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: An Evolutionary Algorithm

Fayez F. Boctor
Marie-Claude Bolduc

August 2016

CIRRELT-2016-41

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2016-009.

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

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
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: An Evolutionary Algorithm

Fayez F. Boctor*, Marie-Claude Bolduc

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

Abstract. This paper is the first one to suggest formulating the inventory replenishment problem as a bi-objective decision problem where, in addition to minimizing the sum of order and inventory holding costs, we should minimize the required storage space. Also, it develops a new solution method, called the two-population evolutionary algorithm (TPEA), to solve the problem. The proposed methods generate a near-optimal Pareto front of solutions with respect to the considered objectives. As the inventory replenishment problem have never been formulated as a bi-objective problem and as the literature does not provide any method to solve the considered bi-objective problem, we developed another but simpler solution method, called the exploratory method (EM) and we compared the results of the two developed Methods. The results obtained suggest that although the two-population evolutionary algorithm produces good Pareto near optimal solutions, we can apply both methods and choose among all the solutions obtained.

Keywords. Evolutionary algorithms, multi-criteria decision making, Pareto optimization, inventory management, heuristics.

Acknowledgements. This research work was partially supported by grant OPG0036509 from the National Science and Engineering Research Council of Canada (NSERC) and grant 96139 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, 2016

1- INTRODUCTION

The inventory replenishment planning problem is a central problem in the area of supply chain management. It involves two important decisions: The lot sizing decision or the determination of the quantities to order, and the determination of replenishment dates (or schedule). Two important objectives should be taken into consideration in making these decisions: (1) minimizing the total cost of ordering and holding the ordered items and (2) minimizing the required storage space. However, it is often impossible to make a replenishment plan that minimizes both objectives in the same time.

Obviously, if we have a sufficiently large amount of storage space, minimizing the required storage space becomes a useless objective. In this case we only need to minimize the total cost of ordering and holding the required items. However, we observe that storage space often becomes insufficient after few years of operating a storage facility and we need to add some extra storage space.

The question becomes how much extra space we need? One way to answer this question is to solve the replenishment planning problem as a bi-objective one where we seek to minimize both the total inventory cost and the maximum required storage space. The extra space can then be determined in function of the difference between the maximum required space and the current storage capacity. Let us also notice that the decision maker in such a situation always prefer to consider several possible solutions. Thus it is important to provide him or her with several non-dominated solutions; i.e., an optimal or near optimal Pareto front of solutions.

We notice that it is a usual practice to solve the inventory replenishment planning problem by decomposing it into two sub-problems and solving them separately. First we determine the optimal or near-optimal quantities to order without neither determining replenishment dates nor taking into consideration the required storage space. Then, using the determined order quantities, we determine the replenishment dates (schedule) that minimize the maximum required storage space. This solution approach cannot allow us to analyze the trade-off between the total cost and the maximum required storage space. It

may even produce an infeasible solution if the resulting maximum required storage space exceeds the available space.

In fact, for many warehouse operations, it is admitted that decomposing the replenishment planning problem into these two sub-problems and solving them separately is not satisfactory. That is why in this paper we suggest to consider the inventory replenishment planning problem as a bi-objective problem. This is the first and only paper that formulates and deals with the replenishment planning problem as a bi-objective one. Consequently, instead of providing the decision maker with a unique solution, we construct a series of non-dominated (Pareto or near-Pareto) solutions leaving the final choice to him or her.

2- LITERATURE REVIEW

Numerous versions of the inventory replenishment planning problem are studied in the open literature. A classification of these versions can be made based on the assumptions made in each one. This classification is based on:

1) Whether warehouse capacity is limited or unlimited;
2) Whether the production capacity of suppliers (or the production facilities) is limited or not,
3) Whether the production process is a single level process or a multi-level one;
4) Whether demands are deterministic or stochastic. Also, deterministic demands are either constant (static) or variable (dynamic),
5) Whether ordering costs of products are completely independent of each other or there is also a common ordering costs for some groups of products,
6) Whether backlogs and/or backorders are allowed or not,
7) Whether the planning horizon is finite or infinite, and
8) Whether the replenishment plan is cyclic or not.

Two types of cyclic replenishment patterns can be used. Either we require that for each product the time between any two successive replenishments of the product is always the same (constant product cycle time), or, we require that some reception schedule (that covers a finite time period) be exactly repeated up to the end of the planning horizon. The

former case will be designated by “product-cyclic” and the later by “schedule-cyclic”. All product-cyclic replenishment solutions are also schedule-cyclic. A special case, called the “common cycle” case, is the one where all cycle times for all products are of the same length.

The simplest version of the replenishment planning problem is the one where: demands are static (constant) and deterministic, the planning horizon is infinite, no backlogs are allowed, we can order any quantity any time, warehouse capacity is unlimited and we require a product-cyclic replenishment plan. In this case there is no interaction between the decisions to make for the different products, the zero-switch rule is optimal and the quantities to order can be determined by the famous Economic Order Quantity (EOQ) formula developed by Harris (1913). Several special cases of this basic problem are presented in Silver, Pyke and Peterson (1998). An extensively studied special case of this basic problem is the “Joint Replenishment” problem where products are grouped in product families and there is a common (major) ordering cost for each family in addition to a minor ordering cost for each product. A review of the contributions to this NP-hard problem is provided in Khouja and Goyal (2008).

In the case where demands are varying over a finite scheduling horizon and all the other assumptions are the same as in the EOQ model, the problem can be solved by dynamic programming as shown by Wagner and Whitten (1958). As indicated in Boctor et al. (2004), the joint replenishment version of this dynamic demand finite horizon problem where there are common ordering costs for product families, is NP-hard. A review of the models and algorithms to solve this dynamic joint replenishment problem is presented in Robinson et al. (2009).

The single machine Economic Lot Sizing Problem (ELSP) generalizes the EOQ problem by taking into consideration the capacity of the machine used to process the required products. There is no recent review of the literature dealing with this problem. A succinct review of some solution methods can be found in Boctor (1985). One of the most efficient solution methods is the G-group heuristic developed in Boctor (1987).

The multiple machine flow shop version of this problem was first solved by El-Najdawi and Kleindorfer (1994) under the assumption that all products have the same cycle time (the common cycle assumption). Five solution approaches were tested in Ouenniche and Boctor (2001) showing that the G-group heuristic proposed in this same paper outperformed previously published methods. The job shop version of this problem is considered in Ouenneiche and Boctor (1998).

The capacitated lot sizing problem (CLSP) is a generalization of the ELSP where the planning horizon is finite and demands varies over this horizon. A review of the literature dealing with the single-level and single machine version of this problem can be found in Karimi et al. (2003) and the literature dealing with the multi-level version is reviewed in Buschkühl et al. (2010).

In all the above mentioned literature, the warehouse capacity is not taken into consideration and as thus there is no need to stagger the replenishment of items in order to reduce the maximum required storage space.

Two approaches are used in order to take into consideration the limits on warehouse capacity or more specifically in order to minimize the maximum storage space required. In all the literature presented hereafter it is assumed that demands are deterministic and static (constant).

The first approach, that will be called the ***decomposition approach***, solves the problem in two steps. It determines the quantities to order and then construct a replenishment schedule. As mentioned above, quantities to order can be determined by the EOQ formula but this is not necessarily feasible if the available storage space is not sufficient to stock the resulting quantities. Murthy et al. (2003) assumes that quantities and replenishment cycle lengths are given and propose a heuristic to determine the replenishment periods (or dates) with the objective to use the smallest possible storage space. Moon et al. (2008) as well as Yao and Chu (2008) present a mixed integer formulation of the problem and use the genetic algorithm to determine the replenishment periods. Boctor (2010) proposes three heuristics to determine replenishment periods and shows, using three sets of 30 test-instances each, that all three proposed heuristics

outperform the one proposed by Murthy et al. (2003). Gallego et al. (1996) also propose a staggering heuristic using powers-of-two cycle times.

The second approach that take into consideration the warehouse capacity is to add a constraint assuring that the sum of required storage space for all ordered quantities is less than or equal to a pre-selected limit. This approach, called hereafter the ***space-constraining approach***, is not appropriate if we need to determine the capacity of the warehouse to build or if we have insufficient capacity and want to minimize the extra capacity to add.

Page and Paul (1976), Zoller (1977) and Hall (1988) use this constraint and propose a method to stagger deliveries optimally assuming a common replenishment cycle for all products. Page and Paul (1976) show that their staggering approach often leads to better solutions than solving the problem using Lagrangian relaxation. Goyal (1978) suggest to improve Page and Paul's method by allowing more than one delivery within the considered cycle time. Hartley and Thomas (1982) and Thomas and Hartley (1983) consider the case where we have only two product. Anily (1991) develop a common cycle heuristic and study its worst case performance. Gallego et al. (1992) show that the problem is *NP*-hard even if only one cycle multiple is different from the others. Rosenblatt and Rothblum (1990) assume a common cycle for all products and minimize the sum of not only order and inventory holding costs but add to them the warehouse capacity usage cost.

To make the problem easier to solve, some contributions pre-determine some of the solution decisions and propose methods to determine the remaining decisions. For example, Hariga and Jackson (1996) formulate the problem where the number of all orders, denoted *m*, as well as the number of orders for each product within the cycle is given (pre-determined). Then they propose a model to determine the product and the quantity to order for each of the *m* orders, the dates for the delivery of these orders, and the time length of the ordering schedule. However, as the suggested model is almost impossible to solve (but for very small problem instances), they also propose to pre-determine an ordering sequence, i.e., to pre-determine which product to order at each of

the *m* orders. Hariga and Jackson called this problem the *warehouse scheduling problem*. However we notice that this name is used by other researchers to designate several other different problems. So this name will not be used in this paper.

From the above literature review we can see that the problem of simultaneously determining the quantities to order and the replenishment dates without imposing any additional constraints or making any decisions a priori is not studied in the literature. This paper considers this problem and rather than searching for a unique solution, it proposes to provide the decision maker with a series of non-dominated, near-optimal Pareto solutions and let him or her to choose the preferred one.

3- PROBLEM DEFINITION AND MATHEMATICAL FORMULATION

This paper deals with inventory replenishment of multiple products supplied by external suppliers to fulfil constant demands. These products are to share the same storage space and we want to minimize the maximum required storage space concurrently with minimizing the total cost. Two cost elements are considered: ordering cost and inventory holding cost. Products order costs are assumed known, constant and independent of each other. Also, per unit inventory holding costs are assumed known and constant. It is required that replenishment be product-cyclic and that cycles be integer multiples of a basic time period called the fundamental cycle. This basic period is also required to be an integer number of time units (e.g. days).

Let us emphasize that minimizing the maximum required storage space is of interest in many cases. For example this is necessary if we want to determine the capacity of a warehouse to build or to determine the extra space to add to an existing warehouse. Also, staggering inventory replenishment in a way that minimizes the maximum required storage space is necessary if the available space is not sufficient otherwise. However, in the case where we have a warehouse of sufficiently large storage space, minimizing the maximum storage space is not necessary.

More precisely, the problem considered in this paper is the problem of determining the time length of the fundamental cycle, the multiples that determine replenishment cycles

of products, as well as replenishment dates. The problem will be formulated as a bi-objective one and the objectives are to minimize the sum of order and inventory holding costs, denoted *C*, and to minimize the maximum required storage space, denoted *S*.

So we seek to determine the replenishment cycles of *N* different products that share the same storage space. Each product *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 product *i*, denoted d_i, is known and constant and as no backlogging is allowed, the replenishment quantity Q_i should equals $m_i b d_i$. For each product *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 of all cycle multipliers m_i. The number of replenishments of *i* within the global cycle, denoted n_i, equals M/m_i. Two more variables are needed: x_{it} which is a binary that takes the value 1 if product *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 alternatively, we assume that the measuring unit of *i* is the quantity that requires one unit of storage space.

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:

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

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

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

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

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

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

The first objective function gives the sum of ordering and inventory holding costs per time unit and the second objective is to minimize the maximum requires storage space. Constraints (3) assure that each product is replenished once and only once during its first replenishment cycle (and the consecutive cycles). Constraints (4) determine the maximum storage space required while constraints (5) determine inventory levels at the beginning of each period and assure that replenishment is product-cyclic.

This is a non-linear bi-objective program where the integers m_i are not only direct decision variables but also determine the limits of the sum of x_{it} in constraints (3).

4- TWO HEURISTIC SOLUTION APPROACHES

As we don't have any efficient method to solve the problem as modeled above, we need to develop some heuristic methods to produce good but not necessarily optimal solutions. In the following, two such heuristics are proposed.

The first approach is an evolutionary algorithm; called hereafter the ***Two-Population Evolutionary Algorithm*** (TPEA), which produces a large number of solutions, identifies the non-dominated ones and leaves the final choice to the decision maker. Notice that a solution (C_1,S_1) is dominated by another solution (C_2,S_2) if either $C_1 \geq C_2$ and $S_1 > S_2$, or $C_1 > C_2$ and $S_1 \geq S_2$.

The second approach, called hereafter the ***exploratory method*** (EM) is proposed to help assessing the performance of the TPEA. It enumerates a number of different vectors of cycle multipliers m_i, $i=1, \ldots, N$. For each vector, it determines the corresponding total cost C and solves the resulting replenishment staggering problem where replenishment periods are determined with the objective to minimize the maximum required storage space. This approach may provide as many solutions as the number of cycle-multiplier vectors we are willing to consider. Dominated solutions should then be discarded.

Let us first present the exploratory method and then we introduce the two-population evolutionary algorithm.

4.1- The exploratory method (EM)

The method can be summarized as follows: we generate a number of cycle multiplier 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. In the following we present the steps of the method.

Step 1: multiplier vectors generation. To generate the required multiplier vectors, we solve the following unconstrained and separable program while using a number of preselected values of *b*:

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

$$\text{Minimize: } C = \sum_{i=1}^{N} C_i(m_i) = \sum_{i=1}^{N} \left\{ \frac{O_i}{m_i b} + \frac{h_i d_i m_i b}{2} \right\}. \tag{7}$$

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

$$m_i^* = \begin{cases} \mu_i & \text{if } C_i(\mu_i) \leq C_i(\mu_i + 1) \\ \mu_i + 1 & \text{otherwise} \end{cases} \quad ; \text{where } \mu_i = \left\lfloor \frac{1}{b} \sqrt{\frac{2O_i}{h_i d_i}} \right\rfloor \tag{8}$$

The number of generated vectors depends on the number of values of *b* we use. In the numerical tests reported in section 5, we used all values of *b* between two chosen values b_l and b_u with a step of Δb. These values of *b* are only used to generate the multiplier vectors and will not be used in the following steps. Instead, in all the solutions generated in step 2 we always use $b=1$.

Step 2: Solutions generation. For each of the obtained multiplier vectors we calculate the corresponding total cost *C* by substituting the values of m_i; $i=1,\ldots, N$, and $b=1$ in (7). Afterwards, we move to determine the replenishment periods for each product. This can be done either by using a heuristic or by solving the model (Model 2):

Minimize: *S*
Subject to: constraints (3) to (6).

This model is a mixed integer model with binary and continuous variables.

Alternatively we can solve a more compact (but completely equivalent) model where all decision variables are binary variables. This alternative model (Model 3) is:

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:

Minimize: S

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

$$\sum_{i=1}^{N} \sum_{s=1}^{m_i} x_{is} I_{ist} \le S \qquad ; t = 1,...,M. \tag{9}$$

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

$$I_{ist} = \begin{cases} Q_i - (t-s)d_i & \text{, for } s \le t < s + m_i \\ I_{is,t \bmod(m_i)} & \text{, for all other values of } t \end{cases} \tag{10}$$

Step 3: Discard dominated solutions. A solution (C_1,S_1) is dominated by another solution (C_2,S_2) if either $C_1 \ge C_2$ and $S_1 > S_2$, or $C_1 > C_2$ and $S_1 \ge S_2$. Discard all dominated solutions and provide the decision maker with the remaining ones. The remaining solutions constitute a near-Pareto front of solutions.

Staggering Heuristic *H*1

Instead of solving a mathematical model to determine the replenishment periods that minimize S, a faster solution can be found by using a heuristic method. In this paper the following heuristic is used. Alternatively, we may use any other heuristic among those presented in Boctor (2010) or develop a new one.

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

Iteration: Repeat the following for all the remaining products in the list:
- Consider the next product in the list, denoted i,
- Schedule its replenishments at $f + nm_i$ (n=0, 1,…, n_i-1) where $f \le m_i$, the first replenishment period, is the one which produces to the smallest maximum space required for the already scheduled products (i.e., products u,…, i).

Improvement: repeat the following until no further improvement can be achieved:

- Consider all products one by one.
- Let $f + nm_i$ ($n = 0, 1,\ldots, n_i\text{-}1$) be the replenishment periods of the considered product i and δ_{ifg} be the variation of the maximum required storage space if the replenishment of i is moved to periods $g + nm_i$ ($n = 0, 1,\ldots, n_i\text{-}1$) with $g \leq m_i$ and $g \neq f$.
- If there are one or more values of g that lead to a reduction of the maximum required space (i.e., such that $\delta_{ifg}<0$), move the replenishment of i to periods $k + nm_i$ ($n = 0, 1,\ldots, n_i\text{-}1$) where: $\delta_{ifk} = min_g\ \delta_{ifg}$

Numerical illustration

Let us consider the ten-product example presented in Table 1 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. Then use the Heuristic $H1$ to determine S, the corresponding maximum required storage space. We obtain the 17 non-dominated solutions given in Table 2. To get a rough estimation of by how much we can reduce the maximum required storage space if we used the mathematical model (2), (3) and (9) instead of heuristic $H1$, we solved this model to determine S^* for each of the 17 non-dominated solutions. The obtained values of S^* are given by the fourth column of Table 2. These results, constituting the empirical Pareto Front (PF) as obtained by the EM, are depicted on Figure 1.

Obviously using solving the mathematical model requires more computational time and gives smaller values of S than those obtained by heuristic $H1$; the average improvement of S is 2.58% with a minimum of 0% and a maximum of 5.1%. Indeed, the additional computational time for solving the model for such a small instance is 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.

product	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 1: Data for the ten-product example

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

Table 2: The 17 non-dominated solutions obtained by the exploratory method

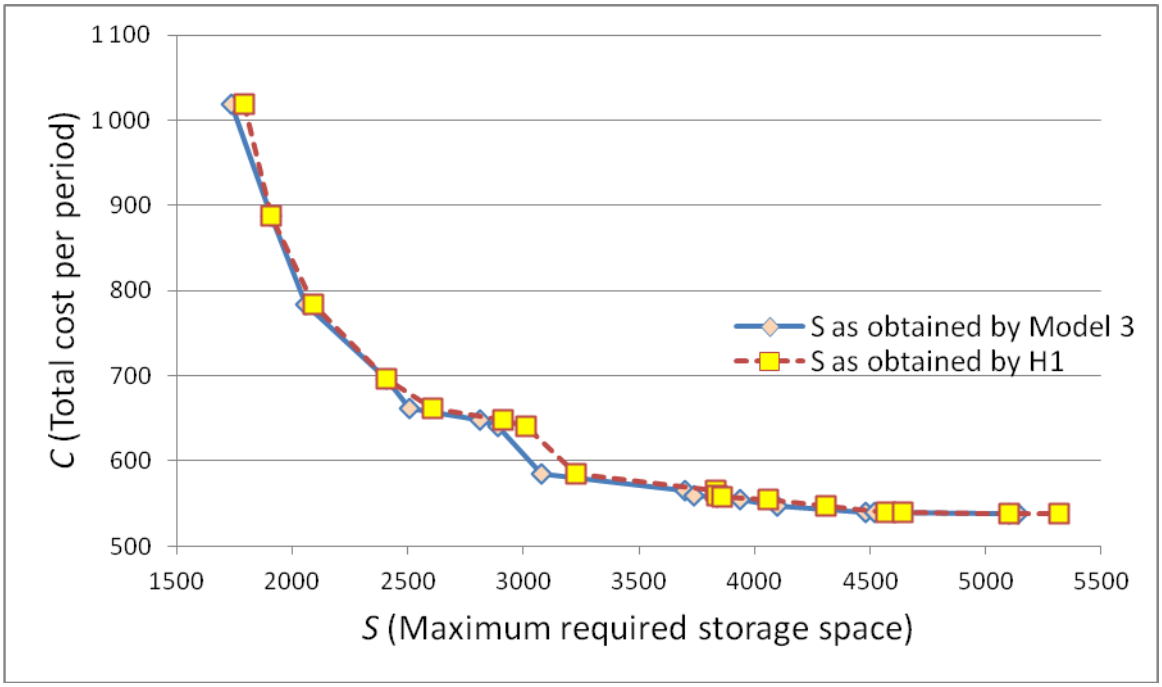

Figure 1: Non-dominated solutions (Pareto Front) as obtained by the Exploratory Method (EM)

4.2- The two-population evolutionary algorithm (TPEA)

The second approach we propose to solve the 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 outperforms the others is not yet settled. Few comparative studies (Zitzler et al. 2000) attempted to answer 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 the first population is measured by the total cost and the fitness of the second population is measured by the maximum required 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 replenishment period for each product are not coded as a part of the solution. Instead, for each multipliers vector we apply heuristic *H*1 to determine the replenishment periods and the maximum required storage space *S*.

Number and size of populations: we need to use as many populations as the number of considered objectives. In our case as we have two objectives (to minimize *C* and to minimize *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 our implementation of the algorithm, we use two populations of 100 solutions each (*P*=100).

Initial population: We generate 2*P* solutions (chromosomes) and we use the first *P* chromosome as the first population and the others as the second one. To generate these 2*P* chromosomes we first generate a seed chromosome (i.e., one cycle multipliers vector) by using equation (8) with *b*=1. Then we generate each of the remaining chromosomes as follows. Randomly select between 4 and 8 genes (products) in the seed vector and randomly modify their cycle multiplier by adding or subtracting a random value between 1 and a predetermined value *w*. However, if a modified multiplier becomes less than 1 we give it the value 1.

For each generated vector (chromosome) we calculate its total cost per period *C* and apply heuristic *H*1 to determine *S* the corresponding maximum storage space required.

Mating pool selection: the selection of individuals to participate in the production of the next generation of the first population is done as follows:

1- Arrange the overall set of all the 2*P* individuals of the current generation of the two populations in the ascending order of their total cost *C*,
2- Retain the top *P* individuals of the resulting list. If a solution has more than one copy in the retained set of individuals, keep only one and remove the other copies.
3- If the number of remaining individuals is greater than a preselected threshold *q* (in our implementation $q=0.8P$) keep only the first *q* individual and remove the others. Let *p* be the number of remaining individuals, add a second copy the best *P-p* individuals (those with the lowest values of *C*) to the mating pool.

To generate the mating pool for the second population, repeat the above steps using *S* instead of *C* for arranging the list.

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-product numerical example

The proposed two-population evolutionary algorithm was applied to the 10-product example presented in Table 1. The algorithm was stopped once 30 generations are produced. The results are depicted on Figure 2. Again, Model 3 was also used instead of the heuristic *H*1 to see by how much the maximum required space can be reduced. This is done for each of the 46 non-dominated solutions obtained. These solutions are presented in Figure 2.

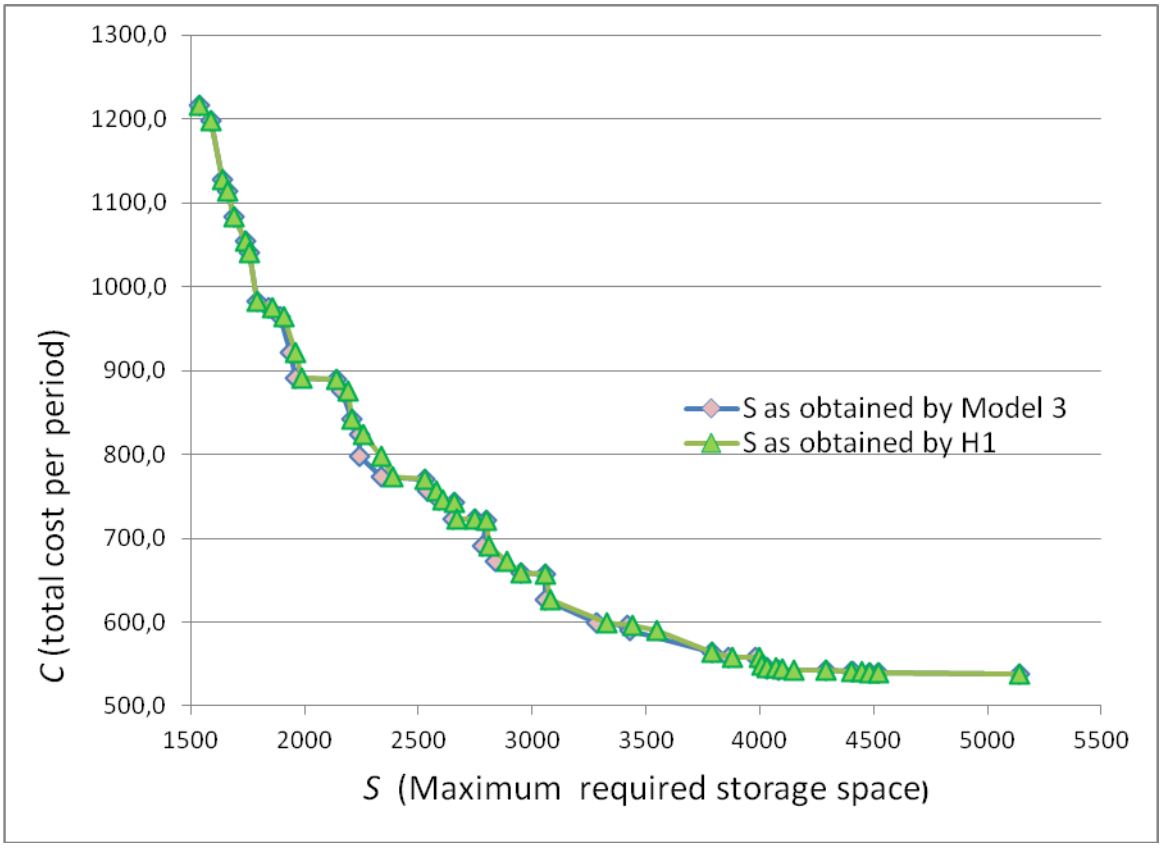

Figure 2: Non dominated solutions as obtained by the Two-Population Evolutionary Algorithm (TPEA)

Figure 3 presents the 46 non-dominated solutions (PF) obtained by the two-population evolutionary algorithm (TPEA) together with the 17 non-dominated solutions (PF) of the exploratory method (EM). If we put all these 63 solutions together we see that only 32 of them are non-dominated. Ten (59%) of the 17 solutions of the exploratory methods are on the Overall Pareto Front (OPF) as they are not dominated by any of the solutions obtained by the evolutionary algorithm while 7 solutions are dominated. Also 22 of the 46 solutions of the evolutionary algorithm are not dominated (48%) while the other 24 are dominated by those obtained by the EM. Notice that although the proposed evolutionary algorithm has more solutions on the Overall Pareto Front (OPF), a higher percentage (52%) of its PF solutions are dominated by those on the PF of the exploratory method.

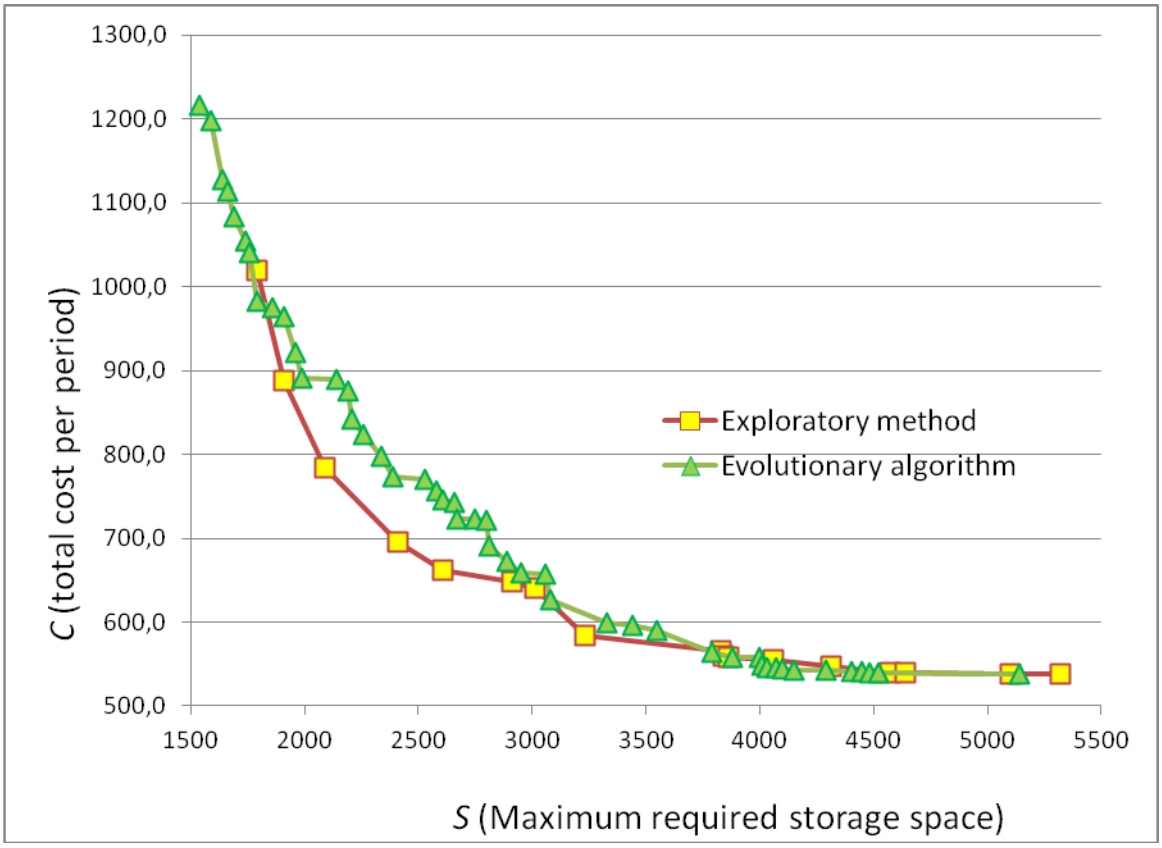

Figure 3: Non-dominated solutions obtained by each method (EM and TPEA)

Figure 4 presents the empirical Overall Pareto Front (OPF) 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 given by the overall minimum value of each 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. (For more details on these concepts see Kasprzak and Lewis 2001 and Marler and Arora, 2004). The distance between the Utopia point and the Pareto solutions is usually measured by a weighted distance reflecting the importance of the each objective. In this case, the identification of the Compromise Solution depends on the used weights.

In this paper, instead of using arbitrary weights, we use the following. Transfer the origin of our coordinates to the Utopia point, rescale both axes by dividing all values by the

range of the values on the corresponding axis, and attribute equal weights to the two objectives. For the illustrative example, the compromise solution is shown on Figure 4 and is one of the solutions obtained by the exploratory method. The compromise solution is often proposed to the decision maker as the suggested solution for the problem.

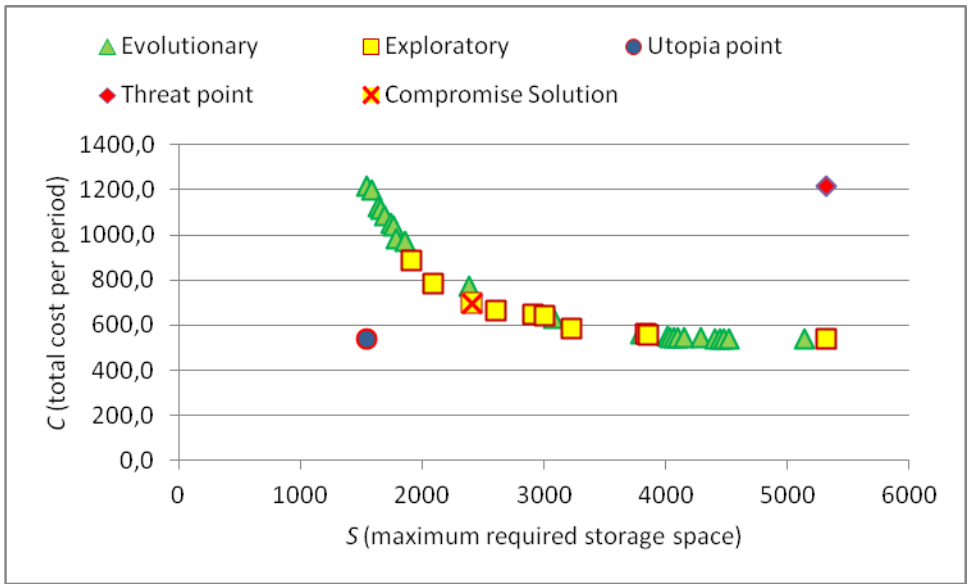

Figure 4: Empirical Overall Pareto Front (OPF)

5- PERFORMANCE EVALUATION

Evaluating the performance of multi-objective solution methods is an important and difficult issue (see Konak et al 2006). Usually, it is suggested to measure this performance by determining how close the generated empirical Pareto Front is to the optimal Pareto Front. However, it is often impossible to identify the optimal Pareto Front. Hereafter we suggest 3 criteria to compare the two developed methods and we believe that these criteria are useful for such a comparison.

5.1- Test instances

To assess the performance of the proposed solution methods we generated 30 test instances with 20 products 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 is generated in a way to produce optimal cycle multipliers (as

determined by equation 8) between 2 and 20. These test-instances are available on: http://www.mcbolduc.com/tests.htm.

5.2- Performance criteria

Each test instance is solved by both the EM and the TPEA and we determine their Pareto fronts (PF). Then we determine the Overall Pareto Front (OPF) of the combined set of the two front's solutions. To evaluate the performance of these two methods we use the following three criteria: (1) the number of solutions on the method's PF that are also on the OPF, (2) the percentage of solutions obtained by the method that are on the OPF with respect to the total number of OPF solutions, and (3) the number of times the compromise solution is obtained by each method.

5.3- Results and analysis

Table 3 gives the results obtained by the two proposed solution methods. The appendix gives the obtained results for each of the 30 test instances. From Table 3 we can see that the proposed two-population evolutionary algorithm (TPEA) produced a larger number of solutions that are on the overall Pareto front (OPF) but the percentage of the solutions of its Pareto front that are on the OPF is not much higher than the percentage produced by the exploratory method (EM). The Exploratory method obtained the compromise solution for 13 out of the 30 test instances (43%) while the two-population evolutionary algorithm produced the compromise solution for the other 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 PF solutions	29.7	54.3
Average number of PF solutions that are on the 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 3: Summery of the obtained results

6- CONCLUSIONS

This research work suggests that, in many situations, the inventory replenishment planning and staggering problem should be treated as a multi-objective one where in addition to minimizing the involved costs we need also to minimize the maximum required storage space. It is 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. However, we also suggest applying both methods, in order to obtain a better overall Pareto near-optimal front as well as to produce a good compromise solution. All these solutions can be offered to the decision maker to choose the one to implement.

Although the proposed methods seem efficient, researchers are encouraged to develop other and hopefully more efficient solution procedures. Also we may need to develop other evaluation criteria and use them to assess the performance of the developed methods.

Acknowledgement

This research work was partially supported by grant OPG0036509 from the National Science and Engineering Research Council of Canada (NSERC) and grant 96139 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 FF, 1985, Single machine lot scheduling: a comparison of some solution procedures, *RA IRO Automatique Productique et Informatique Industrielle* 19, 389-402.

Boctor FF, 1987, The G-group heuristic for single machine lot scheduling, *International Journal of Production Research* 25, 363- 379.

Boctor FF, G Laporte and J Renaud, 2004, Models and algorithms for the dynamic demand joint replenishment problem, *International Journal of Production Research* 42, 2667–78.

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

Buschkühl L, F Sahling, S Helber and H Templemeier, 2010, Dynamic capacitated lot-sizing problems: a classification and review of solution approaches, *OR Spectrum* 32, 231-261.

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.

El-Najdawi M and PR Kleindorfer, 1993, Common cycle lot-size scheduling for multi-product, multi-stage production, *Management Science* 39, 872- 885.

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

Hall NG, 1988, A comparison of inventory replenishment heuristics for minimizing maximum storage, *American Journal of Mathematical and Management Sciences* 18, 245-258.

Hariga MA and PL Jackson, 1995, Time variant lot sizing models for the warehouse scheduling problem, *IIE Transaction* 27, 162-170.

Hariga MA and PL Jackson, 1996, The warehouse scheduling problem: formulation and algorithm, *IIE Transaction* 28, 115-127.

Harris FW, 1913, How many parts to make at once, *Operations Research*, 38, 947-950 (Reprint from: *Factory, The Magazine of Management* 10, 135–36, 152).

Hartley R and LC Thomas, 1982, The deterministic, two-product, inventory with capacity constraint, *Journal of Operational Research Society* 33, 1013–1020.

Karimi B, SMT Fatemi Ghomi and JM Wilson, 2003, The capacitated lot sizing problem: A review of models and algorithms, *Omega* 31, 365-378.

Kasprzak E, and K Lewis, 2001, Pareto analysis in multi-objective optimization using the colinearity theorem and scaling method, *Structural and Multidisciplinary Optimization* 22, 208-218.

Khouja M and S Goyal, 2008, A review of the joint replenishment problem literature: 1989–2005, *European Journal of Operational Research* 186, 1-16.

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

Marler RT and JS Arora, 2004, Survey of multi-objective optimization methods for engineering, *Structural and Multidisciplinary Optimization* 26, 369-395.

Moon IK, BC Cha and SK Kim, 2008, Offsetting inventory cycles using mixed integer programming and genetic algorithm, *International Journal of Industrial Engineering* 15, 245-256.

Murthy NN, WC Benton and PA Rubin, 2003, Offsetting inventory cycles of items sharing storage, *European Journal of Operational Research* 150, 304-319.

Ouenniche, J and FF Boctor, 1998, Sequencing, lot sizing and scheduling of several products in job shops: the common cycle approach. *International Journal of Production Research* 36, 1125-1140.

Ouenniche J and FF Boctor, 2001, The G-group heuristic to solve the multi-product, sequencing, lot sizing and scheduling problem in flow shops, *International Journal of Production Research* 39, 81-98.

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

Robinson P, A Narayananb and F Sahin, 2009, Coordinated deterministic dynamic demand lot-sizing problem: A review of models and algorithms. *Omega* 37, 3-15.

Rosenblatt MJ and UG Rothblum, 1990, On the Single Resource Capacity Problem for Multi-Item Inventory Systems, *Operations Research* 38, 686-693.

Silver E, D Pyke and R Peterson, 1998, *Inventory Management and Production Planning and Scheduling*, Third edition, Wiley.

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

Wagner HM and TM Whitin, 1958, Dynamic version of economic lot size model, *Management Science* 5, 89–96.

Yao MJ and WM Chu, 2008, A genetic algorithm for determining optimal replenishment cycles to minimize maximum warehouse space requirements, *Omega* 36, 619-631.

Thomas LC and R Hartley, 1983, An algorithm for the limited capacity inventory problem with staggering, *Journal of Operational Research Society* 34, 81–85.

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 Pareto Front (PF) solutions	Number of PF solutions also 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	