Constraints (2009) 14:137–176
DOI 10.1007/s10601-007-9039-3

Cost-Based Filtering Techniques for Stochastic Inventory Control Under Service Level Constraints

S. Armagan Tarim · Brahim Hnich · Roberto Rossi · Steven Prestwich

Published online: 15 February 2008

Abstract This paper[1] considers a single product and a single stocking location production/inventory control problem given a non-stationary stochastic demand. Under a widely-used control policy for this type of inventory system, the objective is to find the optimal number of replenishments, their timings and their respective order-up-to-levels that meet customer demands to a required service level. We extend a known CP approach for this problem using three cost-based filtering methods. Our approach can solve to optimality instances of realistic size much more efficiently than previous approaches, often with no search effort at all.

Keywords Inventory control · Non-stationary (R, S) policy · Stochastic demand · Cost-based filtering · Dynamic programming relaxation

This work was supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR) and Grant No. 00/PI.1/C075.

[1] This paper is an extended version of [19].

S. A. Tarim
Department of Management, Hacettepe University, Ankara, Turkey
e-mail: armagan.tarim@hacettepe.edu.tr

B. Hnich
Faculty of Computer Science, Izmir University of Economics, Izmir, Turkey
e-mail: brahim.hnich@ieu.edu.tr

R. Rossi (✉) · S. Prestwich
Cork Constraint Computation Centre, University College,
14 Washington St. West, Cork, Ireland
e-mail: rrossi@4c.ucc.ie

S. Prestwich
e-mail: s.prestwich@4c.ucc.ie

R. Rossi
Centre for Telecommunication Value—Chain Driven Research, Dublin, Ireland

1 Introduction

Inventory theory provides methods for managing and controlling inventories under different constraints and environments. An interesting class of production/inventory control problems is the one that considers the single-location, single-product case under non-stationary stochastic demand. Such a problem has been widely studied because of its key role in practice.

We consider the following inputs: a planning horizon of N periods and a demand d_t for each period $t \in \{1, \ldots, N\}$, which is a random variable with probability density function $g_t(d_t)$. In the following sections we will assume without loss of generality that these variables are normally distributed. We assume that the demand occurs instantaneously at the beginning of each time period. The demand we consider is non-stationary, that is it can vary from period to period, and we also assume that demands in different periods are independent. A fixed delivery cost a is considered for each order and also a linear holding cost h is considered for each unit of product carried in stock from one period to the next. Demands occurring when the system is out of stock are assumed to be back-ordered and satisfied as soon as the next replenishment order arrives. We assume that it is not possible to sell back excess items to the vendor at the end of a period. Our aim is to find a replenishment plan that minimizes the expected total cost, which is composed of ordering costs and holding costs, over the N-period planning horizon, satisfying the service level constraints. As a service level constraint we require that, with a probability of at least a given value α, at the end of each period the net inventory will be non-negative.

We decided to ignore in this model the linear production cost p, incurred for each unit produced. The logic behind this simplification of the problem is as follows. In the deterministic production planning problem, since all the demand has necessarily to be met, any optimal solution is independent of the given production cost. The production cost is therefore a constant of the problem. This is also true for the stochastic production planning problem under infinite horizon, provided that demands occurring when the system is out of stock are back-ordered and satisfied as soon as the next replenishment order arrives. Again the justification is that when time tends to infinity, under a demand back-ordering assumption, all the realized demand will be necessarily satisfied and the production cost will become a constant of the problem. When the planning horizon is finite, as in our case, the production cost may have an impact on the structure of an optimal solution, as in an optimal solution we will tend to clear up stocks when we approach the end of the planning horizon. This may therefore affect the length of some replenishment cycles at the end of the planning horizon. In fact we may have a shorter final cycle in order to keep less buffer stocks at the very last period, especially if the production cost is high. On the other hand the proposed model has to be considered within the more general picture of inventory control. Typically a finite planning horizon assumption is made because forecasts cannot look too far ahead in time. This does not mean that production will stop at the end of the planning horizon: rather, a new optimization will often occur at that point, which considers new forecast information that has become available. This process is common in inventory control and it is known as a *rolling horizon* [17] approach. It is obvious that, under a rolling horizon approach and a demand

back-ordering assumption, again in the long run we will tend to satisfy all the realized demand and the production cost will again become a constant of the problem as in the infinite horizon case. Moreover it should be noted that in this case considering a production cost p may even lead to suboptimal solutions, in fact we may schedule more replenishment cycles than strictly needed in order to keep unsold stocks low at the end of the given finite horizon. But since the production does not stop at the end of the finite horizon this will give no real cost benefit and will instead increase the total fixed delivery cost in the long run. For this reason we ignore such a cost component as Bookbinder and Tan do in their heuristic approach [4]. On the other hand extending the results in this paper to consider a production cost p is easy, and in Appendix 7.1 we will describe how this can be done.

Different inventory control policies can be adopted for the described problem. A policy states the rules to decide when orders have to be placed and how to compute the replenishment lot-size for each order. For a discussion of inventory control policies see [17].

One of the possible policies that can be adopted is the replenishment cycle policy, (R, S).

Under the non-stationary demand assumption this policy takes the form (R^n, S^n) where R^n denotes the length of the nth replenishment cycle and S^n the order-up-to-level for replenishment (Fig. 1). In this policy a wait-and-see strategy is adopted, under which the actual order quantity Q^n for replenishment cycle n is determined only after the demand in former periods has been realized. The order quantity Q^n is computed as the amount of stock required to raise the closing inventory level of replenishment cycle $n-1$ up to level S^n. In order to provide a solution for our problem under the (R^n, S^n) policy we must populate both the sets R^n and S^n for $n = \{1, \ldots, N\}$.

There is a large literature on deterministic production planning. This problem has been mentioned by Garey and Johnson [11]. In [8] Florian et. al. gave an overview for the complexity of this problem. In particular they established NP-hardness for this

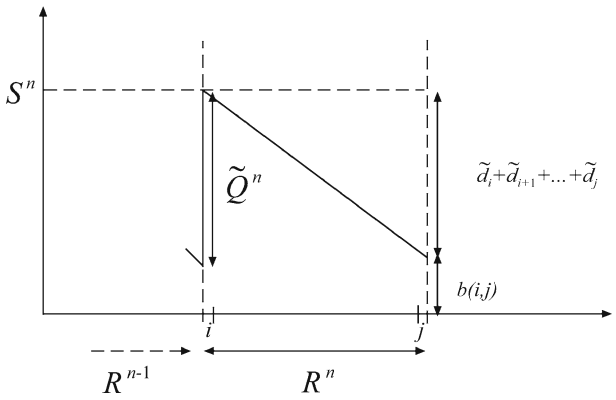

Fig. 1 (R^n, S^n) policy. R^n denotes the set of periods covered by the nth replenishment cycle; S^n is the order-up-to-level for this cycle; $\tilde{Q}^n$ is the expected order quantity; $\tilde{d}_i + \tilde{d}_{i+1} + \ldots + \tilde{d}_j$ is the expected demand; $b(i, j)$ is the buffer stock required to guarantee the required service level α

problem under production cost (composed of a fixed cost and a variable unit cost), zero-holding cost and arbitrary production capacity constraint. They also extended this result by considering other possible cost functions and capacity constraints. Polynomial algorithms are discussed in the same paper for a few specific cases. Among these they cited Wagner and Whitin's [25] work, where the infinite capacity deterministic production planning problem is solved in polynomial time.

In contrast the respective stochastic formulation for this problem has been solved to optimality only recently, due to the complexity involved in the modeling of uncertainty and of the policy-of-response. Early works in this area adopted heuristic strategies such as those proposed by Silver [16], Askin [2] and Bookbinder and Tan [4]. Under some mild assumptions the first complete solution method for this problem was introduced by Tarim and Kingsman [20], who proposed a *deterministic equivalent* Mixed Integer Programming (MIP) formulation for computing (R^n, S^n) policy parameters. Empirical results showed that such a model is unable to solve large instances, but Tarim and Smith [23] introduced a more compact and efficient Constraint Programming (CP) formulation of the same problem that showed a significant computational improvement over the MIP formulation. A *stochastic constraint programming* [22] approach for computing (R^n, S^n) policy parameters is proposed in [14]. In this work the authors drop the mild assumptions originally introduced by Tarim and Kingsman and compute optimal (R^n, S^n) policy parameters. Of course there is a price to pay for dropping Tarim and Kingsman's assumptions, in fact this latter approach is less efficient than the one in [23].

This paper extends Tarim and Smith's work, which builds on Tarim and Kingsman's assumptions. We retain their model and we augment such a model with three *cost-based filtering* methods to enhance domain pruning. One of these techniques, based on a relaxation proposed by Tarim [18] and solved by means of dynamic programming, has been already presented in [19]. In this work we provide two additional cost-based filtering techniques and we extend the discussion on Tarim's relaxation and on the implementation of the respective cost-based filtering method.

Cost-based filtering is an elegant way of combining techniques from CP and Operations Research (OR) [7, 9]. OR-based optimization techniques are used to remove values from variable domains that cannot lead to better solutions. This type of domain filtering can be combined with the usual CP-based filtering methods and branching heuristics, yielding powerful hybrid search algorithms. Cost-based filtering is a novel technique that has been the subject of significant recent research, but to the best of our knowledge it has not previously been applied to stochastic inventory control. In the following sections we will show that it can bring a significant improvement when combined with the state-of-the-art CP model for stochastic inventory control. It should be noted that while the technique based on Tarim's relaxation can easily be recognized as a classic *cost-based filtering* method, the two additional techniques here presented are not based on bounds obtained through a relaxation. Instead, as we will see, they exploit reasoning on the problem cost structure to prune values in the domains of decision variables that cannot lead to optimal solutions. Our experimental results show the efficiency obtained by the combined used of these three filtering techniques during the search for an optimal solution.

The paper is organized as follows. Section 2 describes the CP model and the pre-processing techniques introduced by Tarim and Smith. Section 3 firstly extends one of Tarim and Smith's pre-processing techniques to cost-based filtering method, allowing it to be applied at every search tree node. Secondly it proposes a general approach for applying any sound pre-processing technique at every search tree node in a cost-based filtering fashion. Section 4 describes a relaxation that can be efficiently solved by means of a shortest path algorithm, and produces tight lower bounds for the original problem which is used to perform further cost-based filtering. Section 5 evaluates our methods. Section 6 draws conclusions and discusses future extensions.

2 A CP Model

In this section we review the CP formulation for the (R^n, S^n) policy proposed by Tarim and Smith [23]. First we provide some formal background related to stochastic programming.

Stochastic programming [3] is a well known modeling technique that deals with problems where uncertainty comes into play. Problems of optimization under uncertainty are characterized by the necessity of making decisions without knowing what their full effect will be. Such problems appear in many area of application and present many interesting conceptual and computational challenges. Stochastic programming needs to represent uncertain elements of the problem. Typically random variables are employed to model this uncertainty to which probability theory can be applied. For this purpose such uncertain elements must have a known probability distribution. The typical requirement in stochastic programs is to maintain certain constraints, called *chance constraints* [6], satisfied at a prescribed level of probability. The objective is typically related to the minimization/maximization of some expectation on the problem costs. There are several different approaches to tackle stochastic programs. A first method dealing with stochastic parameters in stochastic programming is the so-called *expected value model* [3], which optimizes the expected objective function subject to some expected constraints. Another method, *chance-constrained programming*, was pioneered by Charnes and Cooper [6] as a means of handling uncertainty by specifying a confidence level at which it is desired that the stochastic constraint holds. Chance-constrained programming models can be converted into deterministic equivalents for some special cases, and then solved by some solution methods of deterministic mathematical programming. A typical example for this technique is given by the Newsvendor problem [17]. However it is almost impossible to do this for complex chance-constrained programming models. A third approach employs scenarios, which are particular representations of how the future might unfold. Each scenario is assigned a probability value, that is its likelihood. An appropriate probabilistic model or simulation is used to generate a batch of such scenarios. The challenge then, is how to make good use of these scenarios in coming up with an effective decision.

The stochastic programming formulation for the general multi-period production/inventory problem with stochastic demand can be expressed as finding the

timing of the stock reviews and the size of the respective non-negative replenishment orders with the objective of minimizing the expected total cost $E\{TC\}$ over a finite planning horizon of N periods. The model is given below,

$$\min\ E\{TC\} = \int_{d_1}\int_{d_2}\cdots\int_{d_N}\sum_{t=1}^{N}(a\delta_t + h\cdot\max(I_t,0)) \\ g_1(d_1)g_2(d_2)\ldots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\ldots\mathrm{d}(d_N) \tag{1}$$

subject to, for $t = 1\ldots N$

$$\delta_t = \begin{cases} 1, & \text{if } \ Q_t > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$I_t = I_0 + \sum_{i=1}^{t}(Q_i - d_i) \tag{3}$$

$$\Pr\{I_t \geq 0\} \geq \alpha \tag{4}$$

$$I_t \in \mathbb{Z}, \quad Q_t \geq 0, \quad \delta_t \in \{0,1\}. \tag{5}$$

Each decision variable I_t represents the inventory level at the end of period t. The binary decision variables δ_t state whether a replenishment is fixed for period t ($\delta_t = 1$) or not ($\delta_t = 0$). If an order is placed in period t, constraint (2), decision variable Q_t denotes the size of the respective non-negative replenishment order. Chance constraint (4) enforces the required service level, that is the probability α that the net inventory will not be negative at the end of each time period. The objective function (1) minimizes the expected total cost over the given planning horizon.

In [20] the authors assume that negative orders are not allowed, so that if the actual stock exceeds the order-up-to-level for that period, this excess stock is carried forward and not returned to the supply source. However, such occurrences are regarded as rare events and accordingly the cost of carrying the excess stock and its effect on the service level of subsequent periods is ignored. Under these assumptions the chance-constrained problem can be expressed by means of a *deterministic equivalent* model where buffer stocks for each possible replenishment cycle are computed independently.

We now recall some basic notions about *constraint programming*. A *Constraint Satisfaction Problem* (CSP) [1, 5] is a triple $\langle V, C, D\rangle$, where V is a set of decision variables each with a discrete domain of values $D(V_k)$, and C is a set of constraints stating allowed combinations of values for subsets of variables in V. Finding a solution to a CSP means assigning values to variables from the domains without violating any constraint in C. We may also be interested in finding a feasible solution that minimizes (maximizes) the value of a given objective function over a subset of the variables. Constraint solvers typically explore partial assignments enforcing a local consistency property using either specialized or general purpose propagation algorithms. Such propagation algorithms in general exploit some structure of the problem to prune decision variable domains in more efficient ways.

The following CP formulation of the *deterministic equivalent* model for the (R^n, S^n) policy is proposed in [23]:

$$\min \; E\{TC\} = \sum_{t=1}^{N} \left(a\delta_t + h\tilde{I}_t \right) \tag{6}$$

subject to, for $t = 1 \ldots N$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} \geq 0 \tag{7}$$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} > 0 \Rightarrow \delta_t = 1 \tag{8}$$

$$\tilde{I}_t \geq b \left(\max_{j \in \{1,\ldots,t\}} j \cdot \delta_j, t \right) \tag{9}$$

$$\tilde{I}_t \in \mathbb{Z}^+ \cup \{0\}, \quad \delta_t \in \{0, 1\}, \tag{10}$$

where $b(i, j)$ is defined by

$$b(i, j) = G^{-1}_{d_i + d_{i+1} + \ldots + d_j}(\alpha) - \sum_{k=i}^{j} \tilde{d}_k.$$

Constraint (9), originally proposed by Tarim and Smith, can be implemented by means of the following set of constraints, for $t = 1 \ldots N$

$$Y_t \geq j \cdot \delta_j \quad j = 1, \ldots, t \tag{11}$$

$$element\,(Y_t, b(\cdot, t), H_t) \tag{12}$$

$$\tilde{I}_t \geq H_t \tag{13}$$

$$\tilde{I}_t, H_t \in \mathbb{Z}^+ \cup \{0\}, \quad \delta_t \in \{0, 1\}, \quad Y_t \in \{1, \ldots, N\}. \tag{14}$$

The $element(X, list[], Y)$ constraint [24] enforces a relation such that variable Y represents the value of element at position X in the given list. $G_{d_i + d_{i+1} + \ldots + d_j}$ is the cumulative probability distribution function of $d_i + d_{i+1} + \ldots + d_j$. It is assumed that G is strictly increasing, hence G^{-1} is uniquely defined.

Each decision variable $\tilde{I}_t$ represents the expected inventory level at the end of period t. Each $\tilde{d}_t$ represents the expected value of the demand in a given period t according to its probability density function $g_t(d_t)$. The binary decision variables δ_t state whether a replenishment is fixed for period t ($\delta_t = 1$) or not ($\delta_t = 0$). The objective function (6) minimizes the expected total cost over the given planning horizon. The two terms that contribute to the expected total cost are ordering costs and inventory holding costs. Constraint (7) enforces a no-buy-back condition, which means that received goods cannot be returned to the supplier. As a consequence of this the expected inventory level at the end of period t must be no less than the expected inventory level at the end of period $t - 1$ minus the expected demand in period t. Constraint (8) expresses the replenishment condition. We have a replenishment if the expected inventory level at the end of period t is greater than the expected inventory level at the end of period $t - 1$ minus the expected demand in period t. This means that we received some extra goods as a consequence of an order. Constraints (9)

enforce the required service level α. This is done by specifying the minimum buffer stock required for each period t in order to assure that, at the end of each and every time period, the probability that the net inventory will not be negative is at least α. These buffer stocks, which are stored in matrix $b(\cdot, \cdot)$, are pre-computed following the approach suggested in [20]. In this approach the authors transformed a chance-constrained model, that is a model where constraints on some random variables have to be maintained at prescribed levels of probability, in a completely deterministic one. For further details about chance-constrained programming see [6].

2.1 Domain Pre-processing

In [23] the authors showed that a CP formulation for computing optimal (R^n, S^n) policies provides a more natural way of modeling the problem. In contrast to the equivalent MIP formulation the CP model requires fewer constraints and provides a neater formulation. However, the CP model has two major drawbacks. Firstly, in order to improve the search process and quickly prove optimality, tight bounds on the objective function are needed. Secondly, even when it is possible to compute *a priori* the maximum values that such variables can be assigned to, these values (and therefore the domain sizes of the $\tilde{I}_t$ variables) are large. The domain size value is equal to the amount of stock required to satisfy subsequent demands till the end of the planning horizon, meeting the required service level when only a single replenishment is scheduled at the beginning of the planning horizon.

To address the domain size issue, Tarim and Smith proposed two pre-processing methods in order to reduce the size of the domains before starting the search process, by exploiting properties of the given model and of the (R^n, S^n) policy. Method I computes a cost-based upper bound for the length of each possible replenishment cycle $T(i, j)$, starting in period i, for all $i, j \in \{1, \ldots, N\}$, $i \leq j$. Note that $T(i, j)$ denotes the time span between two consecutive replenishment periods i and $j+1$. Method I therefore identifies sub-optimal replenishment cycle lengths allowing a proactive off-line pruning, which eliminates all the expected inventory levels that refer to longer sub-optimal replenishment cycles. Method II employs a dynamic programming approach, by considering each period in an iterative fashion and by taking into account in each step two possible courses of action: either an order with an expected size greater than zero is placed, or no order (equivalently an order with a null expected size) is placed in the considered period within our planning horizon. The effects of these possible actions in each step are reflected in the decision variable domains by removing values that are not produced by any course of action.

3 From Pre-processing to Cost-Based Filtering

In the previous section we described a CP formulation for the (R^n, S^n) policy. In [23] the authors discussed the advantages of such a formulation when it is compared to the MIP formulation proposed in [20]. CP not only performs faster than MIP and provides a neater formulation, it also allows us to build dedicated filtering algorithms for pruning infeasible and/or suboptimal values for the domains of decision variables during the search.

In Section 3.1 we extend the first of the two pre-processing methods proposed in [23] in order to exploit partial assignments of decision variables in the model to prune suboptimal values from the domains of the remaining decision variables still unassigned at any point of the search process.

In Section 3.2, we describe a generic approach to applying pre-processing techniques not only in a proactive way, before the search process starts, but also during the search, by exploiting partial information which derives from the current decision variable assignments. We emphasize that this approach may be used in conjunction with any sound pre-processing method developed for our inventory/production problem and it is not limited to the two pre-processing methods proposed in [23].

A running example is given to show that the two methods proposed are incomparable in term of domain reduction achieved.

3.1 Tighter Upper Bounds for Optimal Replenishment Cycle Lengths

We now present a filtering method that is a natural extension of pre-processing method I in [23]. This method prunes variable domains, when a partial solution is given, by enforcing tighter upper bounds for optimal replenishment cycle lengths than those proposed by Tarim and Smith. When no partial solution is provided this filtering method realizes the same domain reduction performed by the respective pre-processing method.

Firstly let $R(i, j) = b(i, j) + \sum_{t=i}^{j} \tilde{d}_t$ be the required minimum opening inventory level in period i, $i \in \{1, \ldots, N\}$, to meet demand until period $j+1$. The cycle cost $c(i, j)$, when a variable holding cost h_t ($t \in \{1, \ldots, N\}$) is considered, can be expressed as

$$c(i, j) = a + \sum_{t=i}^{j} h_t b(i, j) + \sum_{t=i}^{j-1} h_t \sum_{k=t+1}^{j} \tilde{d}_k. \tag{15}$$

The cost (15) of a replenishment cycle is the sum of two components. A fixed ordering cost a, that is charged at the beginning of the cycle when an order is placed, and a variable holding cost h_t charged at the end of each time period within the replenishment cycle and proportional to the amount of stocks held in inventory. In [23], for each period $i \in \{1, \ldots, N\}$ over the planning horizon N, an upper bound for the length of an optimal replenishment cycle $T(i, p)^*$ that starts in such a period is proposed. The authors compute *a priori* this bound for every period i and derive from it a superset of all candidate opening-inventory-levels for any period in the planning horizon. Let us refer to this bound as B (Fig. 2a), and let $j = i + B$. Then the last period p of an optimal replenishment cycle $T(i, p)^*$ satisfies $i \leq p \leq j$. $j = i + B$ can be computed as the minimum j satisfying the following conditions described in [23], which formally identify bound B

$$c(i, k) + c(k+1, j) > c(i, j) \vee b(i, k) > R(k+1, j) \tag{16}$$

for all $k \in \{i, \ldots, j-1\}$, and

$$c(i, k) + c(k+1, j+1) \leq c(i, j+1) \wedge b(i, k) \leq R(k+1, j+1) \tag{17}$$

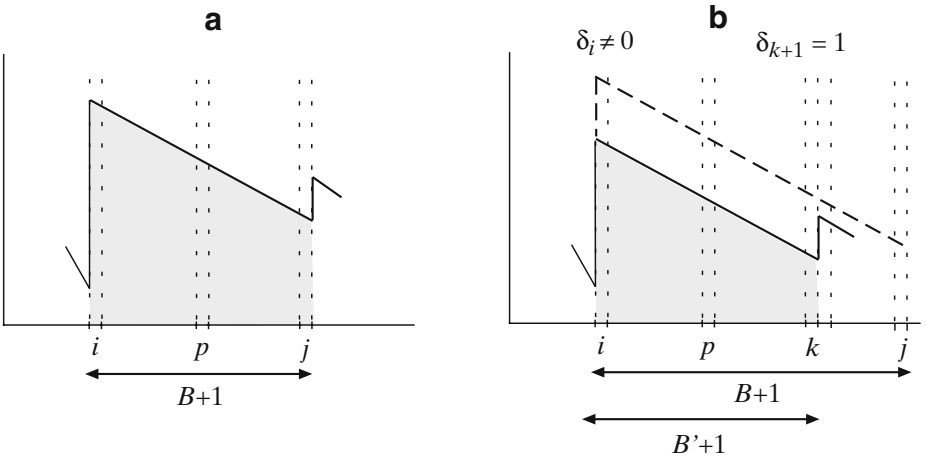

Fig. 2 Bound tightening when a partial solution is given: **a** since it is not optimal to cover more than $B+1$ periods with a single replenishment in i, the optimal policy lies in the gray area; **b** the bound B can be tightened to B' when an order is scheduled in period $k+1$, $i \le k < j$

for some $k \in \{i, \ldots, j\}$, given that $\forall p \in \{j+2, \ldots, N\}$ such a k satisfies

$$-\sum_{t=j+2}^{p}(k+1-i)\tilde{d}_t + (p-k)b(k+1, p) - (j-k+1)b(k+1, j+1)$$
$$\le (p-i+1)b(i, p) - (j-i+2)b(i, j+1). \tag{18}$$

A proof for these conditions is given in Appendix 7.2.

When a partial solution S is given, it is possible to tighten the bound B by using the following observations:

- If δ_i is assigned to 0 then no replenishment cycle starts in period i.
- If δ_i is not assigned to 0 and $\exists k \in \{i, \ldots, i+B-1\}$ such that $\delta_{k+1} = 1$, then B can be tightened to the smallest $k-i$ value B' (Fig. 2b)

In order to compute the tighter bound B' for a given period $i \in \{1, \ldots, N\}$ when a partial solution S is given we introduce the following Lemma.

Lemma 1 *If there exists some $k \in S$ such that $\delta_{k+1} = 1$ and $i \le k < j$, then B can be tightened to $B' = j' - i$ where*

$$j' = \min\left(\{k|\ \delta_{k+1} = 1, k \in \{i, \ldots, j-1\}\} \bigcup \{j\}\right).$$

Proof Trivially the replenishment scheduled in period $k+1$ rules out the chance of covering periods $i, \ldots, j$ where $j > k$ with a single cycle. □

By means of the described tighter bound B' we can now obtain smaller supersets of all candidate opening-inventory-levels than those described in [23]. For convenience in what follows we will refer to the expected closing-inventory-levels, that is opening-inventory-level minus expected demand in the period considered.

A first reduction in the size of the super-sets is due to the fact that if δ_i is assigned to zero, no replenishment cycle starts in period i. Therefore no value that is a candidate expected closing-inventory-level for any replenishment cycle starting in period i is feasible with respect to the given partial solution. Otherwise candidate values can be computed as described in the following:

Lemma 2 *When δ_i is not assigned to 0, a sufficient but not necessary condition that identifies candidate expected closing-inventory-level values in $Dom(\tilde{I}_m)$, $m \in T(i, j')$ for a replenishment cycle starting in period i is defined as follows* (*see Fig.* 3):

$$Dom\left(\tilde{I}_m\right) \supseteq \left\{ \tau \;\middle|\; \tau = R(i,l) - \sum_{t=i}^{m} \tilde{d}_t,\ l \in \{m, \ldots, j'\} \right\}. \tag{19}$$

Proof As shown in [23], (19) considers in $Dom(\tilde{I}_m)$ for each $m \in T(i, j')$ every value that is feasible if there is a replenishment cycle starting in period i. In fact if p denotes the final period of the optimum length replenishment cycle for period i, $\delta_k = 0$, $k = \{i+1, \ldots, p\}$, the optimum expected closing inventory level for period m, where $i \leq m \leq p$, is $R(i, p) - \sum_{t=i}^{m} \tilde{d}_t$. The domain of possible values is therefore obtained by letting p range from m to j. Tightening j to j' is correct because, when a partial solution is given, this ignores values related to every infeasible replenishment cycles $T(i, r)$, where $j' < r \leq j$ and $\delta_{j'+1} = 1$, if any exists. □

The former condition is only sufficient because there may exist other candidate values that should be in $Dom(\tilde{I}_m)$ as we did not take into account *negative order quantity* scenarios. Such situations arise when for some $m \in T(i, j')$, $c(i, m) + c(m+1, j') \leq c(i, j')$ and $b(i, m) > R(m+1, j')$ (Fig. 4a). In this case, since the replenishment policy expects a negative order and is infeasible, an optimal policy can be

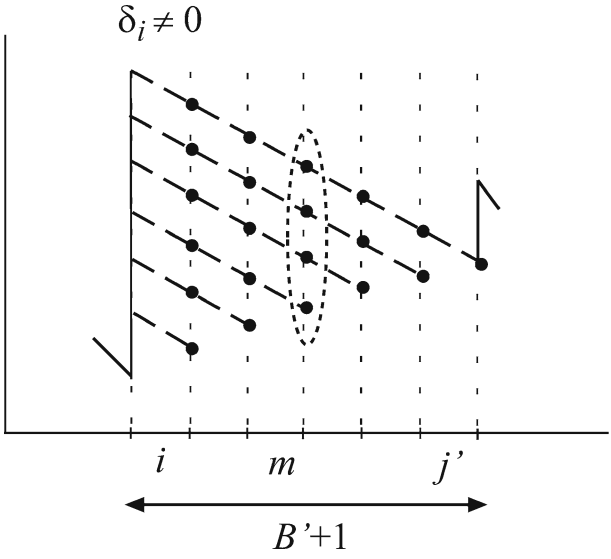

Fig. 3 Subset of candidate optimal expected closing-inventory-levels for period m, $m \in \{i, \ldots, j'\}$. These values can be computed as stated in Lemma 2. The whole set of candidate levels shown in the picture may be computed by ranging m from i to j'

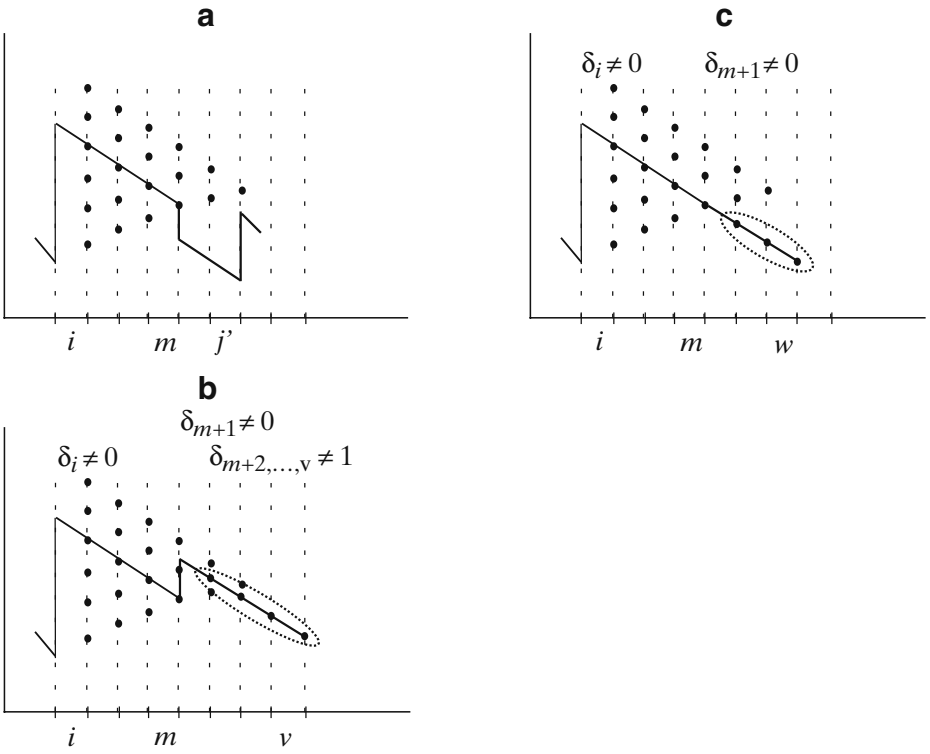

Fig. 4 **a** Negative order quantity scenario. Additional values, computed by Lemma 3, to be considered in the subset of candidate optimal expected closing-inventory-levels for each period p when **b** an order with expected size greater than zero is scheduled in period $m+1$, $p \in \{m+1, \ldots, h'\}$, **c** an order with expected size zero is scheduled in period $m+1$, $p \in \{m+1, \ldots, w\}$. In both cases $\delta_{m+1} \neq 0$ since it must be possible to schedule an order in period $m+1$

either the one that schedules a new order in period $m+1$ with an expected lot-size greater than zero (Fig. 4b) or an expected lot-size of zero (Fig. 4c). Lemma 3 and 4 characterize which additional values have to be considered when a negative order quantity scenario arises.

Lemma 3 *If* $\delta_{m+1} = 0$, (19) *is a necessary and sufficient condition that identifies candidate expected closing-inventory-level values in* $Dom(\tilde{I}_m)$, $m \in T(i, j')$ *for a replenishment cycle starting in period i.*

Proof In [23] it is stated that, if i is a replenishment period and we want to cover subsequent periods up to m, in a feasible policy a replenishment should then be scheduled in $m+1$. Since $\delta_{m+1} = 0$, it is not feasible to cover periods from i to m with a single order in i because to do so we would need an additional order in period $m+1$ that is ruled out by the partial assignment. □

Lemma 4 *If δ_{m+1} is not assigned to zero, every further candidate expected closing-inventory-level value for a replenishment cycle starting in period i can be identified by considering two possible courses of action:*

- *A new order is scheduled for period $m+1$ and its expected size is greater than zero (Fig. 4b). In this case, if $\delta_k \neq 1$ for $k = \{m+2, \ldots, v\}$, we also consider the following candidate expected closing-inventory levels*

$$Dom\left(\tilde{I}_n\right) \supseteq \left\{ \tau \;\middle|\; \tau = R_{(m+1,v)} - \sum_{t=m+1}^{n} \tilde{d}_t \right\}, \tag{20}$$

  for $n = \{m+1, \ldots, v\}$, where $v = \min\left\{ l \,\middle|\, b(m+1, l) + \sum_{t=m+1}^{l} \tilde{d}_t \geq b(i, m) \right\}$.
- *A new order is scheduled for period $m+1$ and its expected size is zero (Fig. 4c). In this case we also consider the following candidate expected closing-inventory levels*

$$Dom\left(\tilde{I}_n\right) \supseteq \left\{ \tau \;\middle|\; \tau = b_{(i,m)} - \sum_{t=m+1}^{n} \tilde{d}_t \right\}, \tag{21}$$

  for $n \in \{m+1, \ldots, w\}$, where

$$w = \max\left\{ l \,\middle|\, \exists q \in \{m+1, \ldots, l\}, b(q, l) + \sum_{t=m+1}^{l} \tilde{d}_t \leq b(i, m) \right\}.$$

Proof As shown in [23], (20) adds to $Dom(\tilde{I}_n)$ every further feasible values by considering the option of placing an order whose expected lot-size is bigger than zero. In fact if we assume that the high levels of opening inventory carried from period m satisfy the service-level constraint for the following $v-1$ consecutive periods, then the remaining inventory is not enough to satisfy this constraint for period v. To comply with the service level constraint in period v, the order quantity must be at least $b(m+1, v) + \sum_{t=m+1}^{v} \tilde{d}_t - b(i, m)$. Hence this replenishment covers the periods until the end of v, where $v = \min\{l | b(m+1, l) + \sum_{t=m+1}^{l} \tilde{d}_t \geq b(i, m)\}$. If an order has been scheduled for a period $t \in \{m+2, \ldots, v\}$, then by definition the remaining inventory at the end of period m is enough to satisfy demands in periods $\{m+1, \ldots, t\}$, therefore the optimal expected order quantity for period $m+1$ is zero.

Equation 21 adds to $Dom(\tilde{I}_n)$ every further feasible values by considering the option of placing an order whose expected lot-size is zero. In this case, since the replenishment expects a zero order quantity, the excess stock may affect subsequent periods regardless of the orders placed. Therefore we look forward in the planning horizon up to the point where no following replenishment cycle may be affected by the excess stock carried on from the current one. Hence, the farthest period that may be affected is $w = \max\{l | \exists q \in \{m+1, \ldots, l\}, b(q, l) + \sum_{t=m+1}^{l} \tilde{d}_t \leq b(i, m)\}$. □

Theorem 1 *When a partial solution is given, by ranging i from 1 to N, (19–21) identify the feasible subset of values within the current $Dom(\tilde{I}_k)$, for $k \in \{1, \ldots, N\}$.*

Proof Directly follows from Lemmas 1, 2, 3 and 4. □

3.1.1 Example

We now present a running example where the planning horizon is $N = 24$ periods and the initial stock level is equal to zero. The demand is normally distributed in each period $t \in \{1, ..., N\}$ with a constant coefficient of variation $\sigma_t/\tilde{d}_t = 1/3$, where σ_t is the standard deviation of the demand in period t. The demand forecasts (mean value for each period) are listed in Table 1. The other parameters for the problem are: $a = 200$, $h = 1$, $\alpha = 0.95$. The optimal solution for the CP model when former inputs are considered is shown in Table 2. The (R^n, S^n) policy parameters, that is replenishment cycle lengths and order-up-to-levels, for this instance can be easily computed from the solution of the CP model. We applied the described filtering method without considering a given partial solution, the domain reduction achieved is therefore equivalent to the one performed by pre-processing method I introduced in [23]. This way we computed the reduced domains $Dom(I_t)$ for the decision variables I_t, $t \in \{1, ..., N\}$. These reduced domains are shown in Table 3. We now consider the partial solution shown in Table 4. Table 5 shows the reduced domains obtained when we enforce tighter upper bounds for optimal replenishment cycle lengths considering the partial solution in Table 4. From Theorem 1 it directly follows that the filtering is performed by removing from decision variables domains (Table 3) values that do not appear in Table 5, which contains the computed reduced domains with respect to the partial solution given.

We shall now see in details how feasible expected closing-inventory-levels in the reduced domains (Table 5) are computed for the first 5 periods. In the given partial solution we place an order in period 1 but not in period 2. An order is placed in period 3 therefore a replenishment cycle over periods $\{1, 2\}$ is uniquely defined. Bound B' for period 1 is 2 periods. The demand in the first period is 73 while in the second is 0. The buffer stock required at the end of period 1 is $70 \cdot 1.645 \cdot 0.3 \simeq 40$. By iterating Lemma 2 over periods $\{1, 2\}$ we obtain an expected closing-inventory-level of 40 for period 1 and again of 40 for period 2. Negative order quantity scenarios do not arise since $\delta_2 = 0$. We do not iterate Lemma 2 for period 2, since $\delta_2 = 0$ and no replenishment cycle may start in this period. In period 3 a replenishment is scheduled. The replenishment decision in period 4 is still unassigned while in period 5 no replenishment is scheduled. We apply Lemma 2 to period 3. The bound B' is 2 periods. Therefore either we may cover only the current period with a replenishment, which yields a closing inventory level of 70, or we may cover both the periods with a single replenishment, in which case the required expected closing-inventory-level is 211 in period 3 and 95 in period 4. Negative order quantity scenarios do not arise. In period 4 the bound B' is again 2. Therefore we may cover only one period with an expected closing-inventory-level of 64, or we may cover two periods by keeping respectively an expected closing-inventory-level of 173 at the end of period 4 and of 81 at the end of period 5. Negative order quantity scenario again do not arise. δ_5 is assigned to 0 therefore no replenishment cycle starts in this period.

We now consider a set of periods where negative order quantity scenarios arise. We refer to periods $\{10, 11, 12\}$. In period 10, B' is 2 periods. Therefore the two candidate expected closing inventory levels computed by Lemma 2 are $\{88, 128\}$. 88 is the expected closing-inventory-level required if only one period is covered by the replenishment scheduled in period 10, 128 is the level required to cover period 10

Table 1 Demand forecasts

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\tilde{d}_i$	73	0	128	116	92	180	28	164	28	161	37	57	181	62
i	15	16	17	18	19	20	21	22	23	24				
$\tilde{d}_i$	34	161	2	10	40	192	17	190	163	32				

Table 2 Optimal solution

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
δ_i	1	0	1	1	0	1	0	1	0	1	1	0	1	1
$\tilde{I}_i$	40	40	70	173	81	128	100	119	91	88	94	37	99	73
i	15	16	17	18	19	20	21	22	23	24				
δ_i	0	1	1	0	0	1	0	1	1	0				
$\tilde{I}_i$	39	88	86	76	36	123	106	104	123	91				

Table 3 Reduced domains after applying our filtering method when no partial solution is given

i	$Dom(\tilde{I}_i)$	i	$Dom(\tilde{I}_i)$
1	{40}	13	{99, 167}
2	{0, 40, 198}	14	{34, 37, 73, 105}
3	{70, 211}	15	{19, 39}
4	{64, 95, 173}	16	{88, 90, 100, 143}
5	{50, 81}	17	{1, 16, 73, 86, 88, 98, 141, 350}
6	{99, 128}	18	{5, 6, 63, 76, 78, 88, 131, 340}
7	{15, 71, 100}	19	{22, 23, 36, 38, 91, 300}
8	{90, 119}	20	{105, 108, 123}
9	{15, 62, 91}	21	{9, 88, 106}
10	{88, 128}	22	{104}
11	{20, 51, 91, 94}	23	{89, 123}
12	{31, 37}	24	{18, 57, 91}

The reduction achieved is equivalent to the one provided by pre-processing method I in [23]. Underlined figures are closing inventory levels of the optimal policy

Table 4 Partial solution

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
δ_i	1	0	1	–	0	1	0	1	0	–	–	0	1	–
i	15	16	17	18	19	20	21	22	23	24				
δ_i	0	1	1	0	0	1	0	1	–	0				

A "–" means that the variable has not been assigned yet

Table 5 Enforcing tighter upper bounds for optimal replenishment cycle lengths, partial solution in Table 4, underlined figures are closing inventory levels of the optimal policy

i	$Dom(\tilde{I}_i)$	i	$Dom(\tilde{I}_i)$
1	{40}	13	{99, 167}
2	{40}	14	{34, 37, 73, 105}
3	{70, 211}	15	{39}
4	{64, 95, 173}	16	{88}
5	{81}	17	{1, 16, 73, 86}
6	{99, 128}	18	{6, 63, 76}
7	{100}	19	{23, 36}
8	{90, 119}	20	{105, 123}
9	{91}	21	{106}
10	{88, 128}	22	{104}
11	{20, 51, 91, 94}	23	{89, 123}
12	{37}	24	{91}

and 11 with a single replenishment. In this case the respective expected closing-inventory-level at the end of period 11 is 91. If an order is placed in period 10 and also in period 11 the overall cost is higher than that incurred by covering both the periods with a single replenishment. On the other hand the order-up-to-level for period 11 in this case is lower than the expected closing-inventory-level in period 10. This generates a negative order quantity scenario. As stated in Lemma 4, either we cover period 11 only by scheduling an order with expected size zero. In this case the candidate level $51 = 88 - 37$ must be considered for period 11. Otherwise we try to cover more periods with the candidate level 94. By doing so we will cover subsequent periods till 12, therefore we add the candidate level $37 = 94 - 57$ to period 12. The other value in the table for period 11 is 20 that refers instead to the case in which we order in this period and we cover only 1 period with the order. This value is computed by applying Lemma 2 to this period. Since $\delta_{12} = 0$ no replenishment cycle may start in this period.

3.2 Merging Adjacent Non-Replenishment Periods

One of the limits of the domain reduction methods proposed in [23] is that they can only be applied before the search process starts. Therefore they do not take into account information regarding partial assignments for decision variables that may become available during the search process. In this section we aim to overcome this limitation with a general approach that may be applied to any pre-processing method.

We consider a given partial solution in which some decision variables δ_i are set to zero. The key idea is to transform the original problem instance into a smaller one by merging adjacent non-replenishment periods into a single new period with new expected demand and variance values. Since the demand in each period is assumed to be independent from the previous and the following demands, these new characteristics for the demand distribution in the new merged time span can be easily computed by exploiting properties of the chosen probability distribution. Once we have the smaller instance fully defined, we can apply any sound pre-processing methods, for instance one of those presented in [23], and then we can reflect the pruning achieved in the smaller instance back onto the original one. It should be noted that the following reasoning can be applied to any reduction method for the

presented CP model, and it is not limited to those presented in [23]. We propose a three-step procedure to apply any pre-processing method not only at the root node, but at every node of the search tree.

Step 1 By considering a partial solution S for the original problem instance $\mathcal{P}$, we construct a reduced problem instance $\mathcal{R}$. $\mathcal{R}$ will be described by a list of $M \leq N$ expected demand values and standard deviations and it will be built as follows. If $\delta_k = 0$ for all $k \in \{i+1, \ldots, j\}$ and $\delta_i = 1$ or δ_i is unassigned, then instead of periods $\{i, \ldots, j\}$ we introduce a new period k^* that represents such a span with an expected demand of

$$\tilde{d}_{k^*} = \sum_{t=i}^{j} \tilde{d}_t$$

and a standard deviation of

$$\sigma_{k^*} = \sqrt{\sum_{t=i}^{j} \sigma_t^2}.$$

These two expressions are well known properties of the normal distribution. The holding cost for period k^* can be expressed as $h \cdot (j - i + 1) I_{k^*} + \sum_{l=i+1}^{j} (l - i)\tilde{d}_l$, and since the second term is constant the new holding cost coefficient will be $h_{k^*} = h \cdot (j - i + 1)$. For any other period in $\mathcal{P}$ we introduce a duplicate period in $\mathcal{R}$ with the same expected demand, variance and holding cost. To avoid confusion, we will refer to the decision variables denoting the closing inventory level at period i in problem $\mathcal{R}$ as $\tilde{I'}_i$, to the binary variables as δ'_i, for all $i \in \{1, \ldots, M\}$ and to the demands as $\tilde{d}'_i$, for all $i \in \{1, \ldots, M\}$.

Step 2 In this step we apply a sound pre-processing method to the reduced problem instance $\mathcal{R}$ defined in the previous step.

Step 3 In this step we reflect the pruning done in the reduced instance back to the original instance. For each period $p \in \{1, \ldots, M\}$ of $\mathcal{R}$ that is the result of merging adjacent periods $\{i, \ldots, j\}, i < j$ of $\mathcal{P}$, we can update the domains of $\tilde{I}_t$ for all $i \leq t \leq j$ by enforcing the following constraints:

$$\tilde{I}_t = \begin{cases} \tilde{I}'_p & \text{if } t = j, \\ \tilde{I}'_p + \tilde{d}_j + \tilde{d}_{j-1} + \ldots + \tilde{d}_{t-1} & \text{if } i \leq t < j. \end{cases} \tag{22}$$

For any other period $p \in \mathcal{R}$ that does not represent merged periods and its corresponding period t in $\mathcal{P}$, we enforce that

$$\tilde{I}_t = \tilde{I}'_p. \tag{23}$$

These three steps compose the core of our algorithm. The following Theorem shows that such a filtering algorithm is sound.

Theorem 2 *We are given a problem instance $\mathcal{P}$ and a partial solution S for it, where $\exists \delta_i, i \in \{1, \ldots, N\}$ such that $\delta_i = 0$. By applying a sound pre-processing method (Step 2) to the reduced problem instance $\mathcal{R}$, obtained as described in Step 1, and by*

computing feasible values for decision variables $\tilde{I}_t$ in the original problem $\mathcal{P}$, as stated in Step 3, no value that is part of any optimal solution S^* with respect to the given partial assignments in S is pruned in the domain of $\tilde{I}_t$, $t \in \{1, \ldots, N\}$.

Proof We will now show that, under the given partial solution S, the reduced problem instance $\mathcal{R}$ is equivalent to the original problem $\mathcal{P}$ and that the reduction in the number of decision variables and constraints is a direct consequence of the linear dependencies induced by the current partial assignment for δ_t variables. This will establish the fact that any sound pre-processing method applied to $\mathcal{R}$ will produce a sound domain reduction in $\mathcal{P}$ when reflected by means of the proposed mapping that is built on these linear dependencies.

Let us consider the model above for our problem $\mathcal{P}$ that is defined by (6–9) and (10).

Consider $\mathcal{P}$ and a partial solution where $\exists k \in \{1, ..., N\}$ s.t. δ_k is set to 0. Let us consider the implications of this assignment in our model $\mathcal{P}$. This assignment affects the *inventory conservation constraints* (7) and obviously the *replenishment decisions* (8), the *constraints that enforce buffer stocks* (9) and the *objective function* (6).

Effects on the replenishment decision and on the inventory conservation constraints Since $\delta_k = 0$, constraint (7) for $t = k$ can be tightened because of (8) as follows:

$$\tilde{I}_k + \tilde{d}_k - \tilde{I}_{k-1} = 0, \tag{24}$$

then, by using $\tilde{I}_{k-1} + \tilde{d}_{k-1} - \tilde{I}_{k-2} \geq 0$ (that is constraint (7) for $t = k - 1$) and (24), we have

$$\tilde{I}_k + \tilde{d}_k + \tilde{d}_{k-1} - \tilde{I}_{k-2} \geq 0. \tag{25}$$

Notice that constraint (8) for $t = k$ is now redundant, since we assume that $\delta_k = 0$. Furthermore by following a reasoning similar to the one used to derive (25), (8) for $t = k - 1$ can be replaced by the following constraint

$$\tilde{I}_k + \tilde{d}_k + \tilde{d}_{k-1} - \tilde{I}_{k-2} > 0 \rightarrow \delta_{k-1} = 1. \tag{26}$$

Effects on the constraints that enforce buffer stocks Let us consider now the implications of constraint (24) on the buffer stock levels. When $t = k - 1$ in constraint 9 we can write

$$\tilde{I}_k + \tilde{d}_k \geq b \left(\max_{j \in \{1,\ldots,k-1\}} j \cdot \delta_j, k - 1 \right). \tag{27}$$

Also notice that for $t = k$

$$\tilde{I}_k \geq b \left(\max_{j \in \{1,\ldots,k\}} j \cdot \delta_j, k \right) \tag{28}$$

and since $\delta_k = 0$, (28) can be rewritten as

$$\tilde{I}_k \geq b \left(\max_{j \in \{1,\ldots,k-1\}} j \cdot \delta_j, k \right). \tag{29}$$

Since the buffer stock level $b(i, j)$ is an increasing function of the number of periods as shown in [23], it is easy to see that

$$\tilde{I}_k \geq b\left(\max_{j\in\{1,\ldots,k-1\}} j\cdot\delta_j, k\right) \geq b\left(\max_{j\in\{1,\ldots,k-1\}} j\cdot\delta_j, k-1\right), \tag{30}$$

it follows that (27) (that is constraint (9) for $t = k-1$) becomes redundant.

Effects on the objective function We now consider the implications of constraint (24) on the objective function. Since $\delta_k = 0$ the fixed ordering cost component for period k is zero. By applying constraint (24) we obtain the following new objective function

$$\min \quad E\{TC\} = \sum_{t=1,t\neq k}^{N} a\delta_t + \sum_{t=1,t\neq k-1}^{N} h\tilde{I}_t + h\left(\tilde{I}_k + \tilde{d}_k\right). \tag{31}$$

We can see that we no longer have a holding cost component for period $k-1$, while the holding cost for period k is now doubled, since we can ignore the constant term $h\cdot\tilde{d}_k$.

Every implication of (24) in the whole model has been considered, therefore we can rewrite

$$h\tilde{d}_k + \min \quad E\{TC\} = \sum_{t=1,t\neq k}^{N} a\delta_t + \sum_{t=1,t\neq k-1}^{N} h\tilde{I}_t + h\tilde{I}_k \tag{32}$$

subject to,

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} \geq 0 \qquad t = 1,\ldots,N; t\neq k-1; t\neq k \tag{33}$$

$$\tilde{I}_k + \tilde{d}_k + \tilde{d}_{k-1} - \tilde{I}_{k-2} \geq 0 \tag{34}$$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} > 0 \Rightarrow \delta_t = 1 \qquad t = 1,\ldots,N; t\neq k-1; t\neq k \tag{35}$$

$$\tilde{I}_k + \tilde{d}_k + \tilde{d}_{k-1} - \tilde{I}_{k-2} > 0 \Rightarrow \delta_{k-1} = 1 \tag{36}$$

$$\tilde{I}_t \geq b\left(\max_{j\in\{1..t\}} j\cdot\delta_j, t\right) \qquad t = 1,\ldots,N; t\neq k-1 \tag{37}$$

$$\tilde{I}_t \in \mathbb{Z}^+ \cup \{0\} \qquad t = 1,\ldots,N; t\neq k-1 \tag{38}$$

$$\delta_t \in \{0,1\} \qquad t = 1,\ldots,N; t\neq k. \tag{39}$$

To summarize, we showed that constraint (7) for $t = k-1$ and $t = k$ can be expressed by (25), and similarly constraint (8) for $t = k-1$ and $t = k$ can be expressed by (26). Both these new constraints (25, 26) are independent of $\tilde{I}_{k-1}$. Constraint (9) for $t = k-1$ becomes redundant. The new objective function (31) reflects the consequences of constraint (24) and is independent of decision variable $\tilde{I}_{k-1}$. Therefore the whole model is now independent of decision variable $\tilde{I}_{k-1}$, whose value is a function of $\tilde{I}_k$ (24).

Since the last model is independent of $\tilde{I}_{k-1}$ and δ_k, we now reduce it to an $(N-1)$-period model $\mathcal{R}$ through a change of variables, by merging periods $k-1$ and k and

realizing the whole demand $\tilde{d}'_{k^*} = \tilde{d}_k + \tilde{d}_{k-1}$ in the new period k^*, where k^* covers the span $\{k-1, k\}$. In such a new model $\mathcal{R}$ the demand $\tilde{d}'_t$ in the other periods $t \in \{1, ..., k^*-1, k^*+1, ..., N-1\}$ is mapped as follows:

$$\tilde{d}'_t = \begin{cases} \tilde{d}_t, & t \in \{1, ..., k-2\} \\ \tilde{d}_{t+1}, & t \in \{k, ..., N-1\}. \end{cases}$$

Since the demand in periods k and $k-1$ of $\mathcal{P}$ is assumed to be normally distributed, the variance for the demand in the new period k^* of $\mathcal{R}$ is

$$\sigma'_{k^*} = \sqrt{\sigma_k^2 + \sigma_{k-1}^2}.$$

$\tilde{I}'_{k^*}$ in $\mathcal{R}$, that is the closing inventory levels in the new model, can be related to the respective closing inventory levels of periods k and $k-1$ in $\mathcal{P}$ using $\tilde{I}_k = \tilde{I}'_{k^*}$ and $\tilde{I}_{k-1} = \tilde{I}'_{k^*} + \tilde{d}_k$, which follow from (24) and the definition of k^*. The other closing inventory levels are mapped as follows:

$$\tilde{I}'_t = \begin{cases} \tilde{I}_t, & t \in \{1, ..., k-2\} \\ \tilde{I}_{t+1}, & t \in \{k, ..., N-1\}. \end{cases}$$

Notice that we only assumed $\delta_k = 0$, so $N-1$ binary decision variables are still unassigned. Therefore we have $\delta'_{k^*} = \delta_{k-1}$ (26) and the following mapping for the remaining variables:

$$\delta'_t = \begin{cases} \delta_t, & t \in \{1, ..., k-2\} \\ \delta_{t+1}, & t \in \{k, ..., N-1\}, \end{cases}$$

where δ'_t are the binary decision variables in $\mathcal{R}$. Equation (31) states that in order to get a model equivalent to the initial one, we must apply a holding cost of $2h$ for the new period k^* in the objective function.

The last model presented can be therefore rewritten in terms of the new decision variables defined by this mapping. The resulting problem instance is $\mathcal{R}$

$$E\{TC\} = h\tilde{d}_k + \min \sum_{t=1}^{N-1} a\delta'_t + \sum_{t=1}^{N-1} h\tilde{I}'_t + h\tilde{I}'_{k^*} \tag{40}$$

subject to

$$\tilde{I}'_t + \tilde{d}'_t - \tilde{I}'_{t-1} \geq 0 \qquad t = 1, \ldots, N-1 \tag{41}$$

$$\tilde{I}'_t + \tilde{d}'_t - \tilde{I}'_{t-1} > 0 \Rightarrow \delta'_t = 1 \qquad t = 1, \ldots, N-1 \tag{42}$$

$$\tilde{I}'_t \geq b \left(\max_{j \in \{1..t\}} j \cdot \delta'_j, t \right) \qquad t = 1, \ldots, N-1 \tag{43}$$

$$\tilde{I}'_t \in \mathbb{Z}^+ \cup \{0\}, \quad \delta'_t \in \{0, 1\} \qquad t = 1, \ldots, N-1. \tag{44}$$

Table 6 Reduced problem instance built as described in Step 1

t	1	2	3	4	5	6	7	8	9
$i, \ldots, j$	1, 2	3	4, 5	6, 7	8, 9	10	11, 12	13	14, 15
$\tilde{d}_t$	73	128	208	208	192	88	94	181	96
σ_t	24.3	42.6	49.3	60.6	55.4	29.3	22.5	60.3	23.5
h_t	2	1	2	2	2	1	2	1	2
t	10	11	12	13	14				
$i, \ldots, j$	16	17, 18, 19	20, 21	22	23, 24				
$\tilde{d}_t$	88	52	209	190	195				
σ_t	29.3	13.7	64.2	63.3	55.3				
h_t	1	3	2	1	2				

For every period t in the new instance $\mathcal{R}$, $i, \ldots, j$ denotes the span covered in the original problem $\mathcal{P}$

It is trivial to recursively extend this reasoning to the case of consecutive periods with δ_k set to zero. This process necessarily ends when we reach an $i < k$ where $\delta_i = 1$ or $\delta_i \in \{0, 1\}$. Furthermore $\delta_1 = 1$, since without loss of generality we assume an initial null inventory and an initial demand greater than zero, therefore we always fix a replenishment in the first period. □

3.2.1 Example

We now refer to the same instance analyzed for the example in Section 3.1. When the partial solution given in Table 4 is considered, a reduced problem instance can be built as described in Step 1. This instance is shown in Table 6. We applied pre-processing method I in [23] to this instance as stated in Step 2. Note that this is equivalent to applying our cost-based filtering method presented in Section 3.1 when in the given partial solution no decision variable has been assigned to a value. The reduced domains are shown in Table 7. From the reduced domains in Table 7, by applying Step 3, we can compute the reduced domain for the original problem instance. These domains are shown in Table 8. The two presented methods are incomparable, in fact this method prunes more values in period 6 while the former one prunes more values in period 16.

Table 7 Effect of pre-processing method I in [23] on the smaller instance with merged periods, underlined figures are closing inventory levels of the optimal policy

i	$Dom(\tilde{I}'_i)$	i	$Dom(\tilde{I}'_i)$
1 : {1, 2}	{40}	8 : {13}	{99}
2 : {3}	{70}	9 : {14, 15}	{39}
3 : {4, 5}	{81}	10 : {16}	{88, 143}
4 : {6, 7}	{100}	11 : {17, 18, 19}	{23, 36, 91}
5 : {8, 9}	{91}	12 : {20, 21}	{106}
6 : {10}	{88}	13 : {22}	{104}
7 : {11, 12}	{37}	14 : {23, 24}	{91}

Table 8 Reduced domains of the original instance obtained through the mapping proposed, underlined figures are closing inventory levels of the optimal policy

i	$Dom(\tilde{I}_i)$	i	$Dom(\tilde{I}_i)$
1	{40}	13	{99}
2	{40}	14	{73}
3	{70}	15	{39}
4	{173}	16	{88, 143}
5	{81}	17	{73, 86, 141}
6	{128}	18	{63, 76, 131}
7	{100}	19	{23, 36, 91}
8	{119}	20	{123}
9	{91}	21	{106}
10	{88}	22	{104}
11	{94}	23	{123}
12	{37}	24	{91}

4 Cost-Based Filtering by Relaxation

The CP model as described so far suffers from a lack of tight bounds on the objective function. In this section we recall a relaxation for our model originally proposed by Tarim in [18]. By means of this relaxation we will introduce a novel approach to compute a locally optimal solution or a valid lower bound at each node of the search tree.

It should be noted that the relaxation as presented in [18] does not take into account a given partial solution if this is available. As we will show this extension is not trivial, especially if we aim to take into account a partial assignment involving both δ_t and $\tilde{I}_t$ decision variables.

Given a problem instance, Tarim's approach adopts a greedy algorithm to solve a relaxed problem instance. This way a replenishment plan (assignment for the δ_t and I_t variables) is generated. Once this replenishment plan is available, it is possible to characterize if it is also feasible with respect to the original problem. If so, the respective computed cost is optimal for the original problem. Otherwise, if the replenishment plan is infeasible with respect to the original problem, the computed cost is a valid lower bound for the optimal solution cost of the original problem.

4.1 Tarim's Relaxation

We shall now describe Tarim's relaxation in details. The core observation consists in the fact that the CP model proposed in Section 2 can be reduced to a **shortest path problem** if we relax inventory conservation constraints (7, 8) for replenishment periods only. That is for each possible pair of replenishment cycles $\langle T(i, k-1), T(k, j)\rangle$ where $i, j, k \in \{1, \ldots, N\}$ and $i < k \leq j$, we do not consider the relationship between the opening inventory level of $T(k, j)$ and the closing inventory level of $T(i, k-1)$. This corresponds to allowing negative replenishments (Fig. 4a), or the ability to sell stock back to the supplier. Since the inventory conservation constraint is now relaxed between replenishment cycles, each replenishment cycle can be now treated independently and its cost can be computed *a priori*. In fact, given a replenishment cycle $T(i, j)$, we recall that $b(i, j)$, as defined above, denotes the minimum buffer stock level required to satisfy a given service level constraint during the replenishment

cycle $T(i, j)$. It directly follows that $\tilde{I}_j = b(i, j)$. Furthermore for each period $t \in \{i, \ldots, j-1\}$ the expected closing-inventory-level is $\tilde{I}_t = b(i, j) + \sum_{k=t+1}^{j} d_k$. Since all the $\tilde{I}_t$ for $t \in \{i, \ldots, j\}$ are known it is easy to compute the expected total cost for $T(i, j)$, which is by definition the sum of the ordering cost and of the holding cost components, $a + h \sum_{t=i}^{j} \tilde{I}_t$. We now have a set $\mathcal{S}$ of $N(N+1)/2$ possible different replenishment cycles and the respective costs. Our new problem is to find an optimal set $\mathcal{S}^* \subset \mathcal{S}$ of consecutive disjoint replenishment cycles that covers our planning horizon at the minimum cost.

It should be noted that, from the characterization of the optimal policy for the deterministic inventory/production problem given by Wagner and Whitin [25], the optimal solution of this relaxation is always feasible for the original problem if buffer stocks are all zero and therefore we are solving a deterministic problem. In fact we recall that, as stated in [25] in the search for the optimal policy for the deterministic production/inventory problem it is sufficient to consider programs in which at period t one does not both place an order and bring in inventory (i.e. zero-inventory ordering property). It directly follows that every relaxed inventory conservation constraint is trivially satisfied under a deterministic setting, as in an optimal solution the closing inventory level at the end of each replenishment cycle must be zero.

4.2 Tarim's Relaxation as a Shortest Path Problem

We shall now show that the optimal solution to this relaxation is given by the shortest path in a graph from a given initial node to a final node where each arc represents a replenishment cycle cost. If N is the number of periods in the planning horizon of the original problem, we introduce $N+1$ nodes. Since we assume, without loss of generality, that an order is always placed at period 1, we take node 1, which represents the beginning of the planning horizon, as the initial node. Node $N+1$ represents the end of the planning horizon. For each possible replenishment cycle $T(i, j-1)$ such that $i, j \in \{1, \ldots, N+1\}$ and $i < j$, we introduce an arc (i, j) with associated cost $c(i, j-1)$. Since we are dealing with a one-way temporal feasibility problem [25], when $i \geq j$, we introduce no arc. The connection matrix for such a graph, of size $N \times (N+1)$, can be built as shown in Table 9. By construction the cost of the shortest path from node 1 to node $N+1$ in the given graph is a valid lower bound for the original problem, as it is a solution of the relaxed problem.

Table 9 Shortest Path Problem Connection matrix

	1	2	…	j	…	$N+1$
1	–	$c(1,1)$	…	$c(1, j-1)$	…	$c(1, N)$
⋮	–	–	⋱	⋮	⋱	⋮
i	–	–	–	$c(i, j-1)$	…	$c(i, N)$
⋮	–	–	–	–	⋱	⋮
N	–	–	–	–	–	$c(N, N)$

4.2.1 Solution Mapping

It is easy to map the optimal solution for the relaxed problem, that is the set of arcs participating to the shortest path, to a solution for the original problem by noting that each arc (i, j) represents a replenishment cycle $T(i, j-1)$. By the definition of replenishment cycle $T(i, j-1)$, $\delta_i = 1$ and $\delta_t = 0$, for $t = i+1, \ldots, j-1$. The set of arcs in the optimal path uniquely identifies a set of disjoint replenishment cycles, that is a replenishment plan (assignment for δ_t decision variables). Furthermore for each period $t \in \{i, \ldots, j-1\}$ in cycle $T(i, j-1)$ we already showed that all the expected closing-inventory-levels $\tilde{I}_t$, $t \in \{i, \ldots, j-1\}$, are known. This produces a complete assignment for decision variables in our model. The feasibility of such an assignment with respect to the original problem can be checked by verifying that it satisfies every relaxed constraint, that is no negative expected order quantity is scheduled.

4.2.2 Shortest Path Algorithm

To find a shortest path in the given graph we use a modified Dijkstra's algorithm that finds a shortest path in $O(n^2)$ time, where n is the number of nodes in the graph. Details on efficient implementations of Dijkstra's algorithm can be found in [15]. Usually Dijkstra's algorithm [15] does not apply any specific rule for labeling when ties are encountered in sub-path lengths. This non-deterministic labeling may produce a loss of optimal solutions if decision variable domains are pre-processed as described in [23]. In fact pre-processing Method I in [23] relies upon an upper-bound for optimal replenishment cycle length. When a replenishment period $i \in \{1, ..., N\}$ is considered, it looks for the lowest $j \in \{i, \ldots, N\}$ after which it is no longer optimal to schedule the next replenishment. This means that, if more policies that share the same expected cost exist, only the one that has shorter, and obviously more, replenishment cycles will be preserved by Method I. Therefore, when the algorithm is implemented in this filtering approach, we need to introduce a specific rule for node selection in order to make sure that, when more optimal policies exist, our modified algorithm will always find the one that has the highest possible number of replenishment cycles (i.e. the shortest path with the highest possible number of arcs). Since there is a complete order among nodes, we can easily implement this rule in the labeling action by always choosing as ancestor the node that minimizes the distance from the source and that has the highest index. The pseudo-code for the proposed modified Dijkstra's algorithm can be found in Appendix 7.3.

4.3 Cost-Based Filtering

So far we described a known possible way to relax the CP model proposed in Section 2. We also proposed a novel Dijkstra's algorithm implementation that makes the relaxation in [18] compatible with the pre-processing methods in [23]. The relaxation described can be seen as a state space relaxation, where we define a new problem with a number of states polynomially bounded in the original problem input. A lower bound for the optimal solution cost is then obtained by solving a Shortest Path Problem in the state space graph. We will now show a novel approach to exploit this lower bound in an *optimization oriented global constraint*. A detailed discussion on state space relaxation and optimization oriented global constraints can be found in [10].

4.3.1 Partial Assignments for δ_k Decision Variables

$\delta_k = 0$ Let us consider the graph built as described in Tarim's relaxation. If in a given partial solution a decision variable δ_k, $k \in \{1, \ldots, N\}$ has been already set to 0, then we can remove from the graph every inbound arc to node k and every outbound arc from node k. This prevents node k from being part of the shortest path, and hence prevents period k from being a replenishment period. By applying Dijkstra's algorithm to this modified graph the cost of the shortest path will provide a valid lower bound for the cost of an optimal solution incorporating the decision $\delta_k = 0$. Furthermore, as seen above, Dijkstra's algorithm will also provide an assignment for decision variables. If this assignment is feasible for the original problem, then it is optimal with the respect to the decision $\delta_k = 0$.

$\delta_k = 1$ On the other hand, if $\delta_k = 1$ then we split the planning horizon into two at period k, thus obtaining two new subproblems $\{i, \ldots, k-1\}$ and $\{k, \ldots, j\}$. We can then separately solve these two subproblems by applying Tarim's relaxation to each of them. Note that the action of splitting the time span is itself a relaxation; in fact it means overriding constraints (7, 8) for $t = k$. It follows that the overall cost obtained by summing the cost of the solution found for each relaxed subproblem is again a valid lower bound for an optimal solution of the original problem that incorporates the decision $\delta_k = 1$. Furthermore both the subproblems identified, $\{i, \ldots, k-1\}$ and $\{k, \ldots, j\}$, when relaxed and solved as explained above, will not only provide a cost component, but also a partial assignment for the original problem — that is an assignment for δ_t and $\tilde{I}_t$ variables, respectively for $t \in \{i, \ldots, k-1\}$ and $t \in \{k, \ldots, j\}$ — as seen above. A complete assignment, feasible or infeasible, for the original problem can be obtained by merging these two partial assignments, that is by considering the complete assignment over $\{i, \ldots, j\}$ defined by the solutions of the two subproblems, that respectively assign values to decision variables in $\{i, \ldots, k-1\}$ and in $\{k, \ldots, j\}$. In the following paragraph we characterize when this assignment is feasible for the original problem.

Let A denote the subproblem $\{i, \ldots, k-1\}$ and B the subproblem $\{k, \ldots, j\}$. We now focus on subproblem A but the reasoning can be repeated for subproblem B. We apply Tarim's relaxation to subproblem A, possibly taking into account decisions $\delta_t = 0$, for $t = \{i, \ldots, k-1\}$, and we solve it by means of dynamic programming. The solution for this relaxed subproblem provides a cost c_A and an assignment for decision variables δ_t and $\tilde{I}_t$, for $t = \{i, \ldots, k-1\}$. For each period $t = \{i, \ldots, k-1\}$ it is easy to verify if the solution found satisfies every relaxed inventory conservation constraint between replenishment cycles. In fact we just need to check that a negative order quantities is never scheduled. If every relaxed constraint is satisfied, the computed cost c_A and the decision variable assignment are optimal with respect to subproblem A, otherwise the computed cost c_A provides a lower bound for the cost of an optimal policy with respect to subproblem A.

Trivially if the assignment found for subproblem A (B) does not satisfy some relaxed constraints, it follows that the overall cost $c_A + c_B$ is a lower bound for the cost of the optimal solution for the original problem over $\{i, \ldots, j\}$ with respect to the decision $\delta_k = 1$.

Let us suppose instead that the assignments found for subproblems A and B satisfy every relaxed constraint. The costs c_A and c_B are therefore optimal with

respect to subproblems A and B. We now want to characterize when the complete assignment defined by the assignment for decision variables in $\{i, \ldots, k-1\}$ (solution of subproblem A) and the assignment for decision variables in $\{k, \ldots, j\}$ (solution of subproblem B) is feasible and optimal with respect to the original problem over $\{i, \ldots, j\}$ when $\delta_k = 1$.

Feasible Complete Assignment, $\delta_k = 1$ Let $R_k = \tilde{I}_k + \tilde{d}_k$ denote the required minimum opening inventory level in period k according to the solution found for subproblem B. When the assignments for the two subproblems are both feasible with respect to the original model, that is they do not schedule negative order quantities, and the condition

$$\tilde{I}_{k-1} \leq R_k \tag{45}$$

is satisfied (Fig. 5), the overall assignment over $\{i, \ldots, j\}$ defined by the assignment for decision variables in $\{i, \ldots, k-1\}$ (solution of subproblem A) and the assignment for decision variables in $\{k, \ldots, j\}$ (solution of subproblem B) is feasible and optimal for the original problem over $\{i, \ldots, j\}$ when $\delta_k = 1$. It directly follows that the overall cost $c_A + c_B$, obtained by summing the cost of each subproblem solution, is also optimal for the original problem over $\{i, \ldots, j\}$ when $\delta_k = 1$.

Infeasible Complete Assignment, $\delta_k = 1$: Otherwise, when condition (45) is not met (Fig. 6), the cost $c_A + c_B$ is a lower bound for the optimal solution cost of the original problem over $\{i, \ldots, j\}$ when $\delta_k = 1$.

We have shown how to act when each of the possible cases, $\delta_k = 1$ and $\delta_k = 0$, is encountered. It is now possible at any point of the search in the decision tree to apply this relaxation and compute a valid lower bound or a solution that is optimal with respect to the given partial assignment.

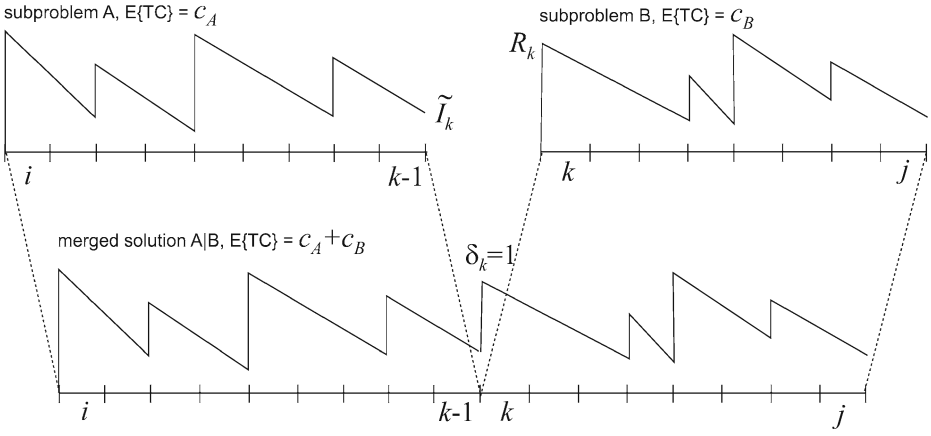

Fig. 5 Condition (45) is met between the solution of subproblem A and that of subproblem B. The overall solution defined by the assignment for decision variables in $\{i, \ldots, k-1\}$ (solution of subproblem A) and the assignment for decision variables in $\{k, \ldots, j\}$ (solution of subproblem B) is feasible and optimal with respect to the original problem over $\{i, \ldots, j\}$

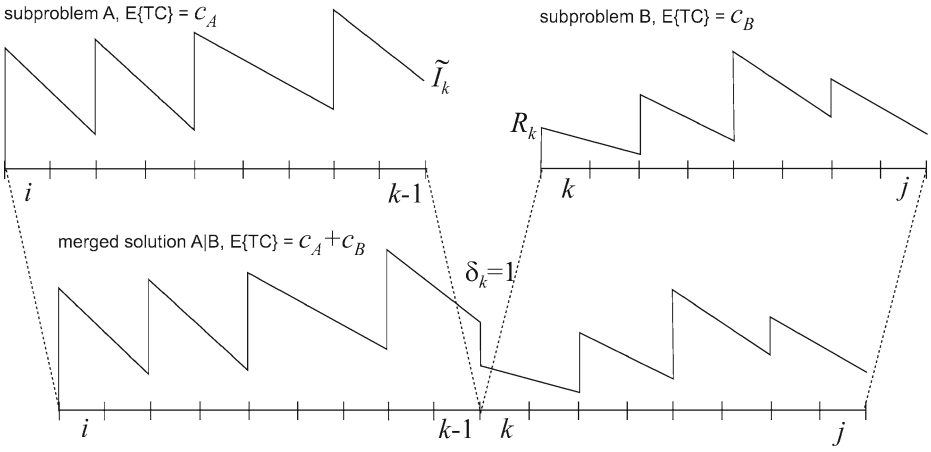

Fig. 6 Condition (45) is not met between the solution of subproblem A and that of subproblem B. The overall solution defined by the assignment for decision variables in $\{i, \ldots, k-1\}$ (solution of subproblem A) and the assignment for decision variables in $\{k, \ldots, j\}$ (solution of subproblem B) is not feasible with respect to the original problem over $\{i, \ldots, j\}$. The computed cost $c_A + c_B$ is a lower bound for the optimal solution cost of the original problem over $\{i, \ldots, j\}$

4.3.2 Partial Assignments for $\tilde{I}_k$ Decision Variables

It is also possible to extend this cost-based filtering method by considering not only the δ_k variable assignments, but also the $\tilde{I}_k$ variable assignments. In fact, when the cost of a given replenishment cycle $T(i, j-1)$ [arc (i, j) in the matrix] is computed, it is also possible to consider the current assignments for the closing inventory levels $\tilde{I}_k$ in the periods of this cycle. Since all the closing inventory levels of the periods within a replenishment cycle are linearly dependent ($\delta_k = 0 \rightarrow \tilde{I}_k + \tilde{d}_k - \tilde{I}_{k-1}$=0), given an assignment for a decision variable $\tilde{I}_k$ we can easily compute all the other closing inventory levels in the cycle by using $\tilde{I}_k - \tilde{d}_k - \tilde{I}_{k-1} = 0$, which is the inventory conservation constraint when no order is placed in period k. When the closing inventory levels in a replenishment cycle $T(i, j-1)$ are known it is easy to compute the overall cost associated to this cycle as seen above. We can therefore associate to arc (i, j) the highest cost that is produced by a current assignment for the closing inventory levels $\tilde{I}_k$, $k \in \{i, \ldots, j-1\}$. If no variable has been assigned yet, we simply use the minimum possible cost $c(i, j-1)$ which we defined above.

5 Experimental Results

This section is organized as follows. Firstly we will consider a particularly hard instance built by adding random elements on a seasonal demand. We will use this instance to gauge the effectiveness of each filtering method we proposed. Furthermore we will also analyze how the proposed methods perform when they are combined together. Secondly we will compare our method with the state-of-the-art results presented in [23]. Thirdly we will present extensive tests to show the

effectiveness of our domain filtering methods with respect to a pure CP approach enhanced with the pre-processing methods presented by Tarim and Smith.

All experiments presented here were performed on an Intel(R) Centrino(TM) CPU 1.50GHz with 500Mb RAM. The solver used for our test is Choco [12], an open-source solver developed in Java.

The heuristic used for the selection of the variable is the usual min-domain/max-degree heuristic. Decision variables have different priorities in the heuristic: the δ_k have higher priority than the $\tilde{I}_k$. The value selection heuristic chooses values in increasing order of size.

In what follows we will refer to the filtering methods presented as follows: Method I (Section 3.1), Method II (Section 3.2), Method III (Section 4). Since Method II can be in principle applied in conjunction with any sound domain reduction method, in all the experiments here presented the domain reduction applied with Method II is pre-processing method II presented in Tarim and Smith [23]. We only apply one pre-processing method since experimentally no improvement was noticed in term of explored nodes and running time when both the methods were used in conjunction as shown in [23].

5.1 Effectiveness of Filtering Methods

A single problem is considered and the period demands are listed in Fig. 7. In each test we assume an initial null inventory level and a normally distributed demand for every period with a coefficient of variation $\sigma_t/\tilde{d}_t = 1/3$ for each $t \in \{1, \ldots, N\}$, where N is the length of the planning horizon considered. The ordering cost ranges in the following set {40, 80, 160, 320}. The holding cost is 1. Our tests consider two different service levels $\alpha = 0.95$ ($z_{\alpha=0.95} = 1.645$) and $\alpha = 0.99$ ($z_{\alpha=0.99} = 2.326$). In Table 10 we compare the effectiveness of each filtering method, when used to augment the CP model enhanced by the pre-processing methods in [23]. The performances achieved by the CP approach enhanced with the pre-processing methods are shown in column "No Filt.". The performances achieved when the filtering methods are all added to the model are shown in column "Combined". In the presented table we can see that Method I and Method II do not perform well when they are used alone.

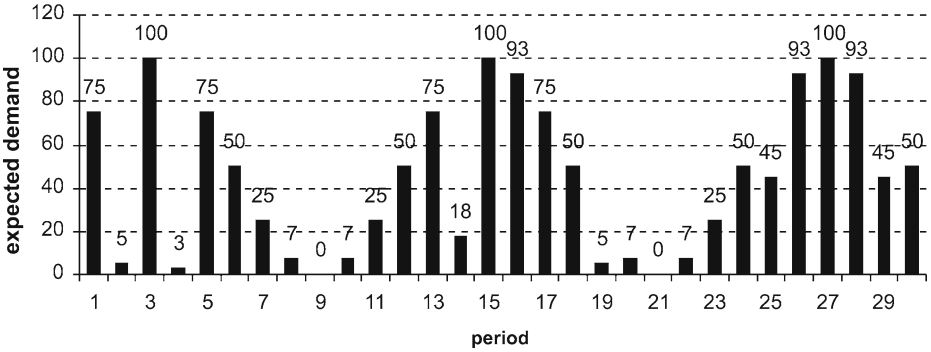

Fig. 7 Expected demand values

Table 10 Filtering methods compared in terms of explored nodes ("Nod") and run time in seconds ("Sec")

α	a	No Filt.		Method I		Method II		Method III		Combined	
		Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
0.95	40	127	1.85	96	1.64	96	1.43	120	1.30	70	1.12
	80	2994	30	1449	16	2586	23	82	1.02	63	0.97
	160	–	–	–	–	–	–	133	1.81	108	1.65
	320	–	–	–	–	–	–	4	0.09	4	0.09
0.99	40	261	3.27	198	4.24	202	2.52	253	2.84	165	2.57
	80	1234	11	611	7.54	1138	10.7	317	2.66	221	2.61
	160	–	–	–	–	–	–	168	2.15	84	1.31
	320	–	–	–	–	–	–	1	0.09	1	0.10

Symbol "–" means that an optimal solution has not be found within the given limit of 60 s

This is again due to the lack of good bounds during the search process. Method III instead is very effective even when it is used alone and especially for high ordering costs, when the contribution of the filtering due to the computed bounds is critical. Nevertheless when the three methods are combined for all the eight instances presented performances are improved both in terms of running time and explored nodes.

5.2 Comparison with State-of-the-Art Results

In this section we compare results obtained with our approach with the state-of-the-art results presented in [23].

A single problem is considered and the period demands are generated from seasonal data with no trend: $\tilde{d}_t = 50[1 + \sin(\pi t/6)]$. In addition to the "no trend" case (P1) we also consider three others:

(P2) positive trend case, $\tilde{d}_t = 50[1 + \sin(\pi t/6)] + t$
(P3) negative trend case, $\tilde{d}_t = 50[1 + \sin(\pi t/6)] + (52 - t)$
(P4) life-cycle trend case, $\tilde{d}_t = 50[1 + \sin(\pi t/6)] + \min(t, 52 - t)$

In each test we assume a coefficient of variation $\sigma_t/\tilde{d}_t = 1/3$ for each $t \in \{1, \ldots, N\}$, where N is the length of the considered planning horizon. As in Tarim and Smith tests are performed using two different ordering cost values $a \in \{400, 900\}$. The holding cost used in these tests is $h = 1$ per unit per period. Our tests consider a service levels $\alpha = 0.95$ ($z_{\alpha=0.95} = 1.645$).

In Table 11 we can observe the improvement of several orders of magnitude brought by our domain filtering techniques. Experiments in [23] employed OPL Studio 3.7 (ILOG Solver 6.0, ILOG Cplex 9.0) used with its default settings. Note that the hardware used for these experiments is comparable to the one used for ours.

Table 11 Comparison with the state-of-the-art results in [23] ("Tarim and Smith")

		$a = 400$				$a = 800$			
		Filt.		Tarim and Smith		Filt.		Tarim and Smith	
	Horizon	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
P1	50	1	0.30	–	–	3	0.10	–	–
	48	1	0.09	–	–	3	0.10	30795352	10100
	46	1	0.09	43721791	12200	3	0.09	8763280	2840
	44	1	0.09	36976882	9700	3	0.01	6896956	2110
P2	44	1	0.09	–	–	4	0.10	–	–
	42	1	0.09	–	–	4	0.10	60884565	15600
	40	1	0.29	–	–	4	0.17	22281926	5590
	38	1	0.09	35848309	6820	4	0.10	7978185	1880
P3	42	1	0.09	–	–	3	0.10	–	–
	40	1	0.09	–	–	3	0.10	55138095	13300
	38	1	0.09	61438266	11300	3	0.10	19600638	4510
	36	1	0.09	24256921	4150	3	0.10	6501541	1510
P4	44	1	0.09	–	–	4	0.09	–	–
	42	1	0.10	–	–	4	0.11	39668737	10700
	40	1	0.09	–	–	4	0.10	18004555	4690
	38	1	0.09	32076069	6680	4	0.09	6093007	1520

"Filt." indicates that Tarim and Smith's model is augmented with our filtering methods. Symbol "–" means that an optimal solution has not been found within the given limit of 5 h

5.3 More Extensive Tests

In this section we show the effectiveness of our approach by comparing the computational performance of the state-of-the-art CP model with that obtained by our approach.

We refer again to (P1), (P2), (P3) and (P4) as defined above. We performed tests using four different ordering cost values $a \in \{40, 80, 160, 320\}$ and two different $\sigma_t/\tilde{d}_t \in \{1/3, 1/6\}$. The planning horizon length takes even values in the range [24, 50] when the ordering cost is 40 or 80 and [14, 24] when the ordering cost is 160 or 320. The holding cost used in these tests is $h = 1$ per unit per period. Our tests also consider two different service levels $\alpha = 0.95$ ($z_{\alpha=0.95} = 1.645$) and $\alpha = 0.99$ ($z_{\alpha=0.99} = 2.326$).

In our test results a time of 0 means that the Dijkstra algorithm proved optimality at the root node. A header "Filt." means that we are applying our cost-based filtering methods, and "No Filt." means that we solve the instance using only the CP model and the pre-processing methods. Tables 12, 13, 14 and 15 compare the performance of the state-of-the-art CP model, implemented in Choco, with that of our new methods.

When a=320, and often when a=160, the Dijkstra algorithm proves optimality at the root node so the other reduction methods are not exploited during search. This is a direct consequence of the fact that under high ordering cost values it is extremely rare that a solution for the relaxed problem violates some inventory conservation constraint. In fact since placing an order is expensive the optimal solution will try to cover several periods with a single order. Such an order requires a high order-up-to-level that typically exceeds the expected closing-inventory-level of the previous

Table 12 Test set P1

a	N	$\sigma_t/d_t = 1/3$								$\sigma_t/d_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$				$\alpha = 0.95$				$\alpha = 0.99$			
		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.	
		Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
40	40	28	0.9	106	2.9	38	0.8	249	6.4	34	0.7	574	16	10	0.1	192	6.4
	42	28	0.6	95	2.8	38	0.8	233	5.9	34	0.7	582	14	10	0.1	196	5.4
	44	29	0.6	133	4.9	47	1.1	266	8.3	35	0.7	884	25	11	0.2	285	9.0
	46	30	0.6	192	7.8	64	1.6	484	18	45	1.0	3495	120	13	0.2	813	30
	48	43	0.9	444	19	72	2.1	1024	41	53	1.1	5182	190	13	0.2	1208	47
	50	43	1.0	444	20	72	2.2	1024	44	53	1.2	4850	200	13	0.2	1208	51
80	40	43	0.8	1742	78	13	0.2	557	15	16	0.2	9316	300	16	0.3	11276	440
	42	43	0.9	1703	60	13	0.2	530	13	17	0.3	17973	530	17	0.3	22291	690
	44	48	1.1	4810	210	14	0.2	980	25	19	0.4	38751	1400	20	0.4	50805	1600
	46	49	1.3	6063	340	16	0.3	2122	78	20	0.3	103401	4300	20	0.4	111295	4100
	48	67	2.0	20670	1400	17	0.3	5284	210	21	0.4	237112	12000	21	0.5	321998	15000
	50	67	2.2	18938	1300	17	0.4	5284	230	21	0.4	251265	13000	21	0.5	358174	17000
160	14	1	0.0	141	3.0	23	0.1	156	2.5	1	0.0	112	2.6	1	0.0	116	2.4
	16	1	0.0	277	9.0	35	0.2	182	5.1	1	0.0	238	6.7	1	0.0	235	6.8
	18	1	0.0	673	18	41	0.4	393	10	1	0.0	799	23	1	0.0	603	15
	20	1	0.0	3008	81	51	0.6	1359	21	1	0.0	2887	86	1	0.0	2820	75
	22	1	0.0	10620	260	57	0.6	7280	70	1	0.0	14125	380	1	0.0	10739	270
	24	1	0.0	61100	1500	153	1.8	31615	310	1	0.0	70996	1800	1	0.0	59650	1500
320	14	1	0.0	149	4.0	1	0.0	181	4.1	1	0.0	109	3.0	1	0.0	128	3.0
	16	1	0.0	335	11	1	0.0	361	12	1	0.0	246	8.7	1	0.0	284	9.3
	18	1	0.0	813	27	1	0.0	831	27	1	0.0	764	26	1	0.0	700	24
	20	1	0.0	2602	93	1	0.0	2415	81	1	0.0	2114	78	1	0.0	2291	82
	22	1	0.0	7434	260	1	0.0	7416	260	1	0.0	7006	260	1	0.0	6608	230
	24	1	0.0	49663	1600	1	0.0	49299	1500	1	0.0	39723	1400	1	0.0	43520	1500

Table 13 Test set P2

a	N	$\sigma_t/d_t = 1/3$								$\sigma_t/d_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$				$\alpha = 0.95$				$\alpha = 0.99$			
		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.	
		Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
40	40	4	0.1	7	0.1	7	0.1	8	0.1	12	0.2	23	0.4	4	0.1	12	0.2
	42	4	0.0	7	0.1	7	0.2	8	0.1	12	0.2	23	0.4	4	0.1	10	0.1
	44	4	0.1	7	0.1	7	0.2	8	0.1	12	0.2	23	0.5	4	0.1	10	0.2
	46	4	0.1	7	0.1	7	0.3	8	0.2	12	0.2	23	0.5	4	0.1	10	0.2
	48	4	0.1	7	0.2	7	0.2	8	0.2	12	0.3	23	0.5	4	0.1	10	0.2
	50	4	0.1	7	0.2	7	0.2	8	0.2	12	0.3	23	0.6	4	0.1	10	0.2
80	40	18	0.3	4592	14	15	0.3	275	8.3	37	0.7	2565	63	32	0.7	1711	44
	42	18	0.4	4866	13	15	0.4	283	6.7	37	0.8	3027	67	32	0.7	2043	47
	44	18	0.4	5091	15	15	0.4	280	7.9	40	0.9	6024	160	37	0.9	4299	120
	46	23	0.5	5291	45	17	0.5	545	16	47	1.3	14058	410	39	1.1	10311	290
	48	23	0.6	5544	51	17	0.5	545	17	47	1.4	14058	440	39	1.2	10311	310
	50	23	0.6	5850	51	17	0.5	545	18	47	1.5	14079	470	39	1.3	10347	330
160	14	1	0.0	166	3.6	19	0.1	84	1.0	1	0.0	148	2.9	1	0.0	171	3.4
	16	30	0.2	154	4.3	19	0.1	65	1.2	1	0.0	329	8.6	1	0.0	383	10
	18	58	0.4	485	11	34	0.3	174	2.9	1	0.0	948	23	1	0.0	1056	27
	20	37	0.3	2041	35	37	0.4	707	7.9	1	0.0	4228	110	1	0.0	4730	120
	22	48	0.4	9534	120	32	0.3	2954	28	1	0.0	20438	500	1	0.0	23675	530
	24	65	0.7	30502	360	41	0.4	7787	87	1	0.0	71514	1800	1	0.0	83001	1900
320	14	1	0.0	238	5.6	1	0.0	278	6.4	1	0.0	166	3.7	1	0.0	191	4.5
	16	1	0.0	505	17	1	0.0	423	13	1	0.0	387	11	1	0.0	452	14
	18	1	0.0	1447	49	1	0.0	1208	40	1	0.0	1100	34	1	0.0	1268	40
	20	1	0.0	4792	156	1	0.0	4219	150	1	0.0	3992	130	1	0.0	4476	150
	22	1	0.0	20999	660	1	0.0	20417	610	1	0.0	15983	520	1	0.0	18663	600
	24	1	0.0	102158	3200	1	0.0	90398	2600	1	0.0	75546	2500	1	0.0	88602	2800

Table 14 Test set P3

a	N	$\sigma_t/d_t = 1/3$								$\sigma_t/d_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$				$\alpha = 0.95$				$\alpha = 0.99$			
		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.	
		Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
40	40	2	0.0	5	0.0	2	0.0	4	0.0	6	0.1	9	0.2	2	0.0	5	0.0
	42	2	0.0	5	0.0	2	0.0	4	0.0	6	0.1	9	0.2	2	0.0	5	0.0
	44	3	0.0	7	0.1	3	0.0	6	0.1	7	0.1	14	0.3	3	0.0	7	0.1
	46	4	0.1	15	0.3	6	0.1	13	0.3	11	0.2	40	1.1	4	0.1	14	0.3
	48	4	0.1	15	0.3	6	0.1	13	0.3	14	0.3	56	1.8	4	0.1	25	0.6
	50	4	0.1	15	0.3	6	0.2	13	0.3	14	0.4	56	1.9	4	0.1	25	0.5
80	40	22	0.4	349	10	6	0.1	55	1.2	19	0.3	722	19	9	0.2	310	8.7
	42	22	0.4	354	8.6	6	0.1	53	1.2	22	0.4	1436	35	9	0.2	315	7.5
	44	24	0.6	571	17	7	0.1	88	2.4	27	0.6	3461	110	13	0.3	1053	31
	46	29	0.8	2787	90	9	0.2	258	8.1	36	0.9	10612	360	16	0.4	2881	94
	48	38	1.1	6803	240	9	0.2	385	12	47	1.3	28334	1100	22	0.6	7790	280
	50	38	1.1	6575	240	9	0.2	385	13	47	1.6	26280	1100	22	0.6	7371	280
160	14	7	0.0	23	0.2	8	0.0	16	0.1	15	0.1	53	0.6	9	0.0	29	0.3
	16	7	0.0	19	0.2	8	0.0	18	0.2	15	0.1	52	0.8	9	0.0	26	0.4
	18	9	0.1	42	0.5	10	0.0	30	0.3	21	0.1	149	2.2	12	0.1	87	1.2
	20	11	0.1	137	1.3	11	0.1	70	0.7	25	0.2	512	6.1	16	0.2	310	3.5
	22	21	0.2	376	4.0	21	0.2	221	2.3	31	0.4	1848	17	17	0.2	938	9.4
	24	32	0.4	995	11	30	0.4	543	6.3	43	0.5	4784	54	23	0.2	2471	30
320	14	1	0.0	253	4.2	1	0.0	232	3.8	1	0.0	310	4.4	1	0.0	217	3.4
	16	1	0.0	518	10	1	0.0	518	10	1	0.0	707	13	1	0.0	465	8.5
	18	1	0.0	1475	35	1	0.0	1170	26	1	0.0	1995	43	1	0.0	1416	33
	20	1	0.0	5342	140	1	0.0	4059	95	1	0.0	6678	160	1	0.0	5232	140
	22	1	0.0	21298	550	1	0.0	18065	440	1	0.0	25522	640	1	0.0	21756	560
	24	1	0.0	86072	2300	1	0.0	70969	1800	1	0.0	101937	2800	1	0.0	91358	2400

Table 15 Test set P4

a	N	$\sigma_t/d_t = 1/3$								$\sigma_t/d_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$				$\alpha = 0.95$				$\alpha = 0.99$			
		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.	
		Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
40	40	5	0.1	21	0.3	10	0.2	24	0.5	25	0.5	89	1.8	5	0.1	33	0.5
	42	5	0.1	18	0.3	10	0.2	21	0.4	25	0.5	91	2.0	5	0.1	31	0.5
	44	6	0.1	32	0.7	11	0.2	37	0.9	28	0.6	152	3.6	6	0.1	51	1.0
	46	7	0.1	83	2.0	16	0.4	93	2.4	42	1.0	474	12	7	0.1	126	2.8
	48	7	0.1	83	2.2	16	0.4	93	2.6	50	1.2	735	20	7	0.2	188	4.5
	50	7	0.1	83	2.3	16	0.4	93	2.8	50	1.4	735	22	7	0.2	188	4.9
80	40	39	0.7	1372	39	16	0.4	433	12	40	0.7	5098	130	33	0.7	2133	54
	42	39	0.8	1673	39	16	0.4	438	10	46	1.0	11452	270	33	0.8	2513	58
	44	43	1.0	2907	74	17	0.5	716	22	56	1.4	27184	780	46	1.3	8776	240
	46	51	1.3	13306	380	21	0.6	2178	73	75	1.9	77332	2600	55	1.6	22582	690
	48	69	1.8	32709	1000	21	0.6	3223	120	100	2.8	202963	7500	73	2.2	60115	2000
	50	69	1.9	31547	1100	21	0.7	3223	130	100	2.9	191836	7600	73	2.4	58171	2100
160	14	1	0.0	166	3.6	19	0.1	84	1.5	1	0.0	148	3.0	1	0.0	171	3.4
	16	30	0.2	154	4.3	19	0.1	65	1.6	1	0.0	329	8.7	1	0.0	383	10
	18	58	0.4	485	11	34	0.3	174	4.0	1	0.0	948	24	1	0.0	1056	27
	20	37	0.3	2041	34	37	0.4	707	11	1	0.0	4228	110	1	0.0	4730	120
	22	48	0.4	9534	120	32	0.3	2954	40	1	0.0	20438	510	1	0.0	23675	540
	24	65	0.7	30502	360	41	0.4	7787	130	1	0.0	71514	1800	1	0.0	83001	1900
320	14	1	0.0	238	5.5	1	0.0	278	8.7	1	0.0	166	3.7	1	0.0	191	4.5
	16	1	0.0	505	17	1	0.0	423	17	1	0.0	387	11	1	0.0	452	13
	18	1	0.0	1447	48	1	0.0	1208	57	1	0.0	1100	33	1	0.0	1268	40
	20	1	0.0	4792	160	1	0.0	4219	200	1	0.0	3992	130	1	0.0	4476	150
	22	1	0.0	20999	660	1	0.0	20417	860	1	0.0	15983	520	1	0.0	18663	600
	24	1	0.0	102158	3200	1	0.0	90398	3700	1	0.0	75546	2700	1	0.0	88602	2800

replenishment cycle. Therefore the solution of the relaxed problem solved by means of dynamic programming is usually feasible with respect to the original problem.

When $a \in \{40, 80\}$ Dijkstra is often unable to prove optimality at the root node, since the solution of the relaxed problem can easily violate inventory conservation constraints in the original problem under low ordering costs. This is due to the fact that the order-up-to-level for a replenishment cycle may easily be lower than the buffer stock levels held at the end of the former cycle. The main contribution brought by our relaxation in this situation consists in computing lower bounds during the search. Therefore in this case the domain reduction achieved with the other two filtering methods developed is critical in reducing the number of feasible values in the domain of expected closing-inventory-level decision variables. As shown in the experiments our approach can easily solve instances with up to 50 periods, both in terms of explored nodes and run time, for every combination of parameters we considered. In contrast, for the CP model both the run times and the number of explored nodes grow exponentially with the number of periods, and the problem becomes intractable for instances of significant size. In all cases our method explores fewer nodes than the pure CP approach, ranging from an improvement of one to several orders of magnitude. Apart from a few trivial instances on which both methods take a fraction of a second, this improvement is reflected in the run times.

6 Conclusions

It was previously shown [23] that CP is more natural than mathematical programming for expressing constraints for lot-sizing under the (R^n, S^n) policy, and leads to more efficient solution methods. This paper further improves the efficiency of the CP-based approach by exploiting three forms of cost-based filtering. The wide test bed considered shows the effectiveness of our approach under many different parameter configurations and demand trends. The improvement reaches several orders of magnitude in almost every instance we analyzed. We are now able to solve to optimality problems of a realistic size, in times of less than a second and often without search, since the bounds produced by our DP relaxation proved to be very tight in a large amount of instances. In future work we aim to extend our model to new features such as lead-time for orders and capacity constraints for the inventory.

7 Appendix

7.1 Considering a Unit Production Cost p

The stochastic programming formulation given can be extended to consider a unit production cost p as follows

$$\min \; E\{TC\} = \int_{d_1}\int_{d_2}\dots\int_{d_N}\sum_{t=1}^{N}(a\delta_t + h\cdot\max(I_t,0) + p\cdot Q_t) \\ g_1(d_1)g_2(d_2)\dots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\dots\mathrm{d}(d_N) \tag{46}$$

subject to Constraint (2), (3), (4) and (5), for $t = 1, \ldots, N$. The given objective function (46) can be rewritten as

$$E\{TC\} = p \cdot K + \min \int_{d_1} \int_{d_2} \ldots \int_{d_N} p \cdot I_N + \sum_{t=1}^{N} (a\delta_t + h \cdot \max(I_t, 0)) \\ g_1(d_1)g_2(d_2)\ldots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\ldots\mathrm{d}(d_N) \tag{47}$$

where $K = \int_{d_1} \int_{d_2} \ldots \int_{d_N} \sum_{t=1}^{N} d_t \; g_1(d_1)g_2(d_2)\ldots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\ldots\mathrm{d}(d_N)$. For further details on this transformation the reader may refer to [13, 21], where a similar transformation is described in details for the stochastic inventory control problem under a penalty cost scheme. Intuitively, objective function (47) shows that the effect of the unit production cost p can be decomposed in a constant factor $p \cdot K$ and in a variable factor $p \cdot I_N$ that depends on the very last closing-inventory-level planned. The deterministic equivalent CP approach is

$$E\{TC\} = p \sum_{t=1}^{N} \tilde{d}_t + \min \left[p \cdot \tilde{I}_N + \sum_{t=1}^{N} \left(a\delta_t + h\tilde{I}_t \right) \right] \tag{48}$$

subject to Constraint (7), (8), (9) and (10), for $t = 1 \ldots N$. It directly follows that the variable effect of the unit production cost p is reflected only on the cost of the very last replenishment cycle scheduled. The cost-based filtering method presented in Section 3.2 is independent of the considerations presented here. It remains sound under a unit production cost if the associated pre-processing method can consider this cost. The pre-processing methods in [23] and the cost-based filtering method in Section 3.1 can be extended to consider a unit production cost p by replacing the definition given in (15) for the cost $c(i, j)$ of a replenishment cycle $T(i, j)$ as follows:

$$\widehat{c}(i, j) = \begin{cases} c(i, j) & \text{if} \quad j \neq N \\ p \cdot b(i, j) + c(i, j) & \text{if} \quad j = N. \end{cases} \tag{49}$$

The cost-based filtering in Section 4 in a similar manner applies to the case where a unit production cost p is considered if, when the connection matrix for the graph constructed is built, $c(i, j)$ is replaced by $\widehat{c}(i, j)$ as just described.

7.2 Proof: Replenishment Cycle Length Bound

By using the definition of $c(i, j)$ we can rewrite (17) as

$$a + h(k - i + 1)b(i, k) + h \sum_{t=1}^{k} (t - i)\tilde{d}_t + a + h(j - k + 1)b(k + 1, j + 1) \\ + h \sum_{t=k+1}^{j+1} (t - k - 1)\tilde{d}_t \leq a + h(j - i + 2)b(i, j + 1) + h \sum_{t=i}^{j+1} (t - i)\tilde{d}_t \tag{50}$$

which can be simplified to

$$\frac{a}{h} - \sum_{t=k+1}^{j+1} (k+1-i)\tilde{d}_t \leq (j-k+1)\left[b(i,\, j+1) - b(k+1,\, j+1)\right] + (k-i+1)\left[b(i,\, j+1) - b(i,\, k)\right]. \tag{51}$$

We now want to prove that if $p > j+1$, then $\exists k+1 \in \{i+1,\, j\}$ s.t. $c(i,\, k) + c(k+1,\, p) \leq c(i,\, p) \wedge b(i,\, k) \leq R(k+1,\, p)$. We can rewrite this condition as we did before and therefore obtain an expression similar to (51), that is

$$\frac{a}{h} - \sum_{t=k+1}^{p} (k+1-i)\tilde{d}_t \leq (p-k)\left[b(i,\, p) - b(k+1,\, p)\right] + (k-i+1)\left[b(i,\, p) - b(i,\, k)\right]. \tag{52}$$

We now subtract both the left and the right term of (51) from (52). Thus we get

$$\begin{aligned} &- \sum_{t=j+2}^{p} (k+1-i)\tilde{d}_t + (j-k+1)\left[b(i,\, j+1) - b\,(k+1,\, j+1)\right] \\ &\quad + (k-i+1)\left[b(i,\, j+1) - b(i,\, k)\right] \leq (p-j-1)\left[b(i,\, p) - b(k+1,\, p)\right] \\ &\quad + (j-k+1)\left[b(i,\, p) - b(k+1,\, p)\right] + (k-i+1)\left[b(i,\, p) - b(i,\, k)\right], \end{aligned} \tag{53}$$

by omitting the term $-\sum_{t=j+2}^{p}(k+1-i)\tilde{d}_t$ to save space and rearranging the other terms we obtain

$$\begin{aligned} &(j-k+1)\left[b(k+1,\, p) - b(k+1,\, j+1)\right] \\ &\quad \leq (j-i+2)\left[b(i,\, p) - b(i,\, j+1)\right] + (p-j-1)\left[b(i,\, p) - b(k+1,\, p)\right], \end{aligned} \tag{54}$$

we change name to the coefficients

$$\begin{aligned} &A \cdot b(k+1,\, p) - A \cdot b(k+1,\, j+1) \\ &\quad \leq B \cdot b(i,\, p) - B \cdot b(i,\, j+1) + C \cdot b(i,\, p) - C \cdot b(k+1,\, p) \end{aligned} \tag{55}$$

and finally

$$(A+C) \cdot b(k+1,\, p) - A \cdot b(k+1,\, j+1) \leq (B+C) \cdot b(i,\, p) - B \cdot b(i,\, j+1), \tag{56}$$

where $A + C = p - k$ and $B + C = p - i + 1$. Reinserting the omitted term we obtain (18). Since $b(i,\, k) \leq R(k+1,\, j+1)$, it also follows that $b(i,\, k) \leq R(k+1,\, p)$.

Therefore, under the given conditions, it is never optimal to cover the span $\{i, \ldots, p\}$, $p > j$ by using a single replenishment cycle $T(i, p)$. Hence the optimum period $k+1$ for the next replenishment after the one scheduled in period i lies in the span $\{i+1, \ldots, j+1\}$ and it cannot be after $j+1$.

7.3 Modified Dijkstra's Shortest Path Algorithm

We will use a modified implementation of Dijkstra's Shortest Path Algorithm in order to enhance performances and make our relaxation compatible with Method I in [23]. Dijkstra's strategy relies on the following well known Shortest Path Theorem, which holds for any directed acyclic graph

Theorem 3 (Shortest Path Theorem) *If P is the shortest path from node u to node v and if P passes through node z, then P is made up by the shortest path Q_1 from u to z and by the shortest path Q_2 from z to v.*

Since we are solving a problem that implies a one-way temporal feasibility, as Wagner and Whitin notice in [25], half of our connection matrix will be set to ∞. Therefore any instance of size N can be solved in $N(N+1)/2$ steps taking this fact into account during the computation as we will see.

Let G be a directed acyclic graph $\langle V, A\rangle$, where V is a set of N numbered vertices $\{v_1, ..., v_N\}$ and A is a set of arcs among these nodes. Let W be a square matrix representing the cost related to each arc that appears in A. Let v_1 be the source we are computing shortest paths from. Let $d[v_i]$ be a label for any vertex $v_i \in V$, and $a[v_i]$ the index of the ancestor of node $v_i \in V$ in the shortest path. At the end of the computation $d[v_i]$ represents the shortest distance from the source v_1 to the vertex v_i. It is also possible to find every vertex in the shortest path from v_i to v_1 following in a recursive fashion the chain of indexes that starts with $a[v_i]$. In particular we will be interested in the shortest path from v_N to v_1, which is the one that covers our planning horizon. The complete code is shown in Algorithm l. In order to reduce steps to $N(N+1)/2$ we introduced $j > i$ as a precondition for the execution of Procedure $Relax(v_i,v_j,W)$. Notice also that in order to make the algorithm compatible with filtering methods in [23] some checks on vertex indexes have been introduced. In particular in Procedure $Relax(v_i,v_j,W)$ when two or more paths exist with the same distance from v_1 we always choose the ancestor v_i that has the highest index i. The reason we do this is related to the way pre-processing Method I in [23] filters values in decision variables domain. In fact, when a replenishment period i, $i \in \{1, ..., N\}$ is considered, such a method looks for the lowest j s.t. $j \geq i$ after which it is not longer optimal to schedule the next replenishment. This means that, if more policies that share the same expected cost exist, only the one that has shorter, and obviously more, replenishment cycles will be preserved by Method I, while values that are feasible with respect to other policies equally costly may be pruned. So we introduced the described checks on vertex indexes in order to make sure that, when more optimal policies exist, our modified algorithm will always find the one that has the highest

possible number of replenishment cycles (i.e. the shortest path with the highest possible number of arcs).

Algorithm 1: Modified Shortest Path Algorithm

```
input  : G, W, v_1
output: d, a
begin
    Initialize(G, v_1)
    Let d[v_i] be the shortest path from v_i to v_1
    Insert all vertices in G in a priority queue Q
    while Q is not empty do
        extract v_i s.t. d[v_i] is minimum
        for each vertex v_j adjacent to v_i s.t. j > i do
            Relax(v_i, v_j, W)
end
```

Procedure `Initialize(`G,v_1`)`

```
begin
    for each vertex v_i in G do
        set d[v_i] to W(v_1, v_i)
        set a[v_i] to 1
    set d[v_1] to 0
end
```

Procedure `Relax(`v_i,v_j,W`)`

```
begin
    if d[v_j] > d[v_i] + W(v_i, v_j) then
        set d[v_j] equal to d[v_i] + W(v_i, v_j)
        set a[v_j] equal to i
    else
        if d[v_j] == d[v_i] + W(v_i, v_j) AND i > a[v_j] then
            set a[v_j] equal to i
end
```

Acknowledgements This work was supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR) and Grant No. 00/PI.1/C075.

References

1. Apt, K. (2003). *Principles of constraint programming*. New York, NY, USA: Cambridge University Press.
2. Askin, R. G. (1981). A procedure for production lot sizing with probabilistic dynamic demand. *AIIE Transactions, 13*, 132–137.
3. Birge, J. R., & Louveaux, F. (1997). *Introduction to stochastic programming*. New York: Springer.
4. Bookbinder, J. H., & Tan, J. Y. (1988). Strategies for the probabilistic lot-sizing problem with service-level constraints. *Management Science, 34*, 1096–1108.
5. Brailsford, S. C., Potts, C. N., & Smith, B. M. (1999). Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research, 119,* 557–581.
6. Charnes, A., & Cooper, W. W. (1959). Chance-constrainted programming. *Management Science, 6*(1), 73–79.
7. Fahle T., & Sellmann, M. (2002). Cost-based filtering for the constrained knapsack problem. *Annals of Operations Research, 115*, 73–93.
8. Florian, M., Lenstra, J. K., & Rinooy Kan, A. H. G. (1980). Deterministic production planning: Algorithms and complexity. *Management Science, 26*(7), 669–679.
9. Focacci, F., Lodi, A., & Milano, M. (1999). Cost-based domain filtering. In *Proceedings of the 5th international conference on the principles and practice of constraint programming* (pp. 189–203). Springer, 1999. Lecture notes in computer science no. 1713.
10. Focacci, F., & Milano, M. (2001). Connections and integrations of dynamic programming and constraint programming. In *Proceedings of the international workshop on integration of AI and OR techniques in constraint programming for combinatorial optimization problems CP-AI-OR 2001*.
11. Garey, M. R., & Johnson, D. S. (1990). *Computers and intractability; A guide to the theory of NP-completeness*. New York, NY, USA: Freeman
12. Laburthe, F., & the OCRE project team. (1994). Choco: Implementing a cp kernel. Technical report, Bouygues e-Lab, France.
13. Rossi, R., Tarim, S. A., Hnich, B., & Prestwich, S. (2007). Replenishment planning for stochastic inventory systems with shortage cost. In *Proceedings of the international conference on integration of AI and OR techniques in constraint programming for combinatorial optimization problems CP-AI-OR 2007* (pp. 229–243). Springer. Lecture Notes in Computer Science No. 4510.
14. Rossi, R., Tarim, S. A., Hnich, B., & Prestwich, S. (2008). A global chance-constraint for stochastic inventory systems under service level constraints. *Constraints* (to appear)
15. Sedgewick, R. (1988). *Algorithms* (2nd edn.). Boston, MA, USA: Addison-Wesley
16. Silver, E. A. (1978). Inventory control under a probabilistic time-varying demand pattern. *AIIE Transactions, 10*, 371–379.
17. Silver, E. A., Pyke, D. F., & Peterson, R. (1998). *Inventory management and production planning and scheduling*. New York: Wiley.
18. Tarim, S. A. (1996). *Dynamic lotsizing models for stochastic demand in single and multi-echelon inventory systems*. Ph.D. thesis, Lancaster University.
19. Tarim, S. A., Hnich, B., Rossi, R., & Prestwich, S. (2007). Cost-based filtering for stochastic inventory control. In *Recent advances in constraints joint ERCIM/CoLogNET international workshop on constraint solving and constraint logic programming, CSCLP 2006*, (pp. 169–183). Springer. Lecture notes in artificial intelligence no. 4651.
20. Tarim, S. A., & Kingsman, B. G. (2004). The stochastic dynamic production/inventory lot-sizing problem with service-level constraints. *International Journal of Production Economics, 88*, 105–119.
21. Tarim, S. A., & Kingsman, B. G. (2006). Modelling and computing (R^n,S^n) policies for inventory systems with Non-stationary stochastic demand. *European Journal of Operational Research, 174*, 581–599.
22. Tarim, S. A., Manandhar, S., & Walsh, T. (2006). Stochastic constraint programming: A scenario-based approach. *Constraints, 11*(1), 53–80.
23. Tarim, S. A., & Smith, B. (2008). Constraint programming for computing non-stationary (R,S) inventory policies. *European Journal of Operational Research* (to appear)
24. Van Hentenryck, P., & Carillon, J. P. (1988). Generality vs. specificity: An experience with ai and or techniques. In *Proceedings of the national conference on artificial intelligence* (*AAAI-88*).
25. Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management Science, 5*, 89–96.