Ecole Polytechnique Fédérale de Lausanne
Département de Mathématiques
Chaire de Recherche Opérationnelle
CH-1015 Lausanne

Abstract: This paper is concerned with a multi-period inventory problem where the demand and the supplier capacity of a given key resource are discrete random variables with known probability distribution. The procurement manager has to answer two types of questions: how much supplier quantity to reserve for each period (where all of these reservations have to be made **prior** to the start of the first period) and how large an order quatity to use at start of each period. An exact procedure, based on branch and bound and dynamic programming, as well as various heuristics methods to solve the problem are presented in this paper. Heuristic methods are shown to perform extremely well in the sense that they provide near optimal strategies while requiring much less computational effort than the exact method.

Exact and approximate algorithms for the multi-period procurement problem where dedicated supplier capacity can be reserved

Daniel Costa *(1)*
Edward A. Silver *(2)*

(1) Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland. Email: *costa@dma.epfl.ch.* Fax: +41 (0)21 693 4250

(2) Faculty of Management, University of Calgary, 2500 University Drive NW, Calgary, Alberta, Canada T2N 1N4. Email: *silver@acs.ucalgary.ca.* Fax: +1 403 282 0095

1. Introduction

In an earlier study [9] a number of procurement managers in large scale projects in the oil and gas industry were interviewed, including asking them to identify problems of concern. One such problem motivated the research leading to this paper. Specifically, we consider a multi-period situation where there are two types of uncertainty present with respect to the procurement of a key resource. First, the requirements period by period are only known probabilistically. Second, the supplier capacity in any period is also a random variable with a known probability distribution. In addition, there is the option of reserving some capacity in one or more periods, but paying a premium for such reservations. In contrast with the replenishment quantities, which can be selected at the start of each period (when the associated inventory level is known), all capacity reservations must be finalized well ahead of time (due to the much longer lead time required by suppliers for such **firm** commitments). Motivation and further clarifications about the problem area under study in this paper can be found in [11], the latter prepared more for a project management audience.

The remainder of this paper is organized as follows. In the next section we present the mathematical model of the above problem as well as the assumptions and notation underlying it. In section 3, we introduce a branch and bound procedure, based on dynamic programming, to find an optimal strategy. This approach is well suited for small scale procurement and inventory management problems but can on no account be used for larger scale problems typically encountered in practice. We present in section 4 some heuristic methods yielding satisfactory (i.e. reasonable, yet mathematically suboptimal) solutions to complex realistic size problems. Various computational experiments are reported in section 5. Finally some conclusions are drawn in section 6.

2. Model description

The purpose of this section is to lay out a detailed mathematical model corresponding to the procurement problem mentioned in the introduction.

We assume that there exist N planning periods, such as a week or month, and that all reorder intervals are an integer multiple of this base period. Let us consider a particular key resource with which two types of uncertainty are associated. Its requirements (X_i) as well as the capacities (C_i) available from the supplier in period i (i=1, ..., N) are independent discrete random variables with known probability functions $p_i(X_i)$ and $g_i(C_i)$ respectively. In most of the studies reported in the literature (see [4, 8] for instance) the problem is to determine at the start of each period i the order quantity q_i which minimizes the expected total cost. As mentioned in the introduction we also permit dedicated capacity levels to be fixed prior to the start of period 1 by paying a premium to the supplier. This ensures the receipt in each period i of the number of resource units corresponding to the capacity reserved r_i if the capacity C_i available from the supplier happens to be smaller than r_i. The two types of decisions that have to be made in this problem are:

a) prior to the start of period 1, the sequence $(r_1, r_2, ..., r_N)$ of dedicated capacity levels, and
b) at the start of each period i, the replenishment quantity $q_i(I_i)$ to order if the inventory level is I_i. Clearly, the quantity $q_i(I_i)$ has to be ordered before the requirement X_i and the capacity C_i are known.

The X_i's, C_i's, and r_i's are limited to discrete values, with the set depending upon the period i, as follows:

$$X_i \in \mathcal{X}_i = \{x_i^1, x_i^2, ..., x_i^{nx_i}\}$$
$$C_i \in \mathcal{C}_i = \{c_i^1, c_i^2, ..., c_i^{nc_i}\}$$
$$r_i \in \mathcal{R}_i = \{r_i^1, r_i^2, ..., r_i^{nr_i}\}$$

We assume that the values in $\mathcal{X}_i$, $\mathcal{C}_i$, and $\mathcal{R}_i$ are sorted in increasing order (e.g. $x_i^j < x_i^{j+1}$ $\forall j=1, ..., nx_i-1$, etc). Moreover, we have $r_i^1=0$ ($\forall i=1,...,N$) since reserving capacity in advance is not mandatory.

Costs in this model include (they could all be made functions of the period):

- v : unit acquisition cost of the resource
- A : fixed cost associated with an acquisition
- B_1 : fixed cost for any shortage in a period
- B_2 : cost per unit short in a period ($B_2>v$)
- $\phi(r)$: monotonic increasing function representing the cost of reserving capacity r. The cost of reserving grows with r because it is more and more difficult for the supplier to ensure the receipt of a larger and larger number of resource units.
- h : inventory carrying charge per unit leftover at the end of a period. In period N, h = -s where s is the salvage value per unit of resource.

The goal is to find a strategy $(\underline{r}^*, \underline{q}^*) = (r_1^*, r_2^*, ..., r_N^*, q_1^*(I_1), \underline{q}_2^*(\cdot), ..., \underline{q}_N^*(\cdot))$ minimizing the expected cost over the N periods considered. We assume that there is no inventory at the start of period 1 ($I_1=0$) and that shortages are not backordered. The B_1 and B_2 costs are due to eliminating the shortages by emergency actions (customary in a project management context). Thus, the period-ending inventory can never go negative. A similar procurement problem for a single period (N=1), where X_1 and C_1 are continuous instead of discrete, has been studied recently by Jain and Silver [7, 10]. It turned out that the replenishment quantity to request can be selected easily contrary to the reserved capacity. In general, the expected cost as a function of the reserved capacity has multiple local minima. However, for the special case of normally distributed demand, normally distributed capacity and a linear function ϕ, an efficient algorithm has been developed for finding the best level of reserved capacity.

Let $COST_i(I_i, q_i, r_i, X_i, C_i)$ be the cost in period i as a function of the five quantities I_i, q_i, r_i, X_i, C_i where the latter two are random variables. The q_i values to consider are all the possible values of $X_i + X_{i+1} + ... + X_N - I_i$ but no higher than the largest possible value of C_i and r_i ($0 \le q_i \le \max(c_i^{nc_i}, r_i^{nr_i})$). Because of the discrete nature of the possible r_i's and the increasing nature of the cost function $\phi(r)$ it makes sense to only consider r_i values up to and including the first value equal to or greater than q_i. Except for the particular case of $q_i \le r_i$ where the inventory manager automatically receives q_i resource units, there are 6 different expressions for $COST_i$. To make things clearer, indices refering to the period will be dropped when they are not essential. $\delta(x)$ is a binary function equal to 1 if x>0 and 0 otherwise.

If r≤q, then

$$\mathrm{COST}(I,q,r,X,C) = \phi(r) + \begin{cases} A \cdot \delta(q) + q \cdot v + h \cdot (I + q - X) & \text{if } C \ge r,\ q \le C,\ X \le q + I \\ A \cdot \delta(q) + q \cdot v + B_1 + B_2 \cdot (X - I - q) & \text{if } C \ge r,\ q \le C,\ X > q + I \\ A \cdot \delta(C) + C \cdot v + h \cdot (I + C - X) & \text{if } C \ge r,\ q > C,\ X \le C + I \\ A \cdot \delta(C) + C \cdot v + B_1 + B_2 \cdot (X - I - C) & \text{if } C \ge r,\ q > C,\ X > C + I \\ A \cdot \delta(r) + r \cdot v + h \cdot (I + r - X) & \text{if } C < r,\ X \le r + I \\ A \cdot \delta(r) + r \cdot v + B_1 + B_2 \cdot (X - I - r) & \text{if } C < r,\ X > r + I \end{cases}$$

whereas, if q≤r, we have

$$\mathrm{COST}(I,q,r,X,C) = \phi(r) + A \cdot \delta(q) + \begin{cases} q \cdot v + h \cdot (I + q - X) & \text{if } X \le q + I \\ q \cdot v + B_1 + B_2 \cdot (X - I - q) & \text{if } X > q + I \end{cases}$$

We give below the expected value (over X and C) of COST(I, q, r) as a function of the inventory level I, the order quantity q and the dedicated capacity r.

$$\mathrm{ECOST}(I, q, r) = \sum_X \sum_C \mathrm{COST}(I, q, r, X, C) \cdot p(X) \cdot g(C)$$

If $r \le q$, then

$$
\begin{aligned}
ECOST(I, q, r) = \phi(r) &+ \sum_{C \ge q} \sum_{X \le I+q} [A \cdot \delta(q) + q \cdot v + h \cdot (I+q-X)] \cdot p(X) \cdot g(C) \\
&+ \sum_{C \ge q} \sum_{X > I+q} [A \cdot \delta(q) + q \cdot v + B_1 + B_2 \cdot (X-I-q)] \cdot p(X) \cdot g(C) \\
&+ \sum_{r \le C < q} \sum_{X \le I+C} [A \cdot \delta(C) + C \cdot v + h \cdot (I+C-X)] \cdot p(X) \cdot g(C) \\
&+ \sum_{r \le C < q} \sum_{X > I+C} [A \cdot \delta(C) + C \cdot v + B_1 + B_2 \cdot (X-I-C)] \cdot p(X) \cdot g(C) \\
&+ \sum_{C < r} \sum_{X \le I+r} [A \cdot \delta(r) + r \cdot v + h \cdot (I+r-X)] \cdot p(X) \cdot g(C) \\
&+ \sum_{C < r} \sum_{X > I+r} [A \cdot \delta(r) + r \cdot v + B_1 + B_2 \cdot (X-I-r)] \cdot p(X) \cdot g(C)
\end{aligned}
$$

$$
\begin{aligned}
= \phi(r) &+ \sum_{C < r} g(C) \cdot \Big\{ A \cdot \delta(r) + r \cdot v + \sum_{X \le I+r} h \cdot (I+r-X) \cdot p(X) \\
&\qquad\qquad + \sum_{X > I+r} [B_1 + B_2 \cdot (X-I-r)] \cdot p(X) \Big\} \\
&+ \sum_{r \le C < q} g(C) \cdot \Big\{ A \cdot \delta(C) + C \cdot v + \sum_{X \le I+C} h \cdot (I+C-X) \cdot p(X) \\
&\qquad\qquad + \sum_{X > I+C} [B_1 + B_2 \cdot (X-I-C)] \cdot p(X) \Big\} \\
&+ \sum_{q \le C} g(C) \cdot \Big\{ A \cdot \delta(q) + q \cdot v + \sum_{X \le I+q} h \cdot (I+q-X) \cdot p(X) \\
&\qquad\qquad + \sum_{X > I+q} [B_1 + B_2 \cdot (X-I-q)] \cdot p(X) \Big\}
\end{aligned}
$$

whereas, for $q \le r$,

$$
\begin{aligned}
ECOST(I, q, r) = \phi(r) + A \cdot \delta(q) + q \cdot v + \Big\{ &\sum_{X \le I+q} h \cdot (I+q-X) \cdot p(X) \\
+ &\sum_{X > I+q} [B_1 + B_2 \cdot (X-I-q)] \cdot p(X) \Big\}
\end{aligned}
$$

Given a sequence $\underline{r}=(r_1, r_2, ..., r_N)$, let $f_{\underline{r},i}(I_i)$ be the expected cost of an optimal (over the q's) strategy $(r_i, r_{i+1}, ..., r_N, q_i^*(I_i), \underline{q}_{i+1}^*(\cdot), ..., \underline{q}_N^*(\cdot))$ for periods i, i+1, ..., N starting with an inventory I_i in period i. $f_{\underline{r},i}(I_i)$ can be computed recursively as follows:

$$
\boxed{
\begin{aligned}
f_{\underline{r},i}(I_i) = \min_{q_i} &\Big(ECOST_i(I_i, q_i, r_i) + Ef_{\underline{r},i+1}(I_{i+1}) \Big) \\
&\text{where } I_{i+1} = I_{i+1}(I_i, q_i, r_i, X_i, C_i) = \max\Big\{ I_i + \min[\max(C_i, r_i), q_i] - X_i, 0 \Big\} \\
&\qquad Ef_{\underline{r},i+1}(I_{i+1}) = \sum_{X_i} \sum_{C_i} f_{\underline{r},i+1}(I_{i+1}) \cdot p_i(X_i) \cdot g_i(C_i) \\
&\qquad f_{\underline{r},N+1}(I_{N+1}) = 0 \ \forall\ \underline{r}, \forall\ I_{N+1} \ge 0
\end{aligned}
}
$$

Given a dedicated capacity sequence $\underline{r}$, the above recursive relationship allows us to use dynamic programming to compute $f_{\underline{r},i}(I_i)\ \forall i,\ \forall I_i$. When the values $f_{\underline{r},i+1}(I_{i+1})$ are known $\forall I_{i+1}$ the complexity of the computation of $f_{\underline{r},i}(I_i)$ is $O(\max(c_i^{nc_i}, r_i)\cdot nx_i\cdot nc_i)$. By setting $N_x=\max_i (nx_i)$, $N_c=\max_i (nc_i)$, $N_q=\max_i (c_i^{nc_i}, r_i^{nr_i})$ and $N_I=\max_i |\{$all possible inventories I_i in period $i\}|$, we have that the computation of $f_{\underline{r},1}(I_1=0)$ has complexity $O(N\cdot N_I\cdot N_q\cdot N_x\cdot N_c)$. It is important to point out that the maximum number of inventories to consider at a period, N_I, is hard to evaluate. It depends essentially on the specific data set of the problem and cannot be bounded by a polynomial function of the data size. In spite of this, the cost $f_{\underline{r},1}(I_1=0)$ and the decisions $(\underline{q}_1^*(\cdot),\ \underline{q}_2^*(\cdot),\ ...,\ \underline{q}_N^*(\cdot))$ can be computed relatively easily for problem sizes equivalent to those encountered in practice. Unfortunately this is not the case for the value:

$$f_{\underline{r}^*,1}(I_1=0) = \min_{r_1, r_2, \ldots, r_N} f_{\underline{r},1}(0)$$

which is of interest in our problem. Let $R = \{\underline{r}=(r_1, r_2, ..., r_N) \mid r_i \in \mathcal{R}_i\ \forall i\}$ be the set of all dedicated capacity sequences. As there are nr_i possible values of r_i, the total number of combinations is $\prod_{i=1}^{N} nr_i$. This number increases very quickly with N and nr_i. An exhaustive enumeration to find the best sequence $\underline{r}$ requires an enormous computational effort even for a small problem. For instance, with N=10 and $nr_i=4\ \forall i$ there are more than a million possible combinations to consider. Thus, in the next section we focus on an implicit enumeration procedure in an effort to reduce the computational effort.

3. An exact procedure

An efficient enumeration procedure needs to be carefully structured so that only a small fraction of R is examined. The branch and bound technique (B&B) provides such a procedure (c.f. [6] for a general introduction).

Let us denote by $R_i(r_1^j, r_2^k, ..., r_i^\ell) = \{\underline{r}\in R \mid r_1=r_1^j, r_2=r_2^k, ..., r_i=r_i^\ell\}$ the subsets of R which define subproblems of the original problem in which the quantities to reserve in periods 1to i are fixed. By convention $R_0=R$.

Let B^* be the expected cost of the best strategy $\underline{r}^*$ found so far and let $LB_i(r_1^j, r_2^k, ..., r_i^\ell)$ be a lower bound on the expected cost of the best strategy in $R_i(r_1^j, r_2^k, ..., r_i^\ell)$. Initially we set $\underline{r}^* = (r_1^1, r_2^1, ..., r_N^1)$, i.e. the r_i's are all set at their lowest levels, and $B^* = f_{r_1^1, r_2^1, ..., r_N^1, 1}(0)$. The procedure starts with the subproblem $R_{N-1}(r_1^1, r_2^1, ..., r_{N-1}^1)$. Once $R_i(r_1^j, ..., r_h^k, ..., r_i^\ell)$ has been reached (i<N), three basic steps can be performed:

1) if i=N-1 then we set the incremental counter m to 1 and we examine sequences of the form $(r_1^j, r_2^k, ..., r_{N-1}^\ell, r_N^m)$ one after the other as long as $(m \le nr_N)$ and $LB_{N-1}(r_1^j, ..., r_h^k, ..., r_{N-1}^\ell) < B^*$. If a sequence $(r_1^j, r_2^k, ..., r_{N-1}^\ell, r_N^m)$ better than the current best one is found (i.e. $f_{r_1^j, ..., r_h^k, ..., r_{N-1}^\ell, r_N^m, 1}(0) < B^*$), then we update B^* and $\underline{r}^*$. As soon as $m = nr_N$ or $LB_{N-1}(r_1^j, ..., r_h^k, ..., r_{N-1}^\ell) = B^*$ we move backward to $R_h(r_1^j, ..., r_h^{k+1})$ where h is the largest index≤N-1 such that $k < nr_h$.

2) if i<N-1 and $LB_i(r_1^j, ..., r_h^k, ..., r_i^\ell) \ge B^*$, then there is no hope to find a strategy in $R_i(r_1^j, ..., r_h^k, ..., r_i^\ell)$ that is better than the current best one. As above we move backward to $R_h(r_1^j, ..., r_h^{k+1})$ where h is the largest index≤i such that $k < nr_h$.

3) if i<N-1 and $LB_i(r_1^j, ..., r_h^k, ..., r_i^\ell) < B^*$, then it is worth examining further the subproblem $R_i(r_1^j, ..., r_h^k, ..., r_i^\ell)$. We move forward to $R_{i+1}(r_1^j, ..., r_h^k, ..., r_i^\ell, r_{i+1}^1)$.

The procedure stops just before backtracking to the original problem $R_0=R$ (in step (1) or (2)) after having explored implicitly all possible dedicated capacities $\underline{r}$. A more formal description of the above branch and bound algorithm is given in Table 1.

<Insert Table 1 here>

3.1 Computation of the lower bound $LB_i(r_1, r_2, ..., r_i)$

The efficiency of a branch and bound algorithm depends essentially on the quality of the lower bound computed at each step of the enumeration process. Let us consider the two following formulations which can be seen as relaxations of the original problem:

$$(P_1)\quad y_i(I_i) = \min_{r_i, q_i} \left(ECOST_i(I_i, q_i, r_i) + Ey_{i+1}(I_{i+1})\right)$$
$$\text{where } I_{i+1} = I_{i+1}(I_i, q_i, r_i, X_i, C_i) = \max\left\{ I_i + \min[\max(C_i, r_i), q_i] - X_i, 0\right\}$$
$$Ey_{i+1}(I_{i+1}) = \sum_{X_i}\sum_{C_i} y_{i+1}(I_{i+1}) \cdot p_i(X_i) \cdot g_i(C_i)$$
$$y_{N+1}(I_{N+1}) = 0 \ \forall I_{N+1} \ge 0$$

$$
\boxed{\begin{aligned}
&\ell b_{r_1,r_2,\ldots,r_i,j}(I_j) = \min_{q_j} \Big(ECOST_j(I_j, q_j, r_j) + E\ell b_{r_1,r_2,\ldots,r_i,j+1}(I_{j+1})\Big) \\
&\quad \text{where } I_{j+1} = I_{j+1}(I_j,q_j,r_j,X_j,C_j) = \max\Big\{ I_j + \min[\max(C_j, r_j), q_j] - X_j, 0\Big\} \\
&\quad E\ell b_{r_1,r_2,\ldots,r_i,j+1}(I_{j+1}) = \sum_{X_j}\sum_{C_j} \ell b_{r_1,r_2,\ldots,r_i,j+1}(I_{j+1})\cdot p_i(X_j)\cdot g_i(C_j) \\
&\quad \ell b_{r_1,r_2,\ldots,r_i,i+1}(I_{i+1}) = y_{i+1}(I_{i+1}) \ \forall\, r_1, r_2, \ldots, r_i, \ \forall\, I_i \geq 0
\end{aligned}}
\qquad P_2 \ (\forall j \leq i)
$$

Problem P_1 is based on the assumption that r_i can be selected at the same time as q_i (i.e. at the start of period i once the inventory level I_i is known). It follows that $y_i(I_i) \leq f_{\underline{r},i}(I_i)\ \forall\, \underline{r}, \forall I_i$. In problem P_2, this assumption is not valid for periods $j \leq i$ and thus only $r_{i+1}, r_{i+2}, \ldots, r_N$ can be selected at the start of their corresponding periods. The values $y_i(I_i)$ and $\ell b_{r_1,r_2,\ldots,r_i,j}(I_j)$ can be computed quickly using dynamic programming. We have $y_j(I_j) \leq \ell b_{r_1,r_2,\ldots,r_i,j}(I_j) \leq f_{r_1,r_2,\ldots,r_N,j}(I_j)$ $\forall j \leq i, \forall I_j, \forall\, \underline{r} = (r_1, r_2, \ldots, r_N)$, and in particular $\ell b_{r_1,r_2,\ldots,r_i,1}(0) \leq f_{r_1,r_2,\ldots,r_N,1}(0)\ \forall\, \underline{r} = (r_1, r_2, \ldots, r_N)$. Thus a lower bound on the expected cost of the best strategy in $R_i(r_1, r_2, \ldots, r_i)$ is given by:

$$LB_i(r_1, r_2, \ldots, r_i) = \ell b_{r_1,r_2,\ldots,r_i,1}(0)$$

The efficiency of a branch and bound algorithm is directly related to the quality of the lower bound considered. $LB_i(r_1, r_2, \ldots, r_i)$ will be shown to discard a large number of dedicated capacity sequences. Because the bound from P_2 is sharper than that from P_1, we did not use P_1 for bounding purposes. However, P_1 will be used as part of one of the heuristic procedures in section 4.1.

3.2 Upper bound on r_i^*

If we can find an upper bound on some of the r_i^*'s, this could eliminate a significant portion of the $|R| = \prod_{i=1}^{N} nr_i$ possible patterns of $\underline{r}$ and thus substantially reduce the search effort. An upper bound on r_i^*, the best value of r_i, can be found as follows. Assume we are at the start of period i with zero inventory (i.e. $I_i=0$) and that capacity can be reserved in period i but not in periods i+1, i+2, ..., N (i.e. $r_{i+1} = r_{i+2} = \ldots = r_N = 0$). Under these circumstances, we can use dynamic programming to select r_i and $q_j(I_j)\ \forall j \geq i, \forall I_j$. Moreover, because of the strong restrictions on I_i and on the r_j's (j>i) the value of r_i determined in this way is clearly an upper bound on r_i^* of the original problem. This can be done for each i=N, N-1, ..., 1. Hereafter we will assume that the sets $\mathcal{R}_i$ do not contain any value greater than the upper bound on r_i^*.

As we saw at the end of the previous section, the total number of sequences $\underline{r}$ can get arbitrarily large very quickly. If $N_r = \max_i (nr_i)$ denotes the maximum number of r's in a period then the complexity bound of the algorithm is $O(N_r^N \cdot N \cdot N_I \cdot N_q \cdot N_x \cdot N_c)$. Due to this exponential complexity, even the fastest computers and the more sophisticated enumeration algorithms cannot guarantee obtaining an optimal solution to a problem of large size. In the remainder of this paper we thus concentrate on heuristics as opposed to exact methods. In practice, it is generallly sufficient to look for a good solution, instead of an optimal one, if the latter can only be found after an enormous computational effort.

4. Heuristic approaches

We present in this section 4 different heuristic methods for finding good strategies to the procurement problem considered in this paper.

4.1 An incomplete enumeration method

The idea of this first heuristic is to substantially restrict the sets $\mathcal{R}_i$ of possible dedicated capacities in each period i in order to lessen the running time of the branch and bound algorithm when applied on the resulting reduced set $R^- \subset R = \{\underline{r} = (r_1, r_2, ..., r_N) \mid r_i \in \mathcal{R}_i \ \forall i\}$. To that purpose, we solve problem P_1 ($i=1$, $I_1=0$) where each r_i can be established at the start of period i and we keep track of all values of r_i that are chosen. Let $\mathcal{R}_i^- \subset \mathcal{R}_i$ be the set of these values chosen in period i. Table 2 provides a description of the resulting incomplete enumeration (IE) method. Clearly, this method yields an optimal strategy if and only if R^- contains the optimal $\underline{r}$.

<Insert Table 2 here>

4.2 A constructive algorithm

This second heuristic builds a strategy $(\underline{r}^*, \underline{q}^*)$ gradually by considering one period at a time. To establish reasonable values of r_1 and $q_1(I_1=0)$ one possibility consists of looking at a shorter horizon, e.g. n periods (n<N) where n is a parameter. Then we go through all possible patterns of $(r_1, r_2, ..., r_n)$ in order to establish components r_1^* and $q_1^*(I_1=0)$ of the best strategy $(r_1^*, r_2^*, ..., r_n^*, q_1^*(0), \underline{q}_2^*(\cdot), ..., \underline{q}_n^*(\cdot))$ obtained. When looking for the best strategy of the form $(r_1, r_2, ..., r_n, q_1(0), \underline{q}_2(\cdot), ..., \underline{q}_n(\cdot))$, it is important to take account of future difficulties caused by not reserving enough capacity in periods 1 through n. Experiments were carried out with a number of possible methods to adequately anticipate such future costly capacity shortfalls. Such methods should be reasonably efficient from a computational standpoint. The best results were obtained by considering a pseudo-project starting in period n+1 with, as initial inventory I_{n+1},

the inventory left at the end of period n. The qualifier "pseudo" is used because we assume that capacity can be reserved at the start of each of periods n+1, n+2, ..., N (cf problem P_1 in section 3.1) which, of course, is not the case in the true project context. The cost function used for choosing the best strategy $(r_1^*, r_2^*, ..., r_n^*, q_1^*(0), \underline{q}_2^*(\cdot), ..., \underline{q}_n^*(\cdot))$ is the sum of the expected cost over periods 1 through n and the expected cost of the pseudo-project. Another way of interpreting what we are doing is that we select r_1 and q_1 so as to minimize the expected total cost over periods 1through N assuming that the r's for periods 1, ..., n must be all selected ahead of time whereas the r's for periods n+1, ..., N can be selected at the start of their corresponding periods.

In general we can look at just periods i, i+1, ..., i+n-1 and use the above principle to get r_i^* and $\underline{q}_i^*(I_i)$. But the difficulty is what to assume about the inventory level at the start of period i (i>1). In contrast with I_1, which is known, I_i for i>1 is a random variable in that its value depends upon the random variables $C_1, C_2, ...,C_{i-1}, X_1, X_2, ...$ and X_{i-1}. One approach is to recursively compute a probability function $u_i(I_i)$ for all possible inventories I_i. Initially we have:

$$u_1(I_1) = \begin{cases} 1 \text{ if } I_1=0 \\ 0 \text{ otherwise} \end{cases}$$

Assuming $r_{i-1}^*, q_{i-1}^*(I_{i-1})$ and $u_{i-1}(I_{i-1})$ are known $\forall I_{i-1}$ we obtain $u_i(I_i)$ as follows:

$$u_i(I_i=inv) = \sum_{(X_{i-1},C_{i-1},I_{i-1})\in S_{i-1}(inv)} p_{i-1}(X_{i-1})\cdot g_{i-1}(C_{i-1})\cdot u_{i-1}(I_{i-1})$$

where $S_{i-1}(inv) = \Big\{ (X_{i-1},C_{i-1},I_{i-1}) \;\Big|\; (u_{i-1}(I_{i-1})>0) \text{ and } (X_{i-1} \in \mathcal{X}_{i-1}) \text{ and } (C_{i-1} \in \mathbb{C}_{i-1}) \text{ and }$
$(inv=\max\{I_{i-1}+\min[\max(C_{i-1}, r_{i-1}^*), q_{i-1}^*(I_{i-1})] - X_{i-1}, 0\}) \Big\}$

The computation of the probability function $u_i(I_i)$ is a very difficult task. Thus, we consider instead weights $w_i(I_i)$ which are approximate values of the $u_i(I_i)$'s obtained by simulation. The $w_i(I_i)$'s can be computed recursively by generating in each period i-1 (i>1) a given number *nsimul* (e.g. ngen ≈ 100) of pairs (X_{i-1},C_{i-1}). Details are given in Table 3.

<Insert Table 3 here>

Suppose that now we were to compute for each possible inventory level I_i and for each $\tilde{r}_i \in \mathcal{R}_i$, the expected cost $\tilde{f}_{i,i+n-1}(I_i, \tilde{r}_i)$ of an optimal strategy of the form $(\tilde{r}_i, r_{i+1}, ..., r_{i+n-1}, q_i(I_i), \underline{q}_{i+1}(\cdot), ..., \underline{q}_{i+n-1}(\cdot))$ over periods i, i+1, ..., i+n-1. Clearly, it is very unlikely to have the same r_i

minimizing $\tilde{f}_{i,i+n-1}(I_i, r_i)$ for every inventory I_i. To reflect the different possible values of I_i we instead consider the weighted function:

$$\tilde{F}_{i,i+n-1}(\tilde{r}_i) = \sum_{(I_i \mid w_i(I_i)>0)} w_i(I_i) \cdot \tilde{f}_{i,i+n-1}(I_i, \tilde{r}_i)$$

and we keep track of the component r_i^* minimizing $\tilde{F}_{i,i+n-1}(\tilde{r}_i)$ over $\mathbf{R}_i$ as well as its associated order quantity vector $\underline{q}^*_i(\cdot)$.

The quality of the strategy $(\underline{r}^*, \underline{q}^*)$ produced in this way depends on the parameter n ($1 \le n < N$). Let us denote by CA(n) the constructive algorithm presented above.

As we are dealing with short horizons in CA(n) ($n<N$), the possibility of reserving capacity at each step of the algorithm is potentially not as attractive as in the original N period project. The first experiments with CA(n) showed that it is worth lowering somewhat the cost of reserving capacity in order to make the possibility of reserving capacity more attractive. The percentage reductions in the cost of reserving that we tested are 10, 20, 25 and 33. The best results were obtained by considering a 25 percent reduction. Thus, we decided to reduce $\phi_k(r_k^j)$ by 25 percent ($k=i, i+1, ..., i+n-1$; $j=1,..., nr_k$) at each step i of CA(n).

A general description of CA(n) is provided in Table 4. The complexity of CA(n) is $O(N_r^n \cdot N^2 \cdot N_I \cdot N_q \cdot N_x \cdot N_c)$.

<Insert Table 4 here>

4.3 Neighborhood search methods

The algorithms described in this section are refinements of the well known classical descent method. Roughly speaking the method consists of starting from an initial sequence $\underline{r} \in R$ and then moving repeatedly to a neighbor sequence that is as good as possible. In order to direct the search rationally in R, the move from a sequence $\underline{r}$ to the next one $\underline{r}'$ is restricted to a class of simple modifications. Let $M^s_{j,k}$ be the set of moves that consist of modifying at most s components r_i of a sequence $\underline{r}$ with i ranging from j to k. In the same fashion we define the neighborhood $N^s_{j,k}(\underline{r})$ of $\underline{r}$ as being the set of all sequences $\underline{r}' = \underline{r} \oplus m \in R$ that can be obtained from $\underline{r}$ by applying a move $m \in M^s_{j,k}$.

The most straightforward implementation of this search principle is to scan at each step the complete neighborhood $N^s_{j,k}(\underline{r})$ of the current $\underline{r}$ and to move to the best sequence encountered. One possible stopping criterion is to interrupt the search if no $\underline{r}'$ better than $\underline{r}$ can be found in $N^s_{j,k}(\underline{r})$. Unfortunately such a method is likely to lead to a local minimum of $f_{\underline{r},1}(0)$ which may be far from the global minimum. To circumvent such an outcome, non improving moves must also be considered when selecting a neighbor sequence. But when a sequence worse than $\underline{r}$ can be selected, cycling can occur, causing the process to be trapped by returning repeatedly to the same sequence. Numerous neighbor selection rules can be designated to reduce considerably the risk of cycling after a local minimum is detected. Two adaptations of this general search principle to the procurement problem are presented hereafter.

Different types of local searches can be obtained by varying the parameters s, j, and k when defining the neighborhood of a given sequence. Let us first restrict to the case where s=1, j=1 and k=N. This means that only sequences $\underline{r}'$ which differ from the current sequence $\underline{r}$ by at most one component can be selected at each step of the search process.

Tabu search (TS) is a general neighborhood search technique which provides guidance through the solution space by incorporating a memory structure forbidding moves that could lead the search to a recently visited solution. This is done by a constantly updated tabu list T which stores some attributes of the most recently moves. When moving from $\underline{r}' = (r_1, ..., r_i', ..., r_N)$ to $\underline{r}'' = (r_1, ..., r_i'', ..., r_N)$ the most natural thing to do is to memorize (i, r_i') in T and to forbid the ith component of $\underline{r}$ to be again r_i' during the next $|T|$ iterations. Clearly such a tabu list T may hinder us from visiting a sequence which has never been reached before. Thus it must be possible to cancel the tabu status of a move under certain favourable circumstances. We will do so if a tabu move from $\underline{r}$ to $\underline{r}'$ yields a sequence $\underline{r}'$ better than the best sequence $\underline{r}^*$ found so far (i.e. if $f_{\underline{r}',1}(0) < f_{\underline{r}^*,1}(0)$). The whole process is stopped as soon as a given number *nimax* of consecutive iterations have been performed without improving the best strategy obtained so far.

Dynamic programming has to be used to compute $f_{\underline{r},1}(0)$. Thus it is very time-consuming to explore the whole neighborhood of the current solution at each step of the process. We propose two techniques to speed up the search for a neighbor sequence at each iteration of TS. We examine only a randomly selected sample $V^* \subset N^1_{1,N}(\underline{r})$ of size $|V^*| = \varepsilon \cdot |N^1_{1,N}(\underline{r})|$ $(0<\varepsilon<1)$ and we rank the best *nmv* moves encountered in expected cost decreasing order with the aim of performing more than one move without a complete reevaluation. It is important to point out that each time one of the moves selected is performed the expected cost of the remaining moves change. We decided to perform the best move automatically and to accept the other moves if they are still improving moves according to their updated expected costs.

Let TS(ε,nmv) denote a TS algorithm in which a proportion ε of the current neighborhood is explored and at most *nmv* nmoves are performed at each step. One iteration of TS(ε,nmv) has complexity $O(N_r \cdot N^2 \cdot N_I \cdot N_q \cdot N_x \cdot N_c)$. An adaptation of TS($\varepsilon$,nmv) to the procurement problem is given in Table 5. Detailed explanations on how TS works, as well as some refinements of the simple version of TS presented above, can be found in [2, 3, 5].

<Insert Table 5 here>

The use of the best move criterion in a neighborhood search technique is based on the supposition that moves with higher evaluations have a higher probability of leading to an optimal (or near optimal) solution in a fewer number of steps. However, as observed in [1], the use of a candidate list of moves often permits neighborhood search techniques significantly superior in overall efficiency (in spite of requiring more iterations), especially when the available moves at each step of the search process are many and costly to evaluate. Generally the basis of a candidate list strategy is to decompose the set of all possible moves according to the structure of the problem under study. Rather than looking repeatedly for the best sequence in $N^{s}_{1,N}(\underline{r})$ (or in a randomly generated sample $V^{*} \subset N^{s}_{1,N}(\underline{r})$) a good idea is to subdivide $M^{s}_{1,N}$ into a given number of subsets $M^{s}_{j,k}$ and to seek a best move from each of these subsets considered in turn. In order to diversify somewhat the search through R, s different decompositons of $M^{s}_{1,N}$ are taken into account successively, namely $(M^{s}_{incr+1-s,incr}, M^{s}_{incr+1,incr+s}, \ldots, M^{s}_{incr+1+(m-1)\cdot s,N})$ where *incr* is an incremental parameter ranging from 0 to s-1 and $m=\lceil \frac{N-incr}{s} \rceil$. By convention $M^{s}_{j,i} = M^{s}_{1,i}$, $M^{s}_{i,k} = M^{s}_{i,N}$ and $M^{s}_{j,0} = \emptyset\ \forall 1 \le i \le N, \forall j<1, \forall k>N$. As we are dealing with s consecutive neighborhoods which are always pairwise disjoint, the use of a list of forbidden moves is no longer essential. We interrupt the procedure after having examined all sets of moves of a decomposition without improving on the best strategy obtained so far. Each decomposition has a complexity $O(N^{s}_{r} \cdot N^{2} \cdot N_{I} \cdot N_{q} \cdot N_{X} \cdot N_{c})$. Details of the above candidate list strategy are given in Table 6.

<Insert Table 6 here>

5. Computational experiments

The experiments were conducted on a Silicon Graphics workstation (9 Mflops). Various classes of problems were generated to provide a basis for comparison between the search methods proposed in this paper. Unfortunately, due to the large number of parameters, namely v_i, A_i, $B_{1,i}$, $B_{2,i}$, h_i, s, $p_i(X_i)$, $g_i C_i)$ and $\phi_i(r_i)$, it was not possible to do a complete experimental design testing of the methods developed. The assumptions we made are given below:

- the unit acquisition cost v_i is equal to 1 in each period i (v_i=v=1 $\forall$i).
- the requirements X_i and the supplier capacities C_i are multiples of ten between 10 and 100 which have binomial distributions with parameters $(9,\alpha_i)$ and $(9,\beta_i)$ respectively.

More formally, we have:

$\mathcal{X}_i = \{10, 20, 30, ..., 100\}$, $p_i(X_i=x_i)= \tilde{p}_i(y_i= \frac{x_i-10}{10})= \binom{9}{y_i}\cdot\alpha_i^{y_i}\cdot(1-\alpha_i)^{9-y_i}$ and

$\mathcal{C}_i = \{10, 20, 30, ..., 100\}$, $g_i(C_i=c_i)= \tilde{g}_i(z_i= \frac{c_i-10}{10})= \binom{9}{z_i}\cdot\beta_i^{z_i}\cdot(1-\beta_i)^{9-z_i}$.

In order to avoid manipulating quantities which occur rarely, we grouped the lowest and greatest x_i (resp. c_i) values such that $p_i(X_i=x_i)<0.02$ (resp. $g_i(C_i=c_i)<0.02$). For instance, if $p_i(X_i=90)=0.05$ and $p_i(X_i=100)=0.01$, we discarded the order quantity $X_i=100$ and we set $p_i(X_i=90)=0.06$. It follows that the numbers nx_i and nc_i depend exclusively on the probabilities α_i and β_i selected ($nx_i \leq 9$, $nc_i \leq 9\ \forall i$).

- the nr_i possible dedicated capacities in each period i are multiples of 20, i.e. $\mathcal{R}_i = \{0, 20, 40, ..., (nr_i-1)\cdot 20\}$. We used $nr_i=nr=4$, 5 or 6 $\forall i$. The function $\phi_i(r_i)$, giving the cost of reserving capacity r_i in period i, was assumed to be piecewise linear as shown in Figure 1. It can be built recursively for any set $\mathcal{R}_i$ by generating successive increasing slopes. Let t_j be the slope of the straight line from $\phi_i(r_i^{j-1})$ to $\phi_i(r_i^j)$. Initially we have $\phi_i(r_i^1=0)=0$ and $\phi_i(r_i^2)= t_1\cdot r_i^2$. The selection of t_1 will be described in a moment. Then we set $\phi_i(r_i^j)=\phi_i(r_i^{j-1}) + t_j\cdot(r_i^j-r_i^{j-1})$ with $t_j=t_{j-1}\cdot m_j\ \forall j=3, ..., nr_i$. The m_j's were independently drawn from a uniform distribution over the interval (1,1.2). In order to make the possibility of reserving capacity attractive, it is important to have $\phi_i(r_i^j)<B_{2,i}\cdot r_i^j\ \forall j=2,..., nr_i$. This inequality is automatically satisfied if $t_1< \frac{B_{2,i}}{(1.2)^{nr_i-1}}$. We set $t_1=(\bar{C}_i)^{-1/2}\cdot\frac{B_{2,i}}{(1.2)^{nr_i-1}}$ where $\bar{C}_i$ is the average supplier capacity in period i. Thus the unit costs for reserving capacity are inversely proportional to the average supplier capacity, i.e. it is less expensive to reserve a specific amount of capacity if it is a smaller proportion of the supplier's average available capacity.

<Insert Figure 1 here>

The other parameter values in our model were set as follows:

$h_i=h$ = 0.015 and 0.030 $\forall i=1,...,N-1$ (based on annual rates of 0.18 and 0.36 if periods refer to months).

s = 0 and 0.2.

A_i : two values to give $T_{EOQ,i} = \sqrt{\frac{2\cdot A_i}{\bar{X}_i\cdot h}} = 0.5$ and 1, where $T_{EOQ,i}$ is the economic order quantity expressed as a time supply (see section 5.2 in [8]) and $\bar{X}_i$ is the average demand in period i.

$B_{1,i}$ = 0 and $\frac{\bar{X}_i}{2}$. The idea of the higher value is to have the fixed cost of a stockout equal to half of the average cost of acquisition of one period's requirement.

$B_{2,i}=B_2$: two values to give probabilities 0.014 and 0.010 of a stockout in the standard single period inventory model where (see section 10.2 in [8]) probability of a stockout = $\frac{h}{B_2+h}$.

X_i, C_i : three cases were taken into account. The abbreviation $X \sim bin(\nu,\alpha)$ means that the random variable X has a binomial distribution with parameters (ν,α).

a) The $p_i(X_i)$'s (resp. the $g_i(C_i)$'s) are all the same: $X_i \sim bin(9,\alpha=0.40)$ and $C_i \sim bin(9,\beta=0.35)$.

b) $X_i \sim bin(9,\alpha_i)$ and $C_i \sim bin(9,\beta_i)$ where α_i and β_i are randomly generated between 0.2 and 0.8 in each period i.

c) There is a seasonal pattern for the X_i's and the C_i's. A season is assumed to be made up of 3 consecutive periods. When demands are highest, available capacities are likely to be the lowest, because of other customers having the same seasonal requirements. For each period i in season S (S=1, ..., $\lceil N/3 \rceil$), we have $X_i \sim bin(9,\alpha_S)$ and $C_i \sim bin(9,\beta_S=1-\alpha_S)$ where α_S is generated randomly between 0.2 and 0.8.

Given a number N of periods and a number nr of possible reserved supplier capacities in each period, we defined a base problem with the following parameter values: h=0.015, s=0, $T_{EOQ,i}=0.5$, $B_{1,i}=\frac{\overline{X_i}}{2}$, probability of a stockout= 0.014, $X_i \sim bin(9,0.40)$ and $C_i \sim bin(9,0.35)$. Then we looked at variations of the base problem by changing each of the parameter values, one at a time. This defined a set of 8 problems which served as test problems in the computational experiments.

Let us first consider the base problem obtained by setting N=2 and nr=6. The optimal strategy turned out to be:

$\underline{r}^* = (r_1^*, r_2^*) = (60, 0)$

i=1	I_1	0							
	$f_{\underline{r}^*,1}(I_1)$	112.41							
	$q_1^*(I_1)$	70							
i=2	I_2	0	10	20	30	40	50	60	70	80
	$f_{\underline{r}^*,2}(I_2)$	77.91	65.15	49.41	30.98	20.98	10.88	6.47	3.56	2.61
	$q_2^*(I_2)$	50	40	30	20	10	0	0	0	0

The expected costs $f_{\underline{r},1}$ (0) for all of the 6^2 possible $\underline{r} = (r_1, r_2)$ vectors are plotted in Figure 2. The colour changes according to the height of the point represented (110-115: dark grey, 115-120: pale grey, 120-125: black, 125-130: white). We see that the function contains four local minima, namely (0,0), (0,60), (60,0) and (60,60). These local minima render the appropriate search for an optimal sequence $\underline{r}^*$ far from obvious. We can easily imagine that the shape of $f_{\underline{r},1}$ (0) becomes even more complex for larger values of N and nr. Other plots like Figure 2 have shown that a single modification in the parameter setting is likely to radically change the form of the expected cost function.

<Insert Figure 2 here>

Let us mention that the value $y_1(I_1{=}0)$ (cf problem P_1 in section 3.1) is equal to 108.76 for the above base problem. The corresponding strategy is presented below. The values $\hat{q}_i(I_i)$ and $\hat{r}_i(I_i)$ denote respectively the replenishment quantity and the capacity reservation which are communicated to the supplier at the start of period i with a stock level equal to I_i (i=1,2). Note that the approximation of problem P_1 gives the optimal values of r_1 and q_1 in this particular instance.

i=1	I_1	0								
	$y_1(I_1)$	108.76								
	$\hat{q}_1(I_1)$	70								
	$\hat{r}_1(I_1)$	60								
i=2	I_2	0	10	20	30	40	50	60	70	80
	$y_2(I_2)$	66.64	54.64	44.64	30.98	20.98	10.88	6.47	3.56	2.61
	$\hat{q}_2(I_2)$	50	40	30	20	10	0	0	0	0
	$\hat{r}_2(I_2)$	60	40	40	0	0	0	0	0	0

All subsequent results are averages obtained by running the algorithms on the associated set of 8 test problems. The first experiments deal with the branch and bound algorithm with N ranging from 6 to 12 and nr from 4 to 5. The expected cost, the CPU time (in seconds) as well as the percent of subproblems $R_i(\cdot)$ (pc_sp) and of sequences $\underline{r}$ (pc_r) examined by the enumeration process are shown in Table 7.

<Insert Table 7 here>

For given N and nr values, we observed important differences in the CPU times of B&B, even though two problems differ by only one change in the parameter setting. Some problems turned out to be very easy to solve whereas others required much more effort.

We see that for each N the optimal expected cost $f_{\underline{r}*,1}(0)$ decreases when nr goes from 4 to 5. It follows that it is generally worth reserving capacity r_i^5=80 at least once in periods 1, ..., N.

The proportions of subproblems $R_i(\cdot)$ examined by the algorithm are extremely low. With only a few exceptions these proportions decrease when the problem size increases. This means that the efficiency of the lower bound LB_i (r_1, r_2, ..., r_i) is somewhat proportional to the problem size. We see, for instance, that up to 100-0.18 or 99.82% of the sequences $\underline{r}$ are discarded on average when N=12 and nr=5.

Relatively large problems can be solved by implicitly exploring all possible sequences $\underline{r}$. However, it is important to point out that even a modest increase in problem size can cause an explosion in computational time beyond feasible limits. For N=11 and N=12 we see that the running time becomes approximately 10 times bigger when nr increases from 4 to 5. Larger problems cannot be solved in a reasonable amount of CPU time.

As mentioned in the introduction, it is generally sufficient to find near optimal strategies in practice. What is equally important is the running time necessary to produce them. Let us now examine the efficiency of the various heuristics proposed in section 4 from solution quality and computational viewpoints. The CA(n), TS(ε,nmv) and CLS(s) heuristics were each run with 3 parameter values, namely {n=1,2,3}, {(ε,nmv)=(0.5,1),(1,1),(1,$\lceil N/4 \rceil$)} and {s=1,2,3}. Twelve values of N ranging between 6 and 40 were considered. In order to limit the number of tests, we only examined the case where nr=5, i.e. $\mathcal{R}_i$ = {0, 20, 40, 60, 80} ∀i. The average expected costs and CPU times (in seconds) measured in our experiments are reported in Tables 8-11. When the optimal strategies were known (i.e. when N≤12) *success* gives the number of times (out of 8) that the heuristic considered found the optimal strategy. OLP refers to the "on line procedure" which solves the procurement problem by assuming that capacity can be reserved at the start of each period once the stock level is known (cf problem P_1 in section 3.1). The resulting expected cost is clearly a lower bound on $f_{\underline{r}*,1}$ (0).

<Insert Tables 8-11 here>

- Among all the heuristics tested, IE is undeniably the method which produces the strategies with lowest expected cost. In all the experiments where $f_{\underline{r}*,1}$ (0) is known, IE finds the optimal strategy. Although IE is much faster than B&B, its running time is very high when compared with the other heuristics. The proportions of subproblems $R_i(\cdot)$ and of sequences $\underline{r}$ explored by IE are not reported in Tables 8-9. These proportions are bigger for IE than for B&B but they remain extremely low. They also decrease when the problem size increases.

- Contrary to our expectations, the quality of the solutions produced by CA(n) often decreases when the parameter n increases. For n larger than 1, what happens in periods i+1,i+2, ..., i+n-1 apparently has a detrimental influence on the choice of r_i and $\underline{q}_i(\cdot)$. Thus, it is generally not worth using CA(2) and CA(3) because they require more computational time than CA(1) without guaranteeing better results. Hence, when building the strategy ($\underline{r}$, $\underline{q}$) periodically it seems more appropriate to restrict the horizon to n=1 period.

  In order to make the possibility of reserving capacity more attractive (cf end of section 4.2), the cost of reserving capacity is reduced by 25 percent in periods i, i+1,..., i+n-1 at each step i of CA(n). However, it is to be observed that the performance of CA(1) is still quite good even without the cost reduction.

  In comparison with the other heuristics, we observe that CA(n) produces better results for small problems than for large ones. The generation of the weights $w_i(I_i)$ is based on $\underline{w}_{i-1}(\cdot)$, r_{i-1} and $\underline{q}_{i-1}(\cdot)$. Thus, once a wrong decision is made at one step of CA(n), it is unlikely to compute the subsequent weights properly. Hence the importance given to the future inventories may not reflect the real situation. The larger N is, the more chances we have to end up with erroneous estimations of the weights $w_i(I_i)$. One possible improvement of CA(n) would be to build the strategy ($\underline{r}$, $\underline{q}$) by considering more than one period at a time, i.e. we select r_i, ..., r_{i+k-1}, $\underline{q}_i(\cdot)$, ..., $\underline{q}_{i+k-1}(\cdot)$ ($1<k\leq n$) instead of just r_i and $\underline{q}_i(\cdot)$ at each step of the algorithm. Further research is being carried out along this line.

- TS(0.5,1) is much faster than TS(1,1) but it yields strategies which are significantly worse than TS(1,1). The principle of looking repeatedly for the best neighbor strategy seems to lead to better final solutions. On the other hand TS(1,$\lceil N/4 \rceil$) is comparable to TS(1,1) both in solution quality and in computational time. Surprisingly TS(1,$\lceil N/4 \rceil$) requires slightly more CPU time than TS(1,1) does for $N\leq 18$.

- Contrary to CA, CLS improves on solution quality when its parameter value increases. Like IE, CLS(2) and CLS(3) find the optimal strategy $\underline{r}^*$ everytime it was known. CLS(3) produces the best strategies for N between 14 and 30.

Neighborhood search techniques have already been applied to a broad variety of combinatorial optimization problems. They define a general framework in computer programming that can challenge very sophisticated algorithms. In our case, however, TS and CLS yield solutions which are quite acceptable but they are not really competitive with respect to the running time. The difficulty with the procurement problem is that it is extremely time-consuming to explore the whole neighborhood at each step of the search. With TS, the principles of generating only part of the neighborhood and/or performing more than one move at each step yield solutions which are not significantly better.

The comparison between the various results reported in Tables 8-11 reveals that there is no algorithm which always outperforms the others with respect to solution quality and computational time. There is somewhat of a tradeoff between computational effort and the quality of the solution. Our recommendation is to use either IE or CLS(3) for problems of small size ($N \leq 16$ and $nr \leq 5$) and CA(1) for larger problems, although CLS(1) could be used in place of CA(1) if computational effort is not a primary concern.

6. Conclusion

In this study we have developed one exact and four approximate algorithms for the multiperiod procurement problem where there is uncertainty in demand and in supplier capacity and where dedicated capacity can be reserved ahead of time.

The exact algorithm is governed by a branch and bound procedure based on dynamic programming. Due to its exponential complexity, the algorithm is applicable only to problems of relatively small size. Its CPU time becomes prohibitive when treating problems with more than $N=12$ periods and $nr=5$ possible dedicated capacities in each period.

The four heuristic methods proposed in this study are based on three different search principles: an incomplete enumeration (IE), a constructive schema (CA) and a neighborhood search (TS and CLS). A wide variety of solutions are produced by the heuristics. However none of the methods appears to be better than the others from both solution quality and computational viewpoints. IE, CLS(s) ($2 \leq s \leq 3$) and TS(1,·) produce solutions with lowest expected cost but CA(1), TS(0.5,1) and CLS(1) are much faster. In our opinion CA(1) and CLS(1) are good compromises between solution quality and computational time for large size real-world procurement problems.

Acknowledgements

The authors would like to thank Fred Glover for having provided copies of helpful research papers, Thomas Liebling for the german translation of the abstract and Yves Mottet for the drawing of Figure 2. The research leading to this paper was partially supported by the Natural Sciences and Engineering Research Council of Canada under Grant A1485 and by the Carma Chair in Operations Management at the University of Calgary. A substantial portion of the work was carried out while Edward Silver held a visiting research professorship at the Ecole Polytechnique Fédérale de Lausanne.

References

[1] Glover F., "Simple Tabu Thresholding in Optimization", Research Paper, Graduate School of Business, University of Boulder, 1992.

[2] Glover F., Taillard E., de Werra D. "A user's guide to tabu search", Annals of Operations Research 41 (1993), 3-28.

[3] Glover F., Laguna M. "Tabu search", Colin R. Reeves ed., Modern Heuristic Techniques for Combinatorial Optimization, Blackwell Scientific Publications, Oxford, 1993.

[4] Graves S.C., Rinnooy Kan A.H., Zipkin P.H. Logistics of production and inventory, Handbooks in Operations Research and Management Science, Volume 4, North-Holland, 1993.

[5] Hertz A., Taillard E., de Werra D. "Tabu search", J.K. Lenstra ed., Local Search in Combinatorial Optimization, to appear.

[6] Hillier F.S., Lieberman G.J. Introduction to Operations Research, Fifth Edition, McGraw-Hill, 1990.

[7] Jain K., Silver E.A. "The single period procurement problem where dedicated supplier capacity can be reserved", Naval Research Logistics, to appear.

[8] Silver E.A., Peterson R. Decision systems for inventory management and production planning, Second Edition, Wiley, New-York, 1985.

[9] Silver E.A. "Policy and procedural issues in procurement and logistics for large scale projects in the oil and gas industry", Project Management Journal 18 (1987), 57-62.

[10] Silver E.A., Jain K. "Some ideas regarding reserving supplier capacity and selecting replenishment quantities in the project context", International Journal of Production Economics 35 (1994), 177-182.

[11] Silver E.A., Costa D. "Dealing with a variable supplier capacity", Research Report 94/05, DMA, Ecole Polytechnique Fédérale de Lausanne, 1994.

Table 1 - A branch and bound algorithm (B&B) for the procurement problem

Variables

B^*	expected cost of the best strategy found so far
$\underline{r}^*$	best strategy found so far
index	array defining the current sequence $\underline{r} = (r_1^{index[1]}, r_2^{index[2]}, ..., r_N^{index[N]})$
forward	boolean variable indicating if the enumeration process goes forward

Initialization

$B^* = f_{r_1^1,..., r_2^1,..., r_N^1,1}(0)$;

$\underline{r}^* = (r_1^1, r_2^1, ..., r_N^1)$;

index[j] := 1 $\forall$ j=1, ..., N;

i:= N-1;

forward := true;

While i>0 **do**

If forward **then**

If i=N-1 **then**

While index[N]<nr_N **and** $LB_{N-1}(r_1^{index[1]}, r_2^{index[2]}, ..., r_{N-1}^{index[N-1]}) < B^*$ **do**

index[N] := index[N]+1;

If $f_{r_1^{index[1]}, r_2^{index[2]},..., r_N^{index[N]},1}(0) < B^*$ **then**

$B^* := f_{r_1^{index[1]}, r_2^{index[2]},..., r_N^{index[N]},1}(0)$;

$\underline{r}^* = (r_1^{index[1]}, r_2^{index[2]}, ..., r_N^{index[N]})$;

forward := false;

else

index[i] := index[i]+1;

If $LB_i(r_1^{index[1]}, r_2^{index[2]}, ..., r_i^{index[i]}) < B^*$ **then** i:=i+1 **else** forward := false;

else

i := max{ j | (0≤j≤i) and [(index[j]<nr_j) or (j=0)] };

index[i+1] := index[i+2] := ... := index[N] := 0;

forward := true;

Table 2 -The incomplete enumeration method (IE)

- compute $y_1(I_1=0)$ and memorize the associated best strategy $(\hat{r}_1(0), \hat{r}_2(\cdot), ..., \hat{r}_N(\cdot), q_1(0), \hat{\underline{q}}_2(\cdot), ..., \hat{\underline{q}}_N(\cdot))$ obtained.
- $\mathcal{R}_i^- = \{r_i \in \mathcal{R}_i \mid \exists\, I_i \text{ such that } \hat{r}_i(I_i) = r_i\}$
- $R^- = \{\underline{r} = (r_1, r_2, ..., r_N) \mid r_i \in \mathcal{R}_i^- \ \forall i\}$
- use the branch and bound algorithm to get the best sequence $\underline{r}^* \in R^-$ and the associated $\underline{q}^*$;

Table 3 - Generation of the weights $w_i(I_i)$

Variables

nsimul	number of simulations performed
$nsimul_i(I_i)$	number of times inventory I_i is simulated

$$nsimul_1(I_1) := \begin{cases} nsimul \text{ if } I_1=0 \\ 0 \quad \text{otherwise} \end{cases} ;$$

For i:=2 **to** N **do**

  $nsimul_i(I_i) := 0\ \forall I_i$;

  For every possible I_{i-1} **do**

    For s:=1 **to** $nsimul_{i-1}(I_{i-1})$ **do**

      generate randomly $X_{i-1} \in \mathcal{X}_{i-1}$ and $C_{i-1} \in \mathcal{C}_{i-1}$ according to their probability functions $p_{i-1}(X_{i-1})$ and $g_{i-1}(C_{i-1})$;

      $I_i := \max\{I_{i-1} + \min[\max(C_{i-1}, r^*_{i-1}), q^*_{i-1}(I_{i-1})] - X_{i-1}, 0\}$;

      $nsimul_i(I_i) := nsimul_i(I_i) + 1$;

For i:=1 **to** N **do** $w_i(I_i) := \dfrac{nsimul_i(I_i)}{nsimul}\ \forall I_i$;

Table 4 - The constructive algorithm CA(n)

- examine all sequences $(r_1, r_2, ..., r_n)$ to find the best strategy $(r_1^*, r_2^*, ..., r_n^*, q_1^*(0), \underline{q}_2^*(\cdot), ..., \underline{q}_n^*(\cdot))$;
- keep track of r_1^* and $q_1^*(0)$;

For i:=2 **to** N **do**

  For every possible I_i **do**

    For every $\tilde{r}_i \in \mathcal{R}_i$ **do**

      examine all sequences $(\tilde{r}_i, r_{i+1}, ..., r_{i+n-1})$ to determine the cost $\tilde{f}_{i,i+n-1}(I_i, \tilde{r}_i)$ of the best strategy of the form $(\tilde{r}_i, r_{i+1}, ..., r_{i+n-1}, q_i(I_i), \underline{q}_{i+1}(\cdot), ..., \underline{q}_{i+n-1}(\cdot))$;

  generate the weights $w_i(I_i)$ as indicated in Table 3;

  look for the value $r_i^* \in \mathcal{R}_i$ minimizing function $\tilde{F}_{i,i+n-1}$;

  keep track of r_i^* and $\underline{q}_i^*(\cdot)$;

Table 5 - The tabu search method TS(ε,nmv)

choose an initial sequence $\underline{r} \in R$ and compute the associated $\underline{q}(\cdot)$;	
$\underline{r}^*:=\underline{r}$ and $\underline{q}^*(\cdot):=\underline{q}(\cdot)$;	(best strategy found so far)
iter:= 0;	(iteration counter)
bestiter:=0;	(iteration at which the best strategy was found)
T:=Ø;	(the tabu list is empty initially)

While iter-bestiter<nitermax **do**

  iter:=iter+1;

  identify the best nmv non tabu moves m yielding $\underline{r}' = \underline{r}\oplus m \in V^* \subseteq N^1_{1,N}(\underline{r})$

  For each acceptable move m **do**

    set $\underline{r} = \underline{r}\oplus m$ and compute the associated $\underline{q}(\cdot)$;

    update the tabu list T;

    If $(f_{\underline{r},1}(0) < f_{\underline{r}^*,1}(0))$ **then**

      bestiter:=iter;

      $\underline{r}^*:=\underline{r}$ and $\underline{q}^*(\cdot):=\underline{q}(\cdot)$;

Table 6 - The candidate list strategy CLS(s)

choose an initial sequence $\underline{r} \in R$ and compute the associated $\underline{q}(\cdot)$;

$\underline{r}^* := \underline{r}$ and $\underline{q}^*(\cdot) := \underline{q}(\cdot)$; (best strategy found so far)

ndec:= 0; (decomposition counter)

stop:= false; (boolean variable indicating whether the stopping criterion is met)

While not stop **do**

stop:=true;

incr:= ndec mod s; ndec:=ndec+1;

For m:=0 **to** $\left\lceil \frac{N\text{-incr}}{s} \right\rceil$ **do**

seek the move $m \in M^{s}_{incr+1+(m-1)\cdot s, incr+m\cdot s}$ yielding the best

$\underline{r}' = \underline{r} \oplus m \in N^{s}_{incr+1+(m-1)\cdot s, incr+m\cdot s}(\underline{r})$ and identify the resulting $\underline{q}'(\cdot)$;

$\underline{r} := \underline{r}'$ and $\underline{q}(\cdot) := \underline{q}'(\cdot)$;

If $f_{\underline{r},1}(0) < f_{\underline{r}^*,1}(0)$ **then**

$\underline{r}^* := \underline{r}$ and $\underline{q}^*(\cdot) := \underline{q}(\cdot)$;

stop:= false;

Table 7 - Results obtained with the branch and bound algorithm

	nr=4				nr=5			
N	$f_{\underline{r}^*,1}(0)$	CPU (sec)	pc_sp	pc_r	$f_{\underline{r}^*,1}(0)$	CPU (sec)	pc_sp	pc_r
5	329.79	<1	7.96	20.01	309.56	3	8.53	16.27
6	383.88	3	3.74	11.91	360.24	10	2.57	11.10
7	443.72	8	4.03	7.69	415.32	30	2.23	5.30
8	498.17	22	2.46	5.56	468.15	72	0.08	1.10
9	552.21	60	0.36	2.00	521.16	184	0.02	0.40
10	612.81	243	0.11	0.95	578.06	916	0.03	0.23
11	678.69	1290	0.01	0.79	640.75	10952	0.02	0.25
12	743.35	3195	0.04	0.34	702.00	48088	<0.01	0.18

Table 8 - Results obtained with the heuristic methods (N= 6, 8)

	N=6			N=8		
method	cost	CPU (sec)	success	cost	CPU (sec)	success
OLP	351.06	<1	-	455.80	<1	-
B&B	360.24	10	-	468.15	72	-
IE	360.24	<1	8	468.15	2	8
CA(1)	360.24	<1	8	468.19	<1	5
CA(2)	360.27	<1	7	468.21	<1	4
CA(3)	360.45	<1	4	468.21	<1	5
TS(0.5,1)	360.90	<1	2	468.34	2	3
TS(1,1)	360.24	1	8	468.15	4	8
TS(1,$\lceil N/4 \rceil$)	360.24	1	8	468.34	3	3
CLS(1)	360.24	<1	8	468.34	2	3
CLS(2)	360.24	<1	8	468.15	1	8
CLS(3)	360.24	4	8	468.15	6	8

Table 9- Results obtained with the heuristic methods (N= 10, 12)

	N=10			N=12		
method	cost	CPU (sec)	success	cost	CPU (sec)	success
OLP	560.57	<1	-	679.29	<1	-
B&B	578.06	916	-	702.00	48088	-
IE	578.06	15	8	702.00	78	8
CA(1)	581.09	<1	2	705.29	<1	1
CA(2)	581.13	<1	1	706.02	<1	2
CA(3)	581.24	1	1	710.12	2	1
TS(0.5,1)	580.38	5	1	703.79	11	1
TS(1,1)	578.42	10	3	702.20	19	3
TS(1,$\lceil N/4 \rceil$)	578.94	14	3	703.13	20	4
CLS(1)	579.67	4	2	706.59	8	2
CLS(2)	578.06	14	8	702.00	32	8
CLS(3)	578.06	55	8	702.00	103	8

Table 10 - Results obtained with the heuristic methods (N= 14, 16, 18, 20)

	N=14		N=16		N=18		N=20	
method	cost	CPU (sec)	cost	CPU (sec)	cost	CPU (sec)	cost	CPU (sec)
OLP	798.61	<1	907.07	<1	1010.95	<1	1129.43	1
IE	826.77	285	938.35	604	1046.18	1090	1171.44	5712
CA(1)	829.41	<1	943.61	<1	1048.20	<1	1175.15	<1
CA(2)	831.31	1	944.39	1	1055.26	1	1180.55	2
CA(3)	830.49	3	945.77	4	1048.67	5	1184.63	6
TS(0.5,1)	827.54	17	938.73	27	1048.27	48	1174.98	65
TS(1,1)	826.83	33	938.65	51	1046.37	75	1171.88	133
TS(1,$\lceil N/4 \rceil$)	826.83	40	939.04	76	1047.41	84	1172.69	128
CLS(1)	832.73	15	948.88	20	1054.79	26	1175.50	59
CLS(2)	830.70	50	941.67	77	1051.99	104	1172.25	144
CLS(3)	827.25	210	939.48	321	1046.64	481	1171.44	509

Table 11 - Results obtained with the heuristic methods (N= 24, 30, 36, 40)

	N=24		N=30		N=36		N=40	
method	cost	CPU (sec)	cost	CPU (sec)	cost	CPU (sec)	cost	CPU (sec)
OLP	1352.57	1	1678.55	2	1998.55	4	2217.59	4
CA(1)	1409.02	<1	1749.39	1	2082.55	1	2313.78	1
CA(2)	1413.34	2	1751.37	4	2084.68	6	2320.41	7
CA(3)	1411.91	9	1748.17	14	2094.76	20	2319.39	26
TS(0.5,1)	1404.76	118	1743.01	304	2075.08	535	2304.81	727
TS(1,1)	1403.41	227	1740.45	485	2073.70	948	2303.04	1345
TS(1,$\lceil N/4 \rceil$)	1403.45	209	1740.73	426	2073.75	892	2303.25	1123
CLS(1)	1408.26	95	1743.61	232	2077.42	412	2306.03	593
CLS(2)	1405.79	242	1741.57	733	2073.92	1228	2302.85	1629
CLS(3)	1403.24	1215	1740.20	2372	-	-	-	-

Figure 1 - Representation of the function $\phi_i(r_i^j)$

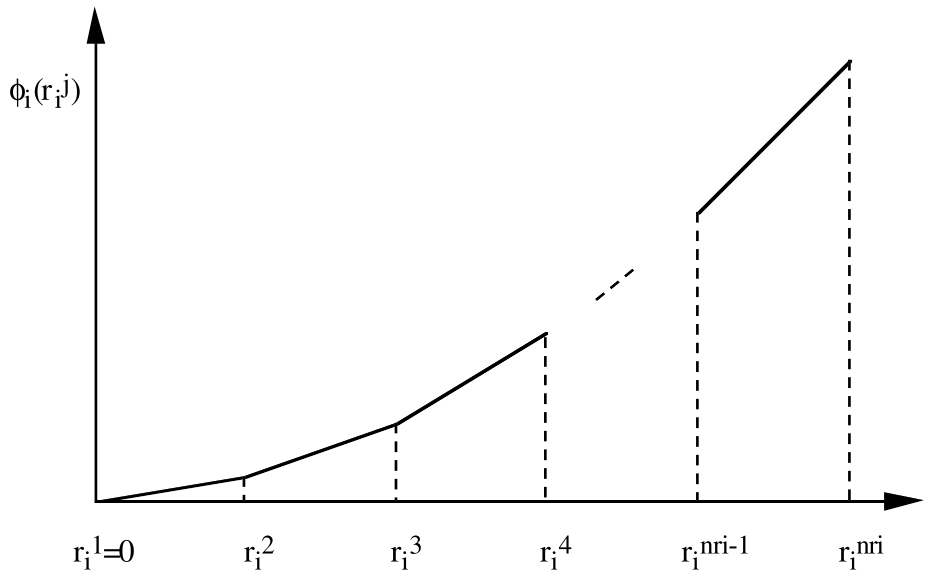

Figure 2 - Graphical representation of $f_{(r_1,r_2),1}(0)$