases} k_j^{(n-1)} + 1 & \text{for } j \in J^{(n)} \\ k_j^{(n-1)} & \text{for } j \notin J^{(n)} \end{cases}$$

Step 2. If $T^{(n)} \le T_{low}$ STOP with $TC_{\min}(T, \mathbf{k}) = TC_{\min}^{(n)}$ and $T_{opt} = T_{n-1}^{*}$.
Otherwise set $n = n + 1$ and GOTO step 1.

In the previous algorithm, the number of T intervals evaluated between T_{low} and T_{upp} depends on the number of $T^{(n)}$ evaluations. Since the maximum in $T^{(n)}$ is taken over M items in each round of the algorithm, it follows that the number n of steps increases only linearly in the number of items. Based on this, the following formula gives the maximum number of steps needed in the algorithm:

$$\textit{Maximum \# of steps} = \sum_{i:b_i \neq b_j} k_i(T_{low}) - k_i(T_{upp})$$

The previous formula was not provided previously in the literature of the JRP.

The previous algorithm to solve (*P*) is essentially the same as the one proposed by van Eijs [2], although implemented in a slightly different way. Our main contribution, apart from using better bounds on *T*, is that in every round of the algorithm we check for local optimality of the solution for each new T^{*} value, something that Goyal [3], van Eijs [2] and Viswanathan [5] omit in their methods. This may not have a great impact in the performance of the algorithm for *TC*, since the optimal solution is always associated with an optimal *T*. However, some CPU time can be saved by not evaluating the total cost in every round of the algorithm. On the other hand, checking for optimality is a necessary condition to get the optimal solution for a corrected version of problem *P*, say $P^{(c)}$, where the total cost function, say $TC^{(c)}$, includes a correction factor for empty replenishments (see Porras and Dekker [4] for a method to solve $P^{(c)}$ with minimum order quantities). When $P^{(c)}$ is solved to optimality, the solution need not be an element of the set of solutions given by equation (3), but one with associated $T=T^{(i)}$ from some interval $[T^{(i)}, T^{(i-1)})$. Accordingly, after defining proper bounds on *T*, we can still use **Porras-Wild** to minimize $TC^{(c)}$.

3. Computational results

We implemented the algorithms **Visw** and **Porras-Wild** as described in the last section. To be fair in the performance comparison of the two algorithms, we also include the feasibility check in **Visw**. We use the same experiment setting as Viswanathan [6], but with the inclusion of two extra values for the major set-up cost, so the values S = 0.5, 1, 5, 10, 20, 50, 100 were considered. The number of items considered were M = 10, 20, 50. For each value of S and M we generated 100 problems, with the minor set-up costs s_j and the unit holding costs h_j randomly generated from U[0.5,5] and U[0.2,2]. For each problem instance, demands for the individual items were randomly generated from U[100, 100000]. Therefore, a total of (7×3×100) = 2,100 problems were solved.

Remark
We consider an order of magnitude of the major set-up cost as compared to the minor set-up costs varying from 1:10 to 1:0.005. From numerical results, we found that for values of the minor set-up costs up to ten times the major set-up cost, the JRP is still relevant (savings up to 3.5% were achieved as compared to independent EOQ ordering). Therefore, the cost values considered in the experiment setting are relevant for the purpose of this study.

In Table 1 we present a comparison on the performance of **Visw** versus **Porras-Wild** for the experiment setting described above. The average number of intervals evaluated to get the optimal solution (including the ones needed to improve the bounds), the average lower bound, the average upper bound and the average CPU time is reported.

As we can see from Table 1, **Porras-Wild** dominated **Visw** for all problem in which the ratio of S to s_j is 1:10 to 1:5. For 10 and 20 items **Porras-Wild** outperformed **Visw** up to a ratio of 1:0.5 in all problems solved. For ratios of 1:0.25 up to 1:0.05 **Visw** outperformed **Porras-Wild**, except for M=10 items, in which both algorithms performed equally. We can conclude that as the number of items increases, the ratio of S to s_j is less relevant for the performance of **Visw**. However, even for a large number of items (M=50) when this ratio is in the range 1:10-1:5, **Porras-Wild** outperformed **Visw**.

4. Conclusions

In this paper we showed by numerical experiments that the proposed algorithm outperforms the best reported in Viswanathan [6] for moderate major set-up cost and relatively high values of the minor set-up costs. Our algorithm performs relatively well for a number of problem configurations in which the major set-up cost becomes more important. Only when the major set-up cost is in the order of 20 times the minor set-up costs, **Visw** clearly outperformed **Porras-Wild**. An important contribution of this paper is the formal proof that the algorithms presented in the literature for the JRP based on van Eijs' method find an overall optimal solution for *TC*.

Table 1. Comparison of JRP algorithms for determining the optimal cyclic policy with s_i~U[0.5,5]

		Average no. of intervals evaluated		*Average T_{low} (years)*		*Average T_{upp} (years)*		*Average CPU time (sec.)*	
M	*S*	*Visw*	*Porras-Wild*	*Visw*	*Porras-Wild*	*Visw*	*Porras-Wild*	*Visw*	*Porras-Wild*
10	0.5	78.3	31.2	0.0022	0.0033	0.0074	0.0077	1.7	0.8
	1	49.1	21.0	0.0034	0.0043	0.0078	0.0082	1.5	0.5
	5	14.4	10.2	0.0080	0.0067	0.0093	0.0100	0.5	0.3
	10	8.2	8.0	0.0101	0.0079	0.0107	0.0113	0.3	0.2
	20	4.6	6.0	0.0122	0.0093	0.0125	0.0131	0.2	0.2
	50	2.4	4.5	0.0163	0.0119	0.0167	0.0170	0.2	0.2
	100	2.0	3.8	0.0214	0.0147	0.0217	0.0219	0.2	0.1
20	0.5	179.8	102.1	0.0018	0.0023	0.0069	0.0071	13.7	7.8
	1	113.5	66.6	0.0026	0.0031	0.0070	0.0076	9.1	4.2
	5	34.3	27.0	0.0059	0.0055	0.0081	0.0090	2.3	1.5
	10	16.3	18.7	0.0084	0.0066	0.0092	0.0099	1.3	1.0
	20	9.3	14.1	0.0101	0.0078	0.0106	0.0113	0.7	0.8
	50	4.4	9.6	0.0130	0.0098	0.0133	0.0139	0.5	0.5
	100	2.6	7.2	0.0162	0.0118	0.0166	0.0170	0.3	0.5
50	0.5	532.2	440.3	0.0013	0.0014	0.0061	0.0063	364.4	333.9
	1	340.3	283.1	0.0019	0.0019	0.0062	0.0068	194.3	90.5
	5	100.4	105.1	0.0043	0.0038	0.0069	0.0080	27.6	29.2
	10	53.2	67.7	0.0060	0.0049	0.0076	0.0087	10.9	15.5
	20	23.0	46.9	0.0081	0.0060	0.0087	0.0095	4.1	9.2
	50	11.8	30.0	0.0101	0.0076	0.0105	0.0112	2.3	5.2
	100	6.1	21.7	0.0120	0.0090	0.0123	0.0130	1.5	3.8

Appendix

Motivation of the proposed algorithm

From equation (4) it follows that as T decreases, the vector $\mathbf{k}$ changes when one of its elements, say k_j, changes from k_j to $k_j + 1$. The new vector $\mathbf{k}$ will remain constant until the next element of $\mathbf{k}$ changes in one unit. In other words, $TC(T)$ is piecewise convex in T for the intervals in which the associated vectors $\mathbf{k}$'s remain unchanged.

More formally, define I_i as the interval $[T^{(i)}, T^{(i-1)})$ inside which the function $TC(T)$ has an associated constant vector $\mathbf{k}^{(i-1)}$, with its elements given by (4). Now observe that for $T \in [T^{(i)}, T^{(i-1)})$ the arguments $-\frac{1}{2}+\frac{1}{2}\sqrt{1+\frac{8b_j}{T^2}}$ in (4) increase as $T \to T^{(i)}$. The vector $\mathbf{k}^{(i-1)}$ will change when one (or more) of its elements increases by one unit just below $T^{(i)}$. Therefore, $T^{(i)}$ can be calculated from equation (5). The elements of the vector $\mathbf{k}$ just below $T^{(i)}$, say $\mathbf{k}^{(i)}$, are given by:

$$k_j^{(i)} = \begin{cases} k_j^{(i-1)} + 1 & for \quad j \in J^{(i)} \\ k_j^{(i-1)} & for \quad j \notin J^{(i)} \end{cases}$$

where $J^{(i)}$ is the set of all elements of $\mathbf{k}$ for which the maximum in (5) is attained.

References

[1] F.M. Andres, H. Emmons, On the optimal packaging frequency of products jointly replenished, Management Science 22 (1976) 1165-1166.

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

[3] S.K. Goyal, Determination of optimum packaging frequency of items jointly replenished, Management Science 21 (1974) 436-443.

[4] E. Porras, R. Dekker, An efficient optimal solution method for the joint replenishment problem with minimum order quantities, in: Report Series Econometric Institute, Erasmus University Rotterdam, EI 2003-52.

[5] S. Viswanathan, A new optimal algorithm for the Joint Replenishment Problem, Journal of the Operational Research Society 47 (1996) 936-944.

[6] S. Viswanathan, On optimal algorithms for the Joint Replenishment Problem, Journal of the Operational Research Society 53 (2002) 1286-1290.

[7] R.E. Wildeman, J.B.G. Frenk, R. Dekker, An efficient optimal solution method for the joint replenishment problem, European Journal of Operational Research 99 (1997) 433-444.