|  |  |
|---|---|
| Publié par :<br>Published by:<br>Publicación de la: | Faculté des sciences de l'administration<br>Université Laval<br>Québec (Québec) Canada  G1K 7P4<br>Tél. Ph. Tel. : (418) 656-3644<br>Télec. Fax : (418) 656-7047 |
| Édition électronique :<br>Electronic publishing:<br>Edición electrónica: | Aline Guimont<br>Vice-décanat - Recherche et affaires académiques<br>Faculté des sciences de l'administration |
| Disponible sur Internet :<br>Available on Internet<br>Disponible por Internet : | **http://www5.fsa.ulaval.ca/sgc/documentsdetravail**<br>rd@fsa.ulaval.ca |

**DOCUMENT DE TRAVAIL 2007-003**

# OFFSETTING INVENTORY REPLENISHMENT CYCLES TO MINIMIZE STORAGE SPACE

**Fayez F. BOCTOR**

|  |  |
|---|---|
| Version originale :<br>Original manuscript:<br>Version original: | ISBN – 2-89524-283-6 |
| Série électronique mise à jour :<br>On-line publication updated :<br>Seria electrónica, puesta al dia | 01-2007 |

# Offsetting inventory replenishment cycles to minimize storage space

**Fayez F. Boctor**
*Université Laval, Québec, Canada*

Abstract

In a recent paper, Murthy, Benton and Rubin (2003) discussed the problem of offsetting inventory replenishment cycles of several items in order to minimize the maximum required storage space. They analyzed the case where replenishment cycles are given integer multiples of a basic period and proposed a heuristic to solve the problem. While they provided a good analysis of the considered problem, the proposed heuristic produces less interesting results. In the following, a simpler, more efficient and easier to implant heuristic is proposed. Numerical results are provided to prove its superiority.

## INTRODUCTION

The problem addressed in this article, also called the replenishment staggering problem, is that of offsetting the replenishment cycles of $N$ different items that share the same storage space (or any other commonly used resource) in order to minimize $S$, the maximum required space. Each item $i$ is replenished in given cycles of length $k_i$ which is an integer multiple of a known basic period that will be considered as the time unit. The demand rate of item $i$, denoted $d_i$, is known and constant. As no backlogs are allowed, the replenishment quantity $Q_i$ is known and equals $k_i d_i$. Under these assumptions, the global replenishment cycle is composed of $K$ basic periods, indexed $t$, where $K$ is the least common multiple of all cycle lengths $k_i$ and the number of replenishments of $i$ within the global cycle, denoted $m_i$, equals $K/k_i$. Two more variables can be used: $x_{if}$ which is a binary that takes the value 1 if item $i$ is replenished at the beginning of period $f$ and $I_{itf}$ which indicates the inventory level of $i$ at period $t$ if it is replenished at periods $f+mk_i$ ;$m$= 0, 1, …, $m_i$-1. Without loss of generality, it is assumed that each unit of $i$ requires one space unit for its storage. We also assume that if the first replenishment of item $i$ is scheduled to occur at period $f_i$, then we should manage to have enough initial inventory to cover the demand up to the beginning of $f_i$.

Very few published research work addressed this problem. Gallego, Shaw and Simchi-Levi (1992) showed that the problem is $NP$-hard even if only one cycle multiple is different from the others. Hariga and Jackson (1995) proposed to solve the problem by varying lot sizes through time while no backlogs are allowed. Hall (1998) examined the problem in the case where all cycle lengths are equal, showed that the problem is *NP*-hard, compared two solution heuristics and provided the worst case ratios for the considered heuristics. More recently, Murthy, Benton and Rubin (2003) proposed a heuristic to solve the staggering problem under the assumption made in this note. However, this heuristic seems to provide solutions of average quality. This note provides a simpler, more efficient and easy to implement heuristic. It will be shown, based on a number of randomly generated instances, that the proposed heuristic outperforms the one by Murthy et al. for all tested problems.

Some related problems are studied in the open literature. Several authors proposed methods to minimize the sum of order and inventory holding costs while trying to stagger replenishment in order to satisfy space availability or other resource constraints. Zoller (1977) and Rosenblatt and Rothblum (1990) considered this problem under the assumption that all items have the same cycle length. Hartley and Thomas (1982) and Thomas and Hartley (1983) considered the two item case.

Other researchers did not consider the staggering aspect of the problem and proposed to minimize the sum of order and inventory holding cost without exceeding the available storage space (Page and Paul 1976, Goyal 1978, Anily 1991, Gallego, Queyranne and Simchi-Levi 1996). Teo, Ou and Tan (1998) proposed to minimize the sum of order, inventory holding and space utilization costs.

## MATHEMATICAL FORMULATION

The problem considered in this note can be formulated as follows:

Determine: $x_{if} \in \{0,1\}$, $i=1, \ldots, N$, $f=1, \ldots, k_i$ which

Minimize: $S$

$$\text{Subject to: (1)} \quad \sum_{f=1}^{k_i} x_{if} = 1 \qquad ; i = 1,\ldots, N$$

$$\text{(2)} \quad \sum_{i=1}^{N} \sum_{f=1}^{k_i} x_{if} I_{itf} \le S \qquad ; t = 1,\ldots, K.$$

Let $s = t \bmod(k_i)$, then $I_{itf}$ can be calculated by:

$$I_{itf} = \left\{ \begin{array}{ll} Q_i - (t-f)d_i & , f \le t < f + k_i \\ I_{isf} & , \text{for all other values of } t \end{array} \right\}.$$

Unlike previously reported formulations, this one can be directly solved by commercial integer programming packages. Within this research work CPLEX was used to solve it. However, as this model contains $\sum_{i=1}^{N} k_i$ binary variables and $K+N$ constraints, it is obvious that it cannot be solved but for small problem instances.

## THE PROPOSED HEURISTIC

The proposed heuristic is composed of two main parts: a solution construction procedure to construct an initial solution and a solution improvement procedure to improve the obtained solution. The following additional notations will be used to present the proposed heuristic:

| | |
|---|---|
| $I_k$ | the replenishment schedule after the $k^{\text{th}}$ iteration of the construction procedure |
| $S_{ikf}$ | maximum storage space requirement if item $i$ is added to $I_k$ and its first replenishment is scheduled at period $f$ |

To construct an initial solution, the proposed heuristic proceeds as follows:

***Initialisation:*** - Order the set of items in the ascending order of their lot sizes $Q_i$,
- Schedule the first item, denoted $u$, in the obtained list to be replenished at periods $mk_u$+1 where $m$ = 0, 1,…, $m_u$-1.

***Iteration k:*** - Consider the next item in the list, denoted $i$,
- Schedule its replenishments at $f_i + mk_i$ where $f_i = \arg\min (S_{ikf})$ and $m$=0, 1, …, $m_i$-1.

To improve the obtained solution, the proposed heuristic considers items one by one and eventually moves its replenishments to the periods that lead to the maximum storage space reduction. The procedure stops if no further improvement can be achieved.

## A SIMULATED ANNEALING ALGORITHM

An adaptation of the simulated annealing (SA) algorithm was developed to solve the considered replenishment staggering problem. The solutions obtained by the above proposed heuristic will be compared to those obtained by the SA adaptation in order to provide a more complete assessment of its performance.

The initial solution used by the SA adaptation is obtained by the construction procedure given above. Neighbour solutions are obtained by randomly modifying the replenishment pattern of a randomly selected item and, as usual, the neighbour solution is accepted if it provides a reduction of the maximum required storage space or if a randomly drawn value is less than the standard Boltzman value. Figure 1 exhibit the main steps of the developed SA adaptation.

```
Call initial (find an initial solution)
Store as current solution
$C$:=0  (initialize stopping counter)
Repeat until $C$=$CMAX$
    $C$:=$C$+1
    $T$=$T_0$
    $r$:=0  (initialize repetition counter)
    Repeat until $r$=$RMAX$
        Call neighbour  (generate a neighbour solution)
        $r$:=$r$+1
        $d$:= max storage space(neighbour) – max storage space(current)
        If $d$<0  or  random(0,1)≤EXP(-$d$/$T$)  do
            Store the neighbour solution as the current one
            If max storage space(current) < max storage space(best)  do
                Store current solution as the best solution
            End if
        End if
        $T$:=$\alpha$*$T$  (reduce cooling temperature)
    End repeat
End repeat
```

**Figure 1:** Pseudo code for the developed SA adaptation

In the numerical tests presented in the last section of this note, the following parameter values were used: $T_0$= 16, *CMAX*= 5, *RMAX*= 1000 and $\alpha$= 0.5.

## NUMERICAL EXAMPLE

Murthy, Benton and Rubin (2003) solved the numerical example of Table 1 and obtained a maximum space requirement of 875 storage units. This same example was solved by the heuristic proposed in this note giving an initial solution requiring 866 storage space units. The solution obtained after applying the improvement procedure requires 861 storage space units. This solution is the optimal solution obtained by solving the corresponding mathematical model.

**Table 1:** Numerical example and obtained solutions

| Item $i$ | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Maximum storage space |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Order size $Q_i$ | | 100 | 200 | 81 | 144 | 150 | 160 | 90 | 60 | 50 | |
| Cycle length $k_i$ | | 4 | 5 | 9 | 12 | 15 | 8 | 6 | 12 | 2 | |
| Period of first replenishment $f_i$ | Murthy, Benton and Rubin's solution | 1 | 5 | 1 | 1 | 1 | 7 | 6 | 5 | 2 | 875 |
| | Proposed heuristic: Initial solution | 1 | 1 | 2 | 6 | 1 | 1 | 5 | 8 | 2 | 866 |
| | Proposed heuristic: improved solution | 1 | 1 | 2 | 6 | 1 | 1 | 5 | 7 | 2 | 861 |

## PERFORMANCE EVALUATION

To assess the performance of the proposed heuristic, 30 test instances were randomly generated. Each problem has 20 items to replenish with demand rates drawn from a uniform distribution between 5 and 30 units and cycles times drawn from a uniform distribution between 2 and 12. Each instance was solved by the heuristic proposed by Murthy, Benton and Rubin, called hereafter the MBR heuristic, by the heuristic proposed in this note and by the developed adaptation of the simulated annealing algorithm. The obtained results are reported in Table 2.

It shows that the suggested construction procedure always outperformed the MBR heuristic. This procedure produced solutions 11.19% better while the whole heuristic produced solutions 11.56% better. Note that the average computation time for the proposed heuristic is only 0.2 seconds. The simulated annealing adaptation produced slightly better solutions but required much larger computation times. The average percentage deviation of the SA adaptation with respect to the MBR heuristic is 11.92% and its average computation time is 20.5 seconds. With respect to the proposed heuristic, the SA adaptation produced an average improvement of 0.41% but multiplies its computation time by 102.5 in average. The SA adaptation produced better solutions for 14 of the 30 instances and worst solutions for 6 instances.

**Table 2:** Results for the 30 test instances

| Instance | MBR Heuristic | Proposed heuristic | | | | Simulated annealing | |
|---|---|---|---|---|---|---|---|
| | | Initial solution | Percentage[1] improvement | Improved solution | Percentage[1] improvement | Solution | Percentage[1] improvement |
| 1 | 1688 | 1567 | 7,17% | **1551** | 8,12% | 1567 | 7,17% |
| 2 | 1634 | **1444** | 11,63% | **1444** | 11,63% | **1444** | 11,63% |
| 3 | 1250 | 1157 | 7,44% | 1157 | 7,44% | **1148** | 8,16% |
| 4 | 1458 | **1268** | 13,03% | **1268** | 13,03% | **1268** | 13,03% |
| 5 | 1377 | 1305 | 5,23% | **1295** | 5,95% | **1295** | 5,95% |
| 6 | 1551 | **1443** | 6,96% | **1443** | 6,96% | **1443** | 6,96% |
| 7 | 1767 | 1595 | 9,73% | 1576 | 10,81% | **1555** | 12,00% |
| 8 | 1295 | 1136 | 12,28% | **1127** | 12,97% | 1136 | 12,28% |
| 9 | 1637 | 1488 | 9,10% | 1488 | 9,10% | **1478** | 9,71% |
| 10 | 1172 | 1067 | 8,96% | 1067 | 8,96% | **1066** | 9,04% |
| 11 | 1549 | 1387 | 10,46% | 1387 | 10,46% | **1386** | 10,52% |
| 12 | 2000 | **1658** | 17,10% | **1658** | 17,10% | **1658** | 17,10% |
| 13 | 1551 | 1317 | 15,09% | 1293 | 16,63% | **1285** | 17,15% |
| 14 | 1554 | 1356 | 12,74% | 1355 | 12,81% | **1314** | 15,44% |
| 15 | 2044 | 1695 | 17,07% | **1684** | 17,61% | 1687 | 17,47% |
| 16 | 1668 | 1416 | 15,11% | 1416 | 15,11% | **1402** | 15,95% |
| 17 | 1789 | 1608 | 10,12% | 1608 | 10,12% | **1565** | 12,52% |
| 18 | 2028 | **1782** | 12,13% | **1782** | 12,13% | **1782** | 12,13% |
| 19 | 1804 | 1631 | 9,59% | 1630 | 9,65% | **1580** | 12,42% |
| 20 | 1704 | 1553 | 8,86% | **1528** | 10,33% | 1544 | 9,39% |
| 21 | 1750 | 1558 | 10,97% | 1549 | 11,49% | **1541** | 11,94% |
| 22 | 1377 | **1268** | 7,92% | **1268** | 7,92% | **1268** | 7,92% |
| 23 | 1254 | 1131 | 9,81% | 1131 | 9,81% | **1121** | 10,61% |
| 24 | 1521 | **1358** | 10,72% | **1358** | 10,72% | **1358** | 10,72% |
| 25 | 1938 | 1679 | 13,36% | **1653** | 14,71% | 1664 | 14,14% |
| 26 | 1434 | **1263** | 11,92% | **1263** | 11,92% | **1263** | 11,92% |
| 27 | 2463 | 2088 | 15,23% | 2088 | 15,23% | **2062** | 16,28% |
| 28 | 1412 | 1289 | 8,71% | 1278 | 9,49% | **1274** | 9,77% |
| 29 | 1321 | **1117** | 15,44% | **1117** | 15,44% | **1117** | 15,44% |
| 30 | 1668 | 1473 | 11,69% | **1449** | 13,13% | 1453 | 12,89% |
| Average | | | 11.19% | | 11.56% | | 11.92% |
| Standard deviation | | | 3.01% | | 3.04% | | 3.14% |
| Minimum | | | 5.23% | | 5.95% | | 5.95% |
| Maximum | | | 17.10% | | 17.61% | | 17.47 |
| Sec.[2] | 0.03 | | 0.07 | | 0.20[3] | | 20.50[3] |

[1] With respect to the MBR solution.
[2] Average computation time in seconds using a centurion microprocessor paced at 1.86 MHz.
[3] Including the time to obtain the initial solution.

## REFERENCES

Anily, S., 1991, Multi-item replenishment and storage problem (MIRSP): heuristics and bounds, *Operations Research*, 39, 233-243.

Gallego, G., D. Shaw, and D. Simchi-Levi, 1992.The complexity of the staggering problem and other classical inventory problems, *Operations Research Letters* 12, 47-52.

Gallego, G., M. Queyranne, and D. Simchi-Levi, 1996, Single resource multi-item inventory systems, *Operations Research* 44, 580-595.

Goyal, S.K., 1978, A note on 'Multi-production inventory situation with one restriction', *Journal of Operational Research Society* 29, 269-271.

Hall, N. G., 1998, A comparison of inventory replenishment heuristics for minimizing maximum storage, American *Journal of Mathematical and Management Sciences* 18, 245-258.

Hariga, M.A. and P.L. Jackson, 1995, Time variant lot sizing models for the warehouse scheduling problem, *IIE Transaction* 27, 162-170.

Hariga, M.A. and P.L. Jackson, 1996, The warehouse scheduling problem: formulation and algorithm, *IIE Transaction* 28, 115-127.

Murthy, N. N., W. C. Benton and P. A. Rubin, 2003, Offsetting inventory cycles of items sharing storage, *European Journal of Operational Research* 150, 304-319.

Page, E. and R.J. Paul, 1976, Multi-production inventory situation with one restriction, *Journal of Operational Research Society* 27, 815-834.

Rosenblatt; M.J. and U.G. Rothblum, 1990, On the Single Resource Capacity Problem for Multi-Item Inventory Systems, *Operations Research*, 38, 686-693.

Teo, C.P., J. Ou and K. Tan, 1998, Multi-Item Inventory Staggering Problems: Heuristics and Bounds, Proceedings of the ninth annual ACM-SIAM symposium on discrete algorithms, 1998, 584-593.

Zoller, K., 1977, Deterministic multi-item inventory systems with limited capacity, *Management Science* 24, 451-455.