CIRRELT
Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport
Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

The Inventory Replenishment Planning and Staggering Problem Revisited

Fayez F. Boctor
Marie-Claude Bolduc

May 2012

CIRRELT-2012-19

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2012-002.

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

The Inventory Replenishment Planning and Staggering Problem Revisited

Fayez F. Boctor*, Marie-Claude Bolduc

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, 2325, de la Terrasse, Université Laval, Québec, Canada G1V 0A6

Abstract. This paper reconsiders the inventory replenishment problem and emphasises the fact that it is a multi-objective problem where, in addition to minimizing the sum of order and inventory holding costs, we should optimize the usage of storage resources. The paper proposes a mathematical formulation of the problem, suggests two heuristic solution approaches, and assesses their performance.

Keywords. Inventory replenishment planning and staggering, lot sizing, heuristics.

Acknowledgements. This research work was partially supported by grants OPG0036509 from the Natural Sciences and Engineering Research Council of Canada (NSERC), and by a grant from Université Laval. This support is gratefully acknowledged.

---

* Corresponding author: Fayez.Boctor@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2012

1- INTRODUCTION

This paper deals with inventory replenishment of multiple items supplied by external suppliers to fulfil constant demands. It is assumed that these items share a limited storage space and/or other storage resources like manpower and handling equipments. Two cost elements are considered: ordering costs and inventory holding costs. Items order costs are assumed known, constant and independent of each other. Also, per unit inventory holding costs are assumed known and constant. In practice it is required that replenishment be cyclic and that cycles be integer multiples of a basic time period also called fundamental cycle. This basic period is also required to be an integer number of time units.

The problem is to determine the time length of the fundamental cycle, the multiples that determine replenishment cycles of items, and items reception (replenishment) periods. The objective is to minimize the sum of order and inventory costs while minimizing the maximum storage space needed and/or other storage resource requirements. This means that we are dealing with a multi-objective problem where all objectives are to be minimized.

It is a common practice to solve this problem by first determining the optimal or near optimal cycle times without neither determining replenishment periods nor taking into consideration the storage space requirements. Then, using the obtained cycle times, determine items replenishment periods that minimizes the maximum required storage space. Handling these two sub-problems separately may lead to less good solutions. Sometimes little modifications of cycle times may allow reducing the maximum required storage space significantly and allows using smaller storage facility. Thus we may be able to avoid important facility expansion investments.

To illustrate this point let us consider the 3-item replenishment problem shown in table 1. If we do not take into consideration storage space requirements, optimal cycle times for these items are respectively 3, 4 and 7 and the corresponding total cost (order cost and inventory holding cost) is 120. Given these cycle times, the maximum required storage space is the same whatever the chosen replenishment periods. This maximum storage

space is 300 space units (Figure 1.a). Now if we modify cycle times to become 3, 3 and 6, the total cost increases to 122.5 which is just 2.1% increase. In the same time the maximum required storage space can be reduced, as shown in Figure 1.b, to 233.3 space units by replenishing these items in periods 1, 2 and 3 respectively. In addition to using less storage space, we now need less manpower and less handling equipments for our warehouse operations. Often this largely recompense for the increase of inventory and order costs.

Item	Demand Unit per time unit	Order cost	Inventory holding cost per time unit	Storage space per unit	Optimal replenishment cycle	Optimal cost per time unit	Modified cycle time	Total cost
1	100	45	0.10	1	3	30	3	30
2	100	96	0.12	1	4	48	3	50
3	100	147	0.06	1	7	42	6	42.5
					Total	120		122.5

Table 1: A three items example

For large ware houses operations, it is recognized now that solving the cycle-times determination problem separately from the replenishment staggering problem is not optimal. Rather we need to consider the minimization of total cost and storage space together; meaning that we have to handle our replenishment problem as a multi-objective problem. Defining a problem as a multi-objective one implies that we are not willing to decide a priori which objective is more important or what the weight to assign to each objective is? Thus, instead of providing the decision maker with a unique solution, we have to construct a series of non-dominated (Pareto or near-Pareto optimum) ones leaving the final choice to him or her.

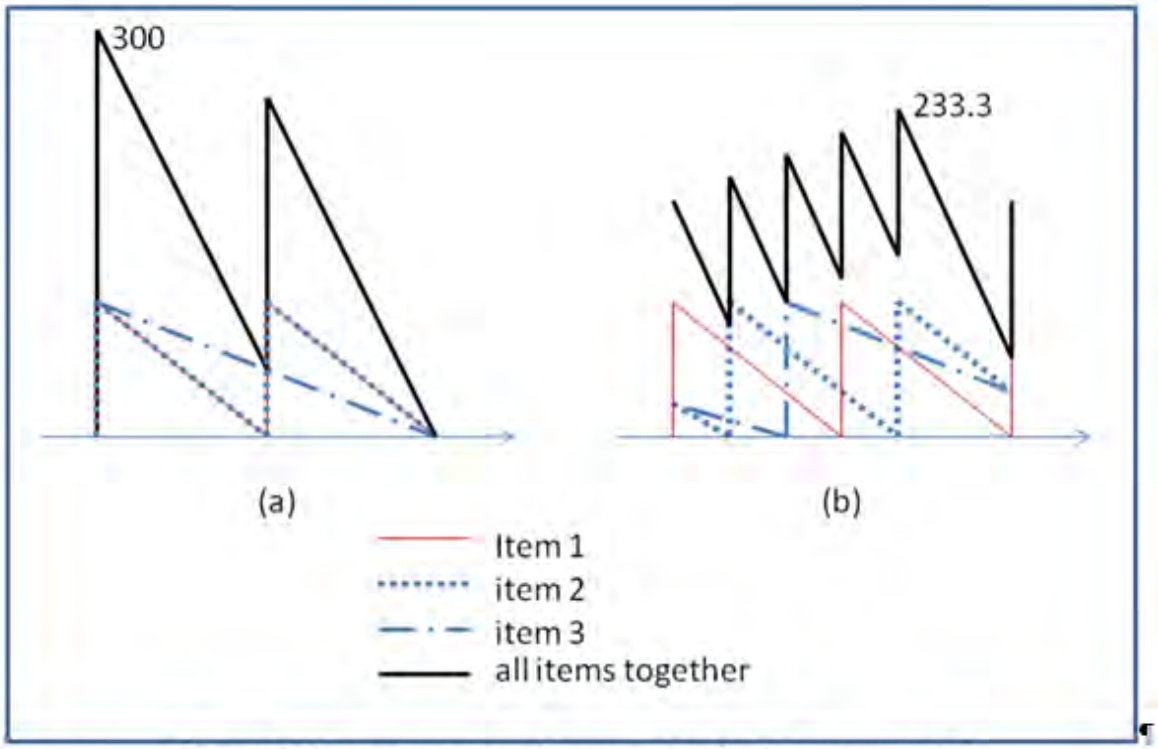

Figure 1: Storage space requirements for the 3 items example

2- PROBLEM DEFINITION AND MATHEMATICAL FORMULATION

The problem addressed in this article, called the inventory replenishment and staggering problem, is that of determining the replenishment cycles of N different items that share the same storage space (and/or any other storage resource) as well as staggering their reception periods in order to minimize: (1) the total cost per period, denoted C, and (2) the maximum required storage space, denoted S.

Each item i is replenished in cycles of length T_i which is an integer multiple m_i of a basic period, also called fundamental cycle, denoted b. The demand rate of item i, denoted d_i, is known and constant and as no backlogging is allowed, the replenishment quantity Q_i is known and equals $m_i b d_i$. For each item i, both the unit inventory holding cost per unit of time, denoted h_i, and the order cost, denoted O_i are known and constant. Under these assumptions, the global replenishment cycle is composed of M basic periods, indexed t, where M is the least common multiple (lcm) of all cycle multipliers m_i, and the number of replenishments of i within the global cycle, denoted n_i, equals M/m_i. Two more variables

can be used: x_{it} which is a binary that takes the value 1 if item i is replenished at the beginning of period t and I_{it} which indicates the inventory level of i at the beginning of period t. Without loss of generality, it is assumed that each unit of i requires one space unit for its storage; or in other words, the measuring unit of i is the quantity that requires one unit of storage space. We also assume that if the first replenishment of item i is scheduled to occur at a given period, we should manage to have enough initial inventory to cover the demand up to the beginning of this period. In the following only two objectives are considered: (1) to minimize C, the total cost, and (2) to minimize S, the maximum storage space required.

Using the above given notation, the problem can be formulated as follows:

Model 1

Find: b *integer* ≥ 0, m_i integer ≥ 0, $x_{it} \in \{0,1\}$ and $I_{it} \geq 0$; $i=1, \ldots, N$, $t=1, \ldots, M$ which:

$$\text{Minimize: } C = \sum_{i=1}^{N} \left\{ \frac{O_i}{m_i b} + \frac{h_i d_i m_i b}{2} \right\} \tag{1}$$

$$\text{Minimize: } S \tag{2}$$

$$\text{Subject to: } \sum_{t=1}^{m_i} x_{it} = 1 \qquad ;i = 1,..., N \tag{3}$$

$$\sum_{i=1}^{N} I_{is_{it}} \leq S \qquad ;t = 1,..., M \tag{4}$$

$$I_{i,s_{it}} = I_{i,s_{it}-1} - d_i + m_i b d_i x_{is_{it}} \qquad ;i = 1,..., N; t = 1,..., M \tag{5}$$

$$\text{where} \quad s_{it} = t \bmod (m_i) \qquad ;i = 1,..., N; t = 1,..., M. \tag{6}$$

Notice that the optimal solution is such that b and m_i; $i=1, \ldots, N$, are strictly larger than zero because of the hyperbolic term of the cost function C.

The first objective function gives the sum of ordering and inventory holding costs per time unit. The first set of constraints assures that each item is replenished once and only once during its first replenishment cycle. The second set determines the maximum storage space required while the third set determines inventory levels at the beginning of each period and assures that replenishment is cyclic. This is a non-linear multi-objective

program where the integers m_i are not only direct decision variables but also determine the limits of a sum on other variables.

Few authors proposed methods to deal with some special cases of this problem. 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 costs under the constraint that the sum of space required by the sum of all ordered quantities is less than or equal to a given limit denoted L (Page and Paul 1976, Goyal 1978, Anily 1991, Gallego, Queyranne and Simchi-Levi 1996). This problem can be modeled as follows:

Find: b *integer* ≥ 0, m_i integer ≥ 0; $i=1,\ldots, N$, which:

$$\text{Minimize: } C = \sum_{i=1}^{N} \left\{ \frac{O_i}{m_i b} + \frac{h_i d_i m_i b}{2} \right\} \tag{1}$$

$$\text{Subject to : } \sum_{i=1}^{N} m_i d_i b \leq L \tag{7}$$

This model can be solved by: (1) generating all feasible vectors m_i; $i=1,\ldots, N$ in lexicographic order, (2) for each vector determine the corresponding optimal value of b and C, and (3) retain the solution producing the minimum cost.

As the minimal value of b and $m_i, \forall i$ is 1, a necessary feasibility condition for model 5 is:

$$\sum_{i=1}^{N} d_i \leq L. \tag{8}$$

The upper bound on m_i is reached when b and all $m_j, \forall j \neq i$ take the value one. Thus the limits on the value of m_i are:

$$1 \leq m_i \leq \frac{1}{d_i}(L - \sum_{j \neq i} d_j). \tag{9}$$

For a given vector m_i; $i=1,\ldots, N$, The optimal corresponding value of b can be determined by:

$$b^* = \min\left(b_2 , \left\lfloor \frac{L}{\sum_{i=1}^{N} m_i d_i} \right\rfloor \right). \quad (10)$$

Where:

$$b_2 = \begin{cases} b_1 & \text{if } C(b_1) \le C(b_1+1) \\ b_1+1 & \text{otherwise} \end{cases}; \; b_1 = \left\lfloor \sqrt{\frac{2\sum_{i=1}^{N} \frac{O_i}{m_i}}{\sum_{i=1}^{N} h_i d_i m_i}} \right\rfloor \text{ and } C(b) = \frac{1}{b}\sum_{i=1}^{N} \frac{O_i}{m_i} + b\sum_{i=1}^{N} h_i d_i m_i \quad (11)$$

Still we don't have any efficient method to solve the general case modeled by model 1 above. Thus we need to develop some heuristic methods to produce good but not necessarily optimal solutions. In the following, two heuristics solution approaches are proposed.

3- TWO HEURISTIC SOLUTION APPROACHES

In the following, two solution approaches are suggested to deal with the considered replenishment planning and staggering problem. The first approach, called hereafter the *exploratory method* (EM), tries different vectors of cycle multipliers m_i, i=1, …, N and always fix the fundamental cycle b =1. For each vector, we determine the corresponding total cost C and solve the corresponding replenishment staggering problem where replenishment (reception) periods within the obtained cycles are determined while trying to minimize the maximum storage space required. This approach may provide as many solutions as the number of cycle multipliers vectors we are willing to consider. Dominated solutions should then be discarded.

The second approach is an evolutionary algorithm; called hereafter the Two-Population Evolutionary Algorithm (TPEA), produces a series of solutions, identify the non-dominated ones and leaves the final choice again to the decision maker. These two approaches are discussed and assessed in the following.

3.1- The exploratory method

The fundamental cycle value for all the solutions produced by this method is one (b=1). The method can be summarized as follows: we generate a number of cycle multipliers vectors, for each vector calculate the corresponding total cost C and solve the corresponding replenishment staggering problem to minimize S. Dominated solutions are then discarded.

Although b equals 1 for all the generated solutions, to generate the required cycle multipliers vectors, we use several different values of b. Once the vector is generated we put back b=1. Precisely we use all the values of b between two chosen limits b_l and b_u with a step of Δb and for each value of b we generate a vector of cycle multipliers m_i; i=1,…, N. This vector is the one that minimizes the total cost without taking in consideration storage space required; i.e., the one that solves the following model:

Find: m_i integer ≥ 0; i=1,…, N which:

$$\text{Minimize: } C = \sum_{i=1}^{N} \left\{ \frac{O_i}{m_i b} + \frac{h_i d_i m_i b}{2} \right\}. \tag{12}$$

This function is separable into N functions and the problem becomes to solve for each of the N items the following problem:

Find: m_i integer ≥ 0 which:

$$\text{Minimize: } C_i(m_i) = \frac{O_i}{m_i b} + \frac{h_i d_i m_i b}{2} \tag{13}$$

It is easy to see that the optimal solution of this problem is:

$$m_i^* = \begin{cases} \mu_i & \text{if } C_i(\mu_i) \leq C_i(\mu_i + 1) \\ \mu_{i+1} & \text{otherwise} \end{cases} \quad ; \text{where } \mu_i = \left\lfloor \frac{1}{b} \sqrt{\frac{2O_i}{h_i d_i}} \right\rfloor \tag{14}$$

For each obtained vector of m_i; i=1,…, N, we calculate the total cost C for the corresponding multipliers with b=1. This can be done by substituting these values in (12).

Afterwards, we move to solve the reception or replenishment staggering problem; *i.e.*, to determine the reception period for each item. This can be done either by using a heuristic (such a heuristic will be presented later in this section) or by solving the following model.

Model 2

Using b=1 and the obtained values of m_i; $\forall i$; find $x_{it} \in \{0,1\}$ and $I_{it} \geq 0$; i=1, .., N, t=1, .., m_i which:

$$\text{Minimize: } S \tag{2}$$

$$\text{Subject to : } \sum_{t=1}^{m_i} x_{it} = 1 \qquad ;i = 1,...,N \tag{3}$$

$$\sum_{i=1}^{N} I_{is_{it}} \leq S \qquad ;t = 1,...,M \tag{4}$$

$$I_{i,s_{it}} = I_{i,s_{it}-1} - d_i + m_i b d_i x_{is_{it}} \qquad ;i = 1,...,N; t = 1,...,M \tag{5}$$

$$\text{where} \quad s_{it} = t \bmod (m_i) \qquad ;i = 1,...,N; t = 1,...,M. \tag{6}$$

This model is a mixed integer model with binary and continuous variables. A more compact, but equivalent, model with only binary variables is:

Model 3

Using b=1 and the obtained values of m_i; i=1, ..., N; find $x_{it} \in \{0,1\}$; i=1, ..., N, t=1, ..., m_i which:

$$\text{Minimize: } S \tag{2}$$

$$\text{Subject to : } \sum_{t=1}^{m_i} x_{it} = 1 \qquad ;i = 1,...,N \tag{3}$$

$$\sum_{i=1}^{N} \sum_{s=1}^{m_i} x_{is} I_{ist} \leq S \qquad ;t = 1,...,M. \tag{15}$$

Where I_{ist} is the inventory level of item i at the beginning of period t if its order is received at the beginning of period s of its replenishment cycle. Formally:

$$I_{ist} = \begin{Bmatrix} Q_i - (t-s)d_i & \text{, for } s \leq t < s + m_i \\ I_{is,t \bmod(m_i)} & \text{, for all other values of } t \end{Bmatrix} \tag{16}$$

Very few published research work addressed this replenishment staggering problem for a given cycle times (see Murthy, Benton and Rubin, 2003 and Boctor 2010). 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. Teo, Ou and Tan (1998) proposed an upper bound and a heuristic for the case where each cycle time is a divisor for another one. More recently, Murthy, Benton and Rubin (2003) proposed a heuristic to solve this staggering problem. However, this heuristic seems to provide solutions of average quality. Boctor (2010) proposed a number of other heuristic methods and shown, based on a number of randomly generated instances, that the proposed heuristics outperform the one by Murthy *et al.* for all tested instances.

Among the proposed heuristics, the following one will be used in this paper to produce an approximate solution to the staggering problem.

Heuristic *H*1

Initialisation: - Order the set of items in the ascending order of their lot sizes Q_i,
- Schedule the first item in the list, denoted u, to be replenished at periods nm_u+1 where $n = 0, 1,\ldots, n_u-1$.

Iteration: - Consider the next item in the list, denoted i,
- Schedule its replenishments at $f_i + nm_i$ (n=0, 1,…, n_i-1) where f_i is the replenishment period leading to the smallest maximum space required for items 1, …,i.

Improvement: - considers items one by one and eventually moves its replenishments to the periods that lead to the maximum reduction of the maximum required storage space. The procedure stops if no further improvement can be achieved.

Numerical illustration

Let us consider the ten-item example presented in Table 2 and use the proposed exploratory approach to solve it. Using the parameters b_l=0.9, b_u=3.9 and Δb =0.03333 to generate a set of different multiplier vectors and the Heuristic *H*1 to determine *S*, the maximum storage space required. We obtain the 17 non dominated solutions given by the

second and third rows of Table 3. To get a rough estimation of by how much we can reduce the maximum storage space required if we used Model 3 instead of heuristic *H*1, we solved the model to determine *S* for each of the 17 non-dominated solutions. The obtained values of *S* are given by the fourth row of the same table. These results (the empirical Pareto front) are also depicted on Figure 2.

item	Demand per period	Cost per Order	Unit inventory holding cost per period
1	100	50	0,1
2	100	90	0,12
3	100	120	0,06
4	120	40	0,08
5	120	100	0,1
6	150	160	0,08
7	150	100	0,05
8	200	150	0,1
9	200	200	0,12
10	200	240	0,05

Table 2: Data for the ten-item example

Solution	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
C the total cost per period	538.4	538.6	539.2	540.2	547.4	554.4	557.4	559.3	565.6	584.5	640.5	647.5	662.5	696.1	783.3	887.3	1019.3
S as obtained by *H*1	5320	5100	4640	4570	4310	4060	3860	3840	3830	3230	3010	2910	2610	2410	2090	1910	1790
S as obtained by Model 3	5140	5100	4520	4480	4100	3940	3840	3740	3700	3080	2890	2810	2510	2410	2060	1910	1740
Deviation of *H*1 from Model 3	2.9%	0.0%	1.5%	0.0%	4.0%	3.6%	4.2%	4.9%	3.5%	2.7%	0.5%	3.1%	5.1%	2.0%	2.7%	0.0%	3.5%

Table 3: Non dominated solutions obtained by the exploratory method

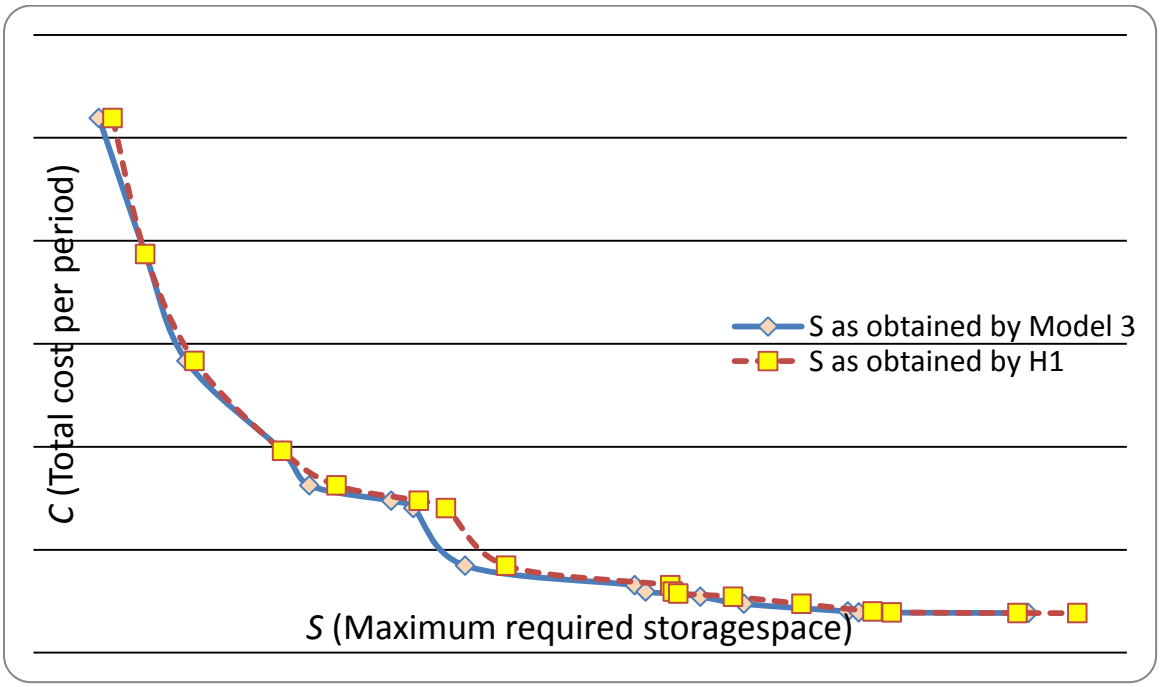

Figure 2: Non dominated (Pareto) solutions as obtained by the Exploratory Method (EM)

Obviously using Model 3 requires more computational time and gives smaller values of S than those obtained if we use heuristic $H1$; the average improvement of S is 2.58%. Indeed, the additional computational time for using Model 3 for such a small instance is quite small. However, including this model in the exploratory approach in order to determine S for all enumerated solutions (dominated and non-dominated) increases the computational time to about 640 times the computational time of the approach if we use the heuristic $H1$. For larger problems the ratio is expected to be much higher than 640.

3.2- The proposed evolutionary algorithm

The second approach we propose to solve the considered inventory replenishment problem is an evolutionary algorithm. The literature indicates that evolutionary algorithms are well suited for solving multi-objective optimization problems as they are able to produce several solutions from which we can extract a set of non-dominated (Pareto near-optimal) solutions.

Many multi-objective evolutionary algorithms (MOEA) were proposed since the mid-eighties (see Van Veldhuizen et al 2000, Konak et al 2006). However the question of which one is better or outperforms others is not yet settled. Few comparative studies (see Zitzler et al 2000) attempted to deal with this question but many new MOEA were developed since.

Our objective here is not to develop a more efficient MOEA but a fast one based on a new concept and to show that it is able to handle efficiently our inventory replenishment problem. Designing a MOEA requires dealing with two important issues: (1) how to guide the search towards the Pareto-optimal front by designing the suitable fitness assignment scheme, and (2) which solutions to preserve all along the evolution process in order to converge to a sufficiently good set of non-dominated solutions. To deal with the first issue we suggest a two-population evolutionary algorithm (TPEA) where the fitness of each population is determined based on either the total cost or the maximum storage space. To deal with the second issue we use an external archive where non-dominated solutions are stored and updated. External archives are used by some of the most recently proposed MOEA like NSGA-II (Deb et al 2002) and SPEA2 (Zitzler et al 2001).

In the following we present the elements of the proposed two-population evolutionary algorithm (TPEA).

Solution coding: each solution is represented by its vector of cycle multipliers. The first replenishment period for each item (and the consecutive replenishment periods) is not coded as a part of the solution. Instead for each multipliers vector we apply a heuristic to solve the staggering problem (determine the replenishment periods).

Number and size of populations: we need to use as many populations as the number of objectives to consider. In our case as our objectives are to minimize C and S, we use two populations. The population size, denoted P, is a parameter to be chosen by the user. To accelerate the evolution procedure, in the tested implementation of the algorithm we use populations of 100 solutions each (P=100).

Initial population: We generate $2P$ chromosomes and we use the first P chromosome as the first population and the P as the second population. To generate these $2P$ chromosomes we first generate one chromosome (cycle multipliers vector) by substituting b=1 in equation (14). Then we generate each of the remaining individuals as follows. Randomly select between 4 and 8 genes (items to replenish) in the first vector and randomly modify their cycle multiplier within a given range (example by adding or subtracting a random value between 1 and a predetermined value u). for each generated vector (chromosome) we calculate its total cost per period C and apply a heuristic (we use $H1$) to solve the corresponding staggering problem and to determine the maximum storage space required S.

Mating pool selection: The selection of individuals to participate in production of the next generation is done in the same way for each of the two populations as follows:

1- We arrange the overall set of all individuals in the ascending order of their total cost C (respectively S for the second population),

2- The top half individuals in the resulting list constitute the first population. If a given solution has more than one copy in this population, keep only one copy and remove the other copies. If there is more than a p individuals remaining (in our

implementation p=80% of the population size) keep only the first p individual and remove the others. Add to the population a copy the best 20% (those with the lowest values of C for the first population and respectively S for the second one).

Cross-over operator: the used operator is a random one position cross-over operator.

Stopping rule: when a predetermined number of generations is produced.

Application to the 10-item numerical example

The proposed two-population evolutionary algorithm was applied to the 10-item example presented in table 2. The algorithm is stopped once 30 generations are produced. The results are depicted on Figure 3. We also solved Model 3 for each of the 46 obtained solutions to see by how much the maximum required storage space can be reduced. The solutions obtained are also presented in Figure 3.

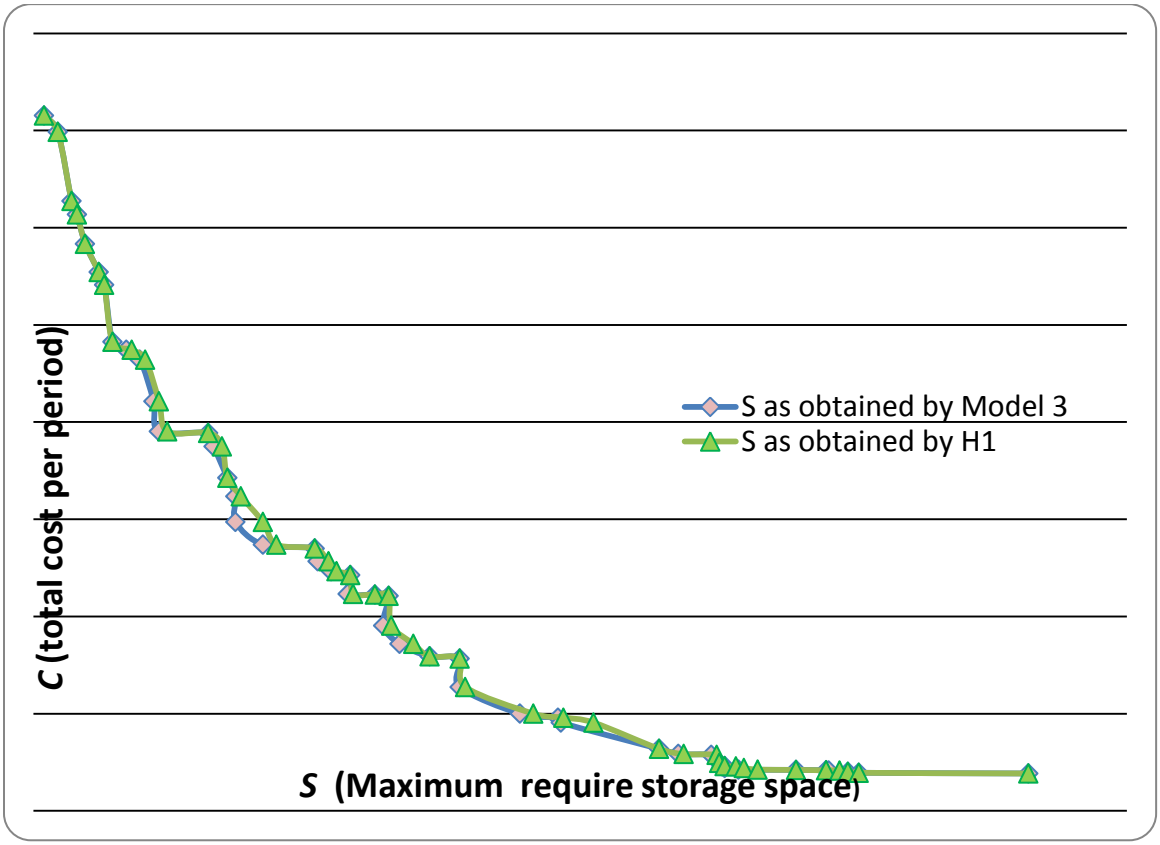

Figure 3: Non dominated (Pareto) solutions as obtained by the Two-Population Evolutionary Algorithm (TPEA)

Figure 4 presents the 46 non-dominated solutions obtained by the genetic algorithm together with the 17 non-dominated solutions of the exploratory method. Among these 63 solutions 32 are non-dominated. Ten (59%) of the 17 solutions of the exploratory methods are not dominated by any of the solutions obtained by the genetic algorithm while 7 solutions are dominated. Also 22 of the 46 solutions of the genetic algorithm are not dominated (48%) while the other 24 are dominated. Thus although the proposed genetic algorithm produce more solutions, a higher percentage of these solutions are dominated by those produced by the exploratory method.

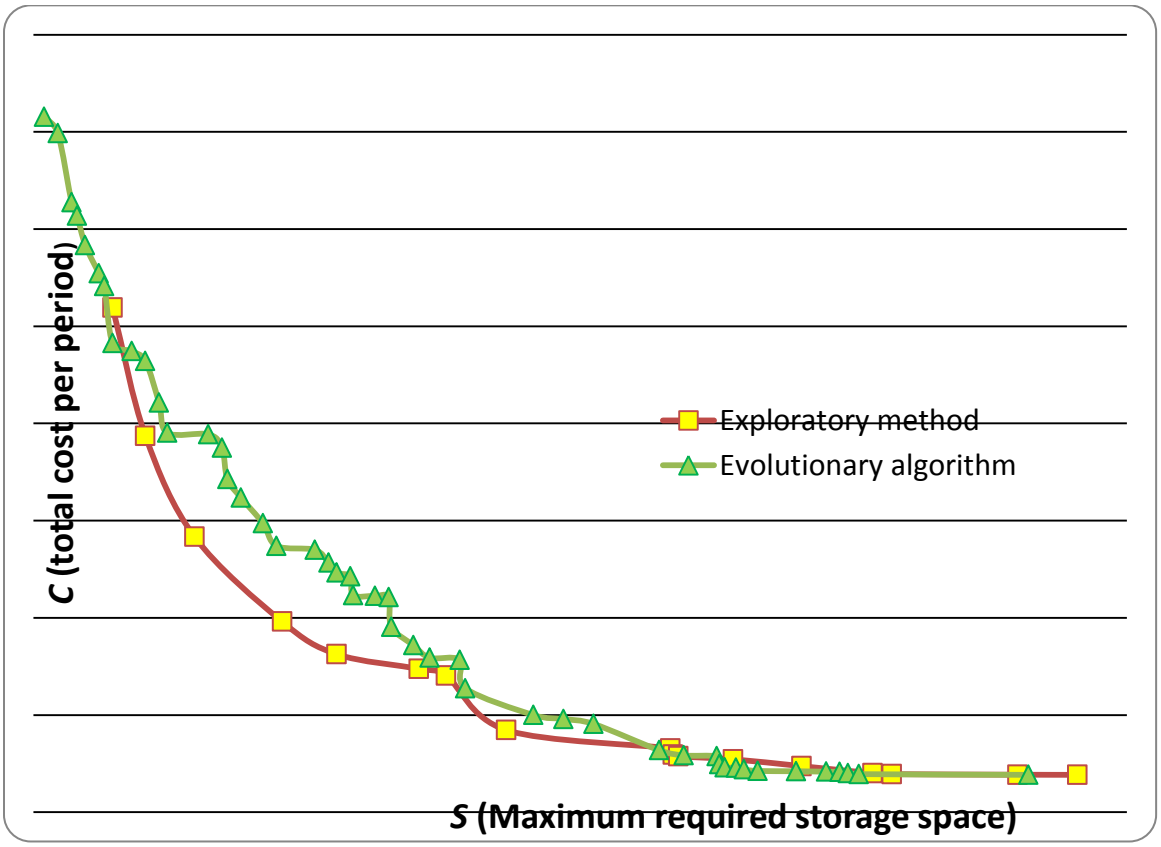

Figure 4: Pareto solutions obtained by each method (EM and TPEA)

Figure 5 presents the empirical Pareto Front (PF) of the solutions obtained by the two approaches together and shows by which approach each solution was produced. It also shows the empirical Utopia Point (UP) and the empirical Threat Point (TP) associated with this Pareto Front. The coordinates of the Utopia Point are the minimum values of the considered objectives while the coordinates of the Threat Point are the maximum values. The Compromise Solution (CS) is the nearest Pareto solution to the Utopia Point.

The distance between the Utopia point and the Pareto solutions is usually a weighted distance reflecting the importance of the different objectives. Thus the identification of the Compromise Solution depends on the used weights. Instead if using arbitrary weights, we transfer the origin of our coordinates to the Utopia point and rescale both axes by dividing all values by the range of the values on the corresponding axis. We notice that doing so, the compromise solution is one of those obtained by the exploratory method (For more details on these concepts see Kasprzak & Lewis 2001 and Marler & Arora, 2004).

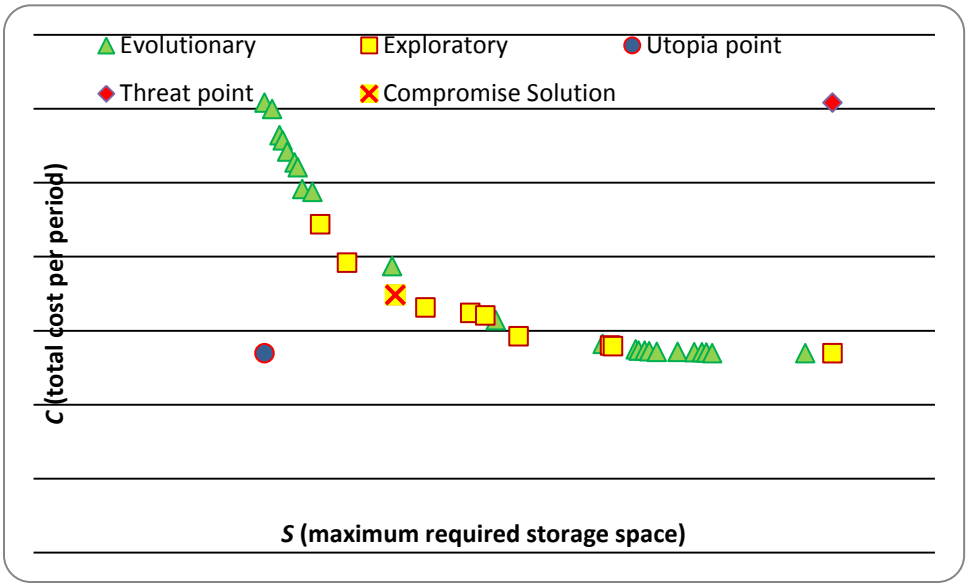

Figure 5: Empirical Overall Pareto Front (OPF)

4- PERFORMANCE EVALUATION

4.1- Test instances

To assess the performance of the proposed solution methods we generated 30 test instances with 20 items each. The demand rates are randomly and uniformly generated between 5 and 30 units per period. Order costs are generated between 10 and 100 while inventory holding cost are generated in a way to produce cycle times between 2 and 20. The generated test instances are available on: http://www.mcbolduc.com/tests.htm.

4.2- Performance criteria

Three criteria will be used to compare the performance of the proposed solution methods. The first criterion is the number of solutions on the Overall Pareto Front (OPF) produced by the method. The second and third are respectively the percentage of solutions on the PF as obtained by the method that are also on the OPF; and the number of time the compromise solution is obtained by the method.

4.3- Results and analysis

Table 4 gives the obtained results by the two proposed solution methods: the exploratory method (EM) and the Two-Population Evolutionary Algorithm (TPEA). The appendix gives the obtained results for each of the 30 test instances. From Table 4 we can see that the proposed evolutionary algorithm produced a larger number of solutions on the overall Pareto front (OPF) but the percentage of its Pareto solutions that are on the OPF is not much higher than the percentage for the exploratory method. The Exploratory method obtained the compromise solution in 13 out of the 30 test instances (43%) while the Two-population evolutionary algorithm obtained the compromise solution for 17 instances (57%). However it is important to notice that the TPEA requires about 26 times the computation time required by the EM.

Evaluation criterion	EM	TPEA
Average number of Pareto solutions obtained	29.7	54.3
Average number of solutions on the Overall Pareto Front (OPF)	15.8	35.9
Average percentage of PF solutions that are on the OPF	51.9	66.9
Number of times the method produced the Compromise Solution	13	17
Average computation time (sec)	42.0	1099.8

Table 4: Summary of the obtained results

5- CONCLUSIONS

This research work emphasises the fact that the inventory replenishment planning and staggering problem is a multi-objective one and in addition to minimizing the involved costs we need also to minimize the storage space and inventory handling resources. It was shown that a simple method, the Exploratory Method can produce useful solutions

and if more solutions are needed we can use the suggested Two-population Evolutionary Algorithm. Actually, it is suggested that we can use the two methods and produce an overall Pareto front of solutions that can be offered to the decision maker to choose the solution to implement.

It is also obvious that more efficient solution procedures should be developed and we also need to develop more evaluation criteria to assess the performance of the developed methods. Thus we consider this research work as the beginning of many future research works by us as well as by the other interested researchers.

Acknowledgement

This research work was partially supported by grants OPG0036509 from the National Science and Engineering Research Council of Canada (NSERC) and a grant from Université Laval. This support is gratefully acknowledged.

REFERENCES

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

Boctor F. F., 2010, Offsetting inventory replenishment cycles to minimize storage space, *European Journal of Operational Research*, 203, 321-325.

Deb K., A. Pratap, S. Agarwal and T. Meyarivan, 2002, A fast and elitist multi-objective Genetic algorithm: NSGA-II, IEEE Transactions on evolutionary computation, 6, 182-197.

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.

Kasprzak E., and K. Lewis, 2001, Pareto Analysis in Multiobjective Optimization Using the Colinearity Theorem and Scaling Method, *Structural and Multidisciplinary Optimization*, Vol. 22, 208-218.

Konak A., D.W. Coit and A.E. Smith, 2006, Multi-objective optimization using Genetic Algorithms: A Tutorial, *Reliability Engineering and System Safety*, 91, 992-1007.

Marler R.T. and J.S. Arora, 2004, Survey of multi-objective optimization methods for engineering, *Structural and Multidisciplinary Optimization*, 26, 369-395.

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.

Van Veldhuizen D.A. and G.B. Lamont, 2000, Multi-objective evolutionary algorithms: Analyzing the state of the art, *Evolutionary Computation*, 8, 125-147.

Zitzler E., K. Deb and L. Thiele, 2000, Comparison of Multi objective evolutionary algorithms: Empirical results, *Evolutionary Computation*, 8, 173-195.

Zitzler E., M. Laumanns and L. Thiele, 2001, SPEA 2: Improving the strength Pareto evolutionary algorithm, TIK report 103, Swiss Federal Institute of Technology, Zurich, Switzerland.

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

Appendix

Instance	EM				TPEA				Method producing the Compromise Solution
	Number of PF solutions	Number of PF solutions on the Overall Pareto Front (OPF)	Percentage of PF solutions that are on the OPF	Computation time (sec)	Number of PF solutions	Number of PF solutions on the Overall Pareto Front (OPF)	Percentage of PF solutions that are on the OPF	Computation time (sec)	
1	31	24	77.4%	8	60	20	33.3%	1089	EM
2	28	7	25.0%	26	56	47	83.9%	1242	TPEA
3	25	5	20.0%	27	55	51	92.7%	842	TPEA
4	31	17	54.8%	66	54	40	74.1%	1102	EM
5	34	12	35.3%	5	53	38	71.7%	1792	TPEA
6	29	16	55.2%	27	44	35	79.6%	1231	TPEA
7	30	11	36.7%	72	48	42	87.5%	871	EM
8	34	10	29.4%	16	44	40	90.9%	891	TPEA
9	34	18	52.9%	59	55	45	81.8%	1149	TPEA
10	30	21	70.0%	18	59	23	39.0%	997	TPEA
11	25	5	20.0%	21	48	39	81.3%	798	TPEA
12	22	19	86.4%	18	52	24	46.2%	954	EM
13	31	20	64.5%	127	54	31	57.4%	1119	EM
14	31	17	54.8%	66	50	34	68.0%	1097	EM
15	36	27	75.0%	183	57	31	54.4%	1035	EM
16	35	21	60.0%	20	48	30	62.5%	1138	TPEA
17	36	19	52.8%	12	66	49	74.2%	1313	TPEA
18	33	32	97.0%	170	61	21	34.4%	2120	EM
19	24	10	41.7%	6	53	33	62.3%	896	TPEA
20	30	4	13.3%	12	59	56	94.9%	1312	TPEA
21	21	6	28.6%	21	49	35	71.4%	1470	TPEA
22	30	7	23.3%	8	59	55	93.2%	811	EM
23	26	4	15.4%	18	55	53	96.4%	608	TPEA
24	31	28	90.3%	41	60	8	13.3%	1186	EM
25	37	28	75.7%	80	64	23	35.9%	1068	EM
26	33	23	69.7%	26	55	39	70.9%	1064	TPEA
27	26	21	80.8%	32	46	20	43.5%	1088	EM
28	34	21	61.8%	55	67	47	70.2%	966	TPEA
29	22	12	54.6%	17	52	30	57.7%	889	EM
30	23	8	34.8%	3	45	38	84.4%	857	TPEA
average	**29.7**	**15.8**	**51.9%**	**42.0**	**54.3**	**35.9**	**66.9%**	**1099.8**	
Minimum	**21**	**4**	**13,3%**	**3**	**44**	**8**	**13,3%**	**608**	
Maximum	**37**	**32**	**97,0%**	**183**	**67**	**56**	**96,4%**	**2120**	