Cost-based filtering for stochastic inventory control

Cost-Based Filtering for Stochastic Inventory Control

S. Armagan Tarim[1], Brahim Hnich[2], Roberto Rossi[3], and Steven Prestwich[3]

Department of Management, Hacettepe University, Turkey[1]
armagan.tarim@hacettepe.edu.tr
Faculty of Computer Science, Izmir University of Economics, Turkey[2]
brahim.hnich@ieu.edu.tr
Cork Constraint Computation Centre, University College, Cork, Ireland[3]
{r.rossi,s.prestwich}@4c.ucc.ie

Abstract. An interesting class of production/inventory control problems considers a single product and a single stocking location, given a stochastic demand with a known non-stationary probability distribution. 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 a cost-based filtering method. Our algorithm can solve to optimality instances of realistic size much more efficiently than previous approaches, often with no search effort at all.

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 Material Requirement Planning [30].

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. We assume that it is not possible to sell back excess items to the vendor at the end of a period. As a service level constraint we require the probability to be at least a given value α that at the end of every period the net inventory will not be negative. 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.

Different inventory control policies can be adopted to cope with 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 [29]. 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 or-

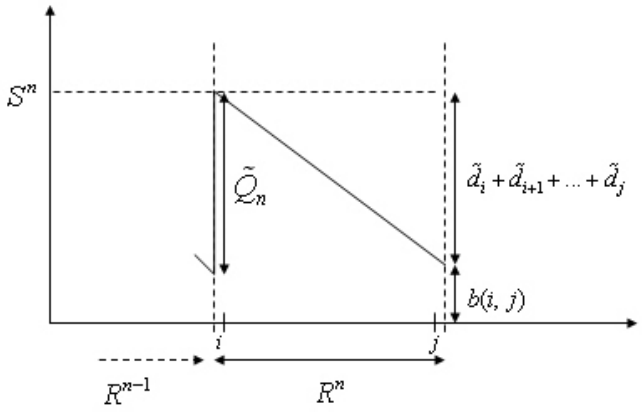

Fig. 1. (R^n,S^n) policy. R^n denotes the set of periods covered by the nth replenishment cycle; S^n is the order-up-to-level for this cycle; $\tilde{Q}_n$ is the expected order quantity; $\tilde{d}_i + \tilde{d}_{i+1} + \ldots + \tilde{d}_j$ is the expected demand; $b(i, j)$ is the buffer stock required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

der 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 $n = \{1, \ldots, N\}$.

Early works in this area adopted heuristic strategies such as those proposed by Silver [20], Askin [2] and Bookbinder & Tan [5]. The first complete solution method for this problem was introduced by Tarim & Kingsman [23], who proposed a certainty-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 & Smith [24] introduced a more compact and efficient Constraint Programming (CP) formulation of the same problem that showed a significant computational improvement over the MIP formulation.

This paper extends Tarim & Smith's work, retaining their model but augmenting it with a *cost-based filtering* method to enhance domain pruning. Cost-based filtering is an elegant way of combining techniques from CP and Operations Research (OR) [7, 8]: 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 has not yet 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.

The paper is organized as follows. Section 2 describes the CP model introduced by Tarim & Smith. Section 3 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 4 evaluates our methods. Section 5 draws conclusions and discusses future extensions.

2 A CP model

In this section we review the CP formulation proposed by Tarim & Smith [24]. First we provide some formal background related to constraint programming. Recall that a Constraint Satisfaction Problem (CSP) [1, 6] 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 stochastic programming (for a detailed discussion on stochastic programming see [33]) formulation for the (R^n, S^n) policy proposed in [5] is

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

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

$$I_t + d_t - I_{t-1} \geq 0 \tag{2}$$
$$I_t + d_t - I_{t-1} > 0 \Rightarrow \delta_t = 1 \tag{3}$$
$$Pr\{I_t \geq 0\} \geq \alpha \tag{4}$$
$$I_t \in \mathbf{Z}, \quad \delta_t \in \{0, 1\} \tag{5}$$

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 ($\delta_t = 0$). The objective function (1) minimizes the expected total cost over the given planning horizon.

The respective CP formulation proposed in [24] is

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

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

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} \geq 0 \tag{7}$$
$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} > 0 \Rightarrow \delta_t = 1 \tag{8}$$
$$Y_t \geq j \cdot \delta_j \quad j = 1, \dots, t \tag{9}$$
$$element\,(Y_t, b(\cdot, t), H_t) \tag{10}$$
$$\tilde{I}_t \geq H_t \tag{11}$$
$$\tilde{I}_t, H_t \in \mathbf{Z}^+ \cup \{0\}, \quad \delta_t \in \{0, 1\}, \quad Y_t \in \{1, \dots, N\} \tag{12}$$

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

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

The $element(X, list[], Y)$ constraint [31] enforces a relation such that variable Y represents the value of element at position X in the given list. $G_{d_i + d_{i+1} + \dots + d_j}$ is the cumulative probability distribution function of $d_i + d_{i+1} + \dots + d_j$. 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 net inventory at period t must be no less than the expected net inventory in period $t+1$ plus the expected

demand in period t. Constraint (8) expresses the replenishment condition. We have a replenishment if the expected net inventory at period t is greater than the expected net inventory in period $t+1$ plus the expected demand in period t. This means that we received some extra goods as a consequence of an order. Constraints (9,10,11) enforce the required service level α. This is done by specifying the minimum buffer stock required for each period t in order to ensure 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 [23]. 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 [32]. More specifically the authors developed a certainty-equivalent constraint for each chance constraint that enforces the required service level at the end of each replenishment cycle.

2.1 Computational Complexity

The chance-constrained problem presented in [5] for the (R^n, S^n) policy under stochastic demand is PSPACE-complete as shown in [25]. We 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 the certainty-equivalent model we presented, where buffer stocks for each possible replenishment cycle are computed independently. In [4] Florian et. al. gave an overview for the complexity of deterministic production planning. In particular they established NP-hardness for this 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 specific cases. Among these they cited Wagner and Whitin's [27] work, where the infinite capacity deterministic production planning problem is solved in polynomial time. Wagner and Whitin's algorithm relies upon their *Planning Horizon Theorem*, which exploits the fact that the feasible region is a closed bounded convex set and that the cost function is concave [4], thus the minimum value for such an objective function is achieved at one of the extreme points of this set. The special structure of the set allows a simple characterization of the production plans corresponding to its extreme points. The core insight proposed by Wagner and Whitin is the fact that in the search for the optimal policy it is sufficient to consider programs in which at period t one does not both place an order and bring in inventory. Their *Planning Horizon Theorem* states that if it is optimal to incur a setup cost in period t, when periods $1,\ldots,t$ are considered in isolation, then we may retain this decision for the N period model without losing optimality.

Therefore it is possible to adopt an optimal program for period $1, \ldots, t-1$ considered separately. It is easy to see that the certainty-equivalent model described in the former section is an over-constrained version of the infinite capacity deterministic production planning problem. The additional constraints in the model we presented enforce buffer stocks for each replenishment cycle. Since we have buffer stocks, the last period of a replenishment cycle usually requires a positive inventory in our certainty-equivalent model, so it is possible that an order is placed even if the inventory level is not null. Therefore the simple characterization of optimal programs proposed by Wagner and Whitin cannot be applied, since buffer stock carried from former periods may affect the cost of subsequent programs.

We shall now show, by using a counter-example, that Wagner and Whitin's algorithm cannot be applied to the over-constrained problem, for which therefore no polynomial algorithm is known. Let us consider a 3-period planning horizon. The demand is normally distributed in each period with coefficient of variation 0.3. The mean values of the demand are respectively 240, 60, 200 for periods 1, 2, 3. The required service level is 95%, the ordering cost is 130 and the holding cost is 1. The required buffer stock levels for the possible replenishment cycles are $b(1,1) = 118, b(2,2) = 30, b(3,3) = 99, b(1,2) = 122, b(2,3) = 103, b(1,3) = 157$. The optimal policy can be easily obtained by solving our certainty-equivalent model (Fig. 2 - c). Such a policy fixes orders in periods 1 and 3 and its cost is 663. Following the same reasoning in [27] (Table 1) the optimal plan for period

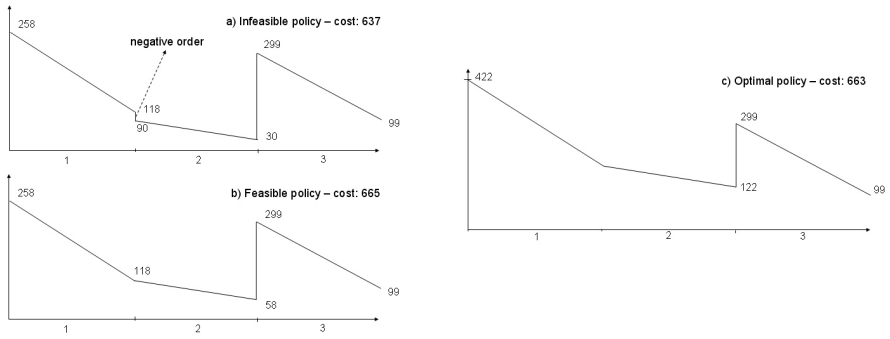

Fig. 2. a) The infeasible policy and its cost obtained by means of Wagner and Whitin algorithm. b) The respective feasible policy and its cost. c) The optimal policy and its cost obtained by using our certainty-equivalent model.

1 is to order (entailing an ordering cost of 130 and a holding cost of 118). Two possibilities must be evaluated for period 2; order in period 2, and use the best policy for period 1 considered alone (at a cost of $160 + 248 = 408$); or order in period 1 for both periods, and carry inventory into period 2 (at a cost of $130 + 122 \cdot 2 + 60 = 434$). The better policy appears to be the first one, but it is actually the second one. In period 3, if the algorithm in [27] worked there would

Period	1	2	3
	248	408	637
		434	784
			1061
Minimum cost	248	408	637
Optimal policy	$\underline{1}$	$\underline{2}$	$\underline{3}$

Table 1. Wagner and Whitin algorithm steps. In the optimal policy row only the last period is shown; $\underline{3}$ indicates that the optimal policy for periods 1 through 3 is to order in period 3 to satisfy $\tilde{d}_3$ and adopt an optimal policy for periods 1 through 2 considered separately.

be three alternatives: order in period 3, and use the best policy for period 1 and 2 considered alone (at a cost of $229 + 408 = 637$); or order in period 2 for the latter two periods and use the best policy for period 1 considered alone (at a cost of $536 + 248 = 784$); or order in period 1 for the entire three periods (at a cost of 1061). The policy obtained by Wagner and Whitin's algorithm is therefore the best among these three, which places an order in period 1, 2 and in period 3 at a cost of 637. Unfortunately this policy is infeasible because it requires a negative order quantity in period 2 (Fig. 2 - a). The respective feasible policy that places orders in the same periods has a higher cost of $130 \cdot 3 + 118 + 58 + 99 = 665$ (Fig. 2 - b). This counter-example shows that Wagner and Whitin's algorithm is not suitable for our deterministic equivalent problem.

2.2 Domain pre-processing

In [24] 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 nicer 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 until 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 & 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 course of action: 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 Cost-based filtering by relaxation

The CP model as described so far suffers from a lack of tight bounds on the objective function. We now propose a relaxation for our model to compute a valid lower bound at each node of the search tree. We first show that the CP model can be reduced to a Shortest Path Problem if we relax constraints (7,8) for replenishment periods. 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, or the ability to sell stock back to the supplier. In this way we obtain a set $\mathcal{S}$ of $N(N+1)/2$ possible different replenishment cycles. 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. We will 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 has a specific 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 one. Node $N+1$ represents the end of the planning horizon. Recall that $b(i,j)$ denotes the minimum buffer stock level required to satisfy a given service level constraint during the replenishment cycle $T(i,j)$. 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 $Q(i,j)$, where

$$Q(i,j) = a + h \sum_{k=i+1}^{j-1} (k-i)d_k + h(j-i)b(i,j-1) \tag{13}$$

The cost 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 charged at the end of each time period within the replenishment cycle and proportional to the amount of stocks held in inventory. Since we are dealing with a one-way temporal feasibility problem [27], 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 2.

	1	2	...	j	...	$N+1$
1	–	$Q(1,2)$	...	$Q(1,j)$	...	$Q(1,N+1)$
$\vdots$	–	–	$\ddots$	$\vdots$	$\ddots$	$\vdots$
i	–	–	–	$Q(i,j)$	...	$Q(i,N+1)$
$\vdots$	–	–	–	–	$\ddots$	$\vdots$
N	–	–	–	–	–	$Q(N,N+1)$

Table 2. Shortest Path Problem Connection matrix

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. Furthermore 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 $T(i, j-1)$. The feasibility of such a solution with respect to the original problem can be checked by verifying that it satisfies every relaxed constraint. To find a shortest path, and hence a valid lower bound, we use an improved Dijkstra 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 the Dijkstra algorithm can be found in [19]. Usually Dijkstra's algorithm does not apply any specific rule for labeling when ties are encountered in sub-path lengths. This is incorrect if we pre-process decision variable domains as described in [24]. In fact pre-processing Method I in [24] relies upon an upper bound for optimal replenishment cycle length. When a replenishment period i, $i \in \{1, ..., N\}$ is considered, it looks for the lowest j s.t. $j \geq i$ after which it is no longer optimal to schedule the next replenishment. This means that, if other policies exist that share the same expected cost, 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). As there is a complete ordering among nodes, we can easily implement this rule when labeling by always choosing as ancestor the node that minimizes the distance from the source and that has the highest index.

We now see how to use this relaxation during the search process when a partial solution is provided. 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 network 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. 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 relaxing them and applying Dijkstra's algorithm. 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 cost of the overall solution obtained by merging the two subproblem solutions is again a valid lower bound for the original problem. Let $R(i, j)$ denote the required minimum opening inventory level in period i, $i \in \{1, \ldots, N\}$, to meet demand until period $j + 1$, where $R(i, j) = b(i, j) + \sum_{t=i}^{j} \tilde{d}_t$. We can characterize when such a bound is an exact one: when the solutions of the two subproblems are both feasible with respect to the original model and the condition

$$b(i, k-1) \leq R(k, j) \tag{14}$$

is satisfied, the solution obtained by merging those for the independent subproblems is both feasible and optimal for the original problem. We have shown how to act when each of the possible cases, $\delta_i = 1$ and $\delta_i = 0$ is encountered. It is now possible at any point of the search in the decision tree to apply this relaxation to compute valid lower bounds. It is also possible to extend this cost-based filtering by considering not only the δ_t variable assignments, but also the $\tilde{I}_t$ variable assignments. In fact, when we compute the cost of a given replenishment cycle $T(i, j-1)$ (arc (i, j) in the matrix), we can also consider the current assignments for the closing inventory levels $\tilde{I}_t$ in the periods of this cycle. Since all the closing inventory levels of the periods within a replenishment cycle are linearly dependent, given an assignment for a decision variable $\tilde{I}_t$ we can easily compute all the other closing inventory levels in the cycle using $\tilde{I}_t - \tilde{d}_t - \tilde{I}_{t-1} = 0$, which is the inventory conservation constraint when no order is placed in period t. When the closing inventory levels in a replenishment cycle $T(i, j-1)$ are known it is easy to compute the overall cost associated with this cycle, which is by definition the sum of the ordering cost and of the holding cost components, $a + h \sum_{t=i}^{j-1} \tilde{I}_t$. We can therefore associate to arc (i, j) the highest cost that is produced by a current assignment for the closing inventory levels $\tilde{I}_t$, $t \in \{i, \ldots, j-1\}$, if no variable has been assigned yet, we simply use the minimum possible cost $Q(i, j)$, which we defined before.

4 Experimental results

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. 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, $\tilde{d}_t = 50[1 + \sin(\pi t/6)] + \min(t, 52 - t)$

		$\sigma_t/d_t = 1/3$								$\sigma_t/d_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$				$\alpha = 0.95$				$\alpha = 0.99$			
		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.	
a	N	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
40	40	28	0.4	106	2.9	86	1.2	249	6.4	40	0.6	574	17	10	0.1	192	6.4
	42	28	0.5	95	2.8	87	1.2	233	5.9	40	0.7	582	15	10	0.2	196	5.4
	44	29	0.6	133	4.9	88	1.3	266	8.3	41	0.8	884	26	11	0.2	285	9.0
	46	30	0.8	192	7.8	100	1.9	484	19	44	0.9	3495	120	11	0.2	813	31
	48	39	1.3	444	20	158	3.2	1024	42	66	2.0	5182	190	18	0.5	1208	48
	50	38	0.9	444	21	151	3.6	1024	45	55	1.8	4850	200	15	0.4	1208	52
80	40	52	0.8	1742	78	13	0.2	557	15	19	0.3	9316	300	16	0.3	11276	440
	42	49	0.9	1703	61	13	0.2	530	14	20	0.3	17973	530	17	0.3	22291	690
	44	51	1.0	4810	210	14	0.2	980	26	21	0.4	38751	1400	18	0.4	50805	1600
	46	52	1.1	6063	350	14	0.3	2122	79	31	0.7	103401	4300	18	0.4	111295	4100
	48	57	1.9	20670	1400	15	0.3	5284	210	29	0.7	237112	12000	19	0.5	321998	15000
	50	57	1.7	18938	1300	15	0.3	5284	230	23	0.6	251265	13000	19	0.5	358174	17000
160	14	1	0.0	141	3.0	56	0.2	156	2.5	1	0.0	112	2.6	1	0.0	116	2.4
	16	1	0.0	277	9.0	71	0.3	182	5.1	1	0.0	238	6.7	1	0.0	235	6.8
	18	1	0.0	673	19	50	0.3	393	11	1	0.0	799	24	1	0.0	603	16
	20	1	0.0	3008	82	61	0.5	1359	22	1	0.0	2887	86	1	0.0	2820	75
	22	1	0.0	10620	260	116	1.3	7280	71	1	0.0	14125	380	1	0.0	10739	280
	24	1	0.0	61100	1500	165	1.9	31615	320	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	12	1	0.0	361	13	1	0.0	246	8.7	1	0.0	284	9.3
	18	1	0.0	813	28	1	0.0	831	28	1	0.0	764	27	1	0.0	700	25
	20	1	0.0	2602	94	1	0.0	2415	82	1	0.0	2114	79	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 3. Test set P1

In each test we assume an initial null inventory level and a normally distributed demand for every period with a coefficient of variation σ_t/d_t for each $t \in \{1, \ldots, N\}$, where N is the length of the considered planning horizon. We performed tests using four different ordering cost values $a \in \{40, 80, 160, 320\}$ and two different $\sigma_t/d_t \in \{1/3, 1/6\}$. 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 $\alpha = 0.99$ ($z_{\alpha=0.99} = 2.326$). All experiments were performed on an Intel(R) Centrino(TM) CPU 1.50GHz with 500Mb RAM. The solver used for our test is Choco [15], an open-source solver developed in Java. The heuristic used for the selection of the variable is the usual min-domain / max-degree heuristic. The value selection heuristic chooses values in increasing order of size. 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 3, 4, 5 and 6 compare the performance of the state-of-the-art CP model with that of our new method.

When a=320, and often when a=160, the Dijkstra algorithm proves optimality at the root node. When $a \in \{40, 80\}$ Dijkstra is unable to prove optimality at the root node, so its main contribution consists in computing lower bounds

		$\sigma_t/d_t = 1/3$								$\sigma_t/d_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$				$\alpha = 0.95$				$\alpha = 0.99$			
		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.	
a	N	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
40	40	5	0.1	7	0.1	7	0.1	8	0.1	14	0.3	23	0.4	5	0.1	12	0.2
	42	5	0.1	7	0.1	7	0.1	8	0.1	14	0.2	23	0.4	5	0.1	10	0.1
	44	5	0.1	7	0.1	7	0.1	8	0.1	14	0.3	23	0.5	5	0.1	10	0.2
	46	5	0.1	7	0.1	7	0.1	8	0.2	14	0.3	23	0.5	5	0.1	10	0.2
	48	5	0.1	7	0.2	7	0.1	8	0.2	14	0.3	23	0.5	5	0.1	10	0.2
	50	5	0.1	7	0.2	7	0.2	8	0.2	14	0.3	23	0.6	5	0.1	10	0.2
80	40	24	0.4	4592	14	17	0.3	275	8.3	46	0.9	2565	63	44	0.9	1711	45
	42	24	0.4	4866	13	17	0.3	283	6.7	46	1.0	3027	68	44	0.7	2043	48
	44	24	0.4	5091	15	17	4.7	280	7.9	47	1.1	6024	160	45	0.9	4299	120
	46	46	0.9	5291	45	19	0.4	545	17	51	1.3	14058	410	49	1.1	10311	290
	48	37	0.8	5544	51	19	0.5	545	18	53	1.5	14058	440	53	1.3	10311	310
	50	34	0.7	5850	51	19	0.5	545	19	56	1.8	14079	470	54	1.4	10347	330
160	14	2	0.0	166	3.6	25	0.1	84	1.0	1	0.0	148	2.9	1	0.0	171	3.4
	16	25	0.1	154	4.3	25	0.1	65	1.2	1	0.0	329	8.6	1	0.0	383	11
	18	24	0.1	485	11	27	0.2	174	2.9	1	0.0	948	24	1	0.0	1056	28
	20	34	0.3	2041	35	50	0.4	707	7.9	1	0.0	4228	110	1	0.0	4730	120
	22	50	0.6	9534	120	35	0.3	2954	29	1	0.0	20438	500	1	0.0	23675	530
	24	52	0.5	30502	360	40	0.4	7787	88	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	14	1	0.0	387	12	1	0.0	452	14
	18	1	0.0	1447	49	1	0.0	1208	41	1	0.0	1100	34	1	0.0	1268	41
	20	1	0.0	4792	160	1	0.0	4219	150	1	0.0	3992	130	1	0.0	4476	150
	22	1	0.0	20999	670	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 4. Test set P2

		$\sigma_t/d_t = 1/3$								$\sigma_t/d_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$				$\alpha = 0.95$				$\alpha = 0.99$			
		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.	
a	N	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
40	40	3	0.0	5	0.0	3	0.0	4	0.0	7	0.1	9	0.2	3	0.0	5	0.0
	42	3	0.0	5	0.0	3	0.0	4	0.0	7	0.1	9	0.2	3	0.0	5	0.0
	44	4	0.0	7	0.1	4	0.0	6	0.1	8	0.1	14	0.3	4	0.0	7	0.1
	46	9	0.2	15	0.3	5	0.1	13	0.3	17	0.3	40	1.1	8	0.2	14	0.3
	48	8	0.2	15	0.3	5	0.1	13	0.3	17	0.4	56	1.8	14	0.3	25	0.6
	50	7	0.2	15	0.3	5	0.1	13	0.3	17	0.4	56	1.9	13	0.3	25	0.5
80	40	24	0.5	349	10	10	0.1	55	1.2	24	0.4	722	20	10	0.1	310	8.7
	42	26	0.5	354	8.6	8	0.1	53	1.2	23	0.5	1436	35	12	0.2	315	7.5
	44	27	0.5	571	17	9	0.1	88	2.4	24	0.5	3461	110	13	0.2	1053	32
	46	42	1.0	2787	90	10	0.2	258	8.1	37	1.3	10612	360	21	0.5	2881	94
	48	41	1.1	6803	240	10	0.2	385	13	33	1.3	28334	1100	21	0.6	7790	280
	50	42	1.2	6575	250	10	0.2	385	14	35	1.5	26280	1100	21	0.7	7371	280
160	14	7	0.0	23	0.2	9	0.0	16	0.1	14	0.1	53	0.6	10	0.1	29	0.3
	16	5	0.0	19	0.2	9	0.0	18	0.2	19	0.1	52	0.8	9	0.1	26	0.4
	18	7	0.0	42	0.5	10	0.1	30	0.3	21	0.1	149	2.2	11	0.1	87	1.2
	20	17	0.2	137	1.3	12	0.1	70	0.7	23	0.2	512	6.1	20	0.3	310	3.5
	22	9	0.1	376	4.0	15	0.1	221	2.3	28	0.3	1848	18	15	0.2	938	9.4
	24	10	0.2	995	12	25	0.3	543	6.3	37	0.7	4784	55	19	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	11	1	0.0	518	11	1	0.0	707	14	1	0.0	465	8.5
	18	1	0.0	1475	35	1	0.0	1170	27	1	0.0	1995	44	1	0.0	1416	33
	20	1	0.0	5342	140	1	0.0	4059	96	1	0.0	6678	170	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 5. Test set P3

		$\sigma_t/d_t = 1/3$								$\sigma_t/d_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$				$\alpha = 0.95$				$\alpha = 0.99$			
		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.	
a	N	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
40	40	7	0.1	21	0.3	11	0.1	24	0.5	30	0.6	89	1.8	7	0.1	33	0.5
	42	7	0.1	18	0.3	11	0.2	21	0.4	30	0.9	91	2.0	7	0.1	31	0.5
	44	8	0.1	32	0.7	12	0.2	37	0.9	31	0.7	152	3.6	8	0.1	51	1.0
	46	14	0.5	83	2.0	14	0.3	93	2.4	46	1.4	474	12.4	13	0.3	126	2.8
	48	12	0.2	83	2.2	14	0.3	93	2.6	56	2.3	735	20.9	19	0.4	188	4.5
	50	11	0.2	83	2.3	14	0.3	93	2.8	58	2.5	735	22.0	18	0.4	188	4.9
80	40	46	0.7	1372	39	24	0.4	433	13	53	1.1	5098	130	36	0.7	2133	55
	42	51	1.5	1673	39	20	0.4	438	11	50	1.1	11452	270	41	1.1	2513	59
	44	52	1.0	2907	74	21	0.4	716	23	52	1.3	27184	780	43	1.3	8776	240
	46	78	2.2	13306	380	23	0.5	2178	74	76	2.4	77332	2600	62	2.3	22582	690
	48	75	1.8	32709	1000	23	0.6	3223	120	76	3.1	202963	7500	61	2.3	60115	2000
	50	77	1.9	31547	1100	23	0.6	3223	130	81	3.2	191836	7600	63	3.3	58171	2100
160	14	11	0.0	166	3.6	25	0.1	84	1.5	1	0.0	148	3.0	1	0.0	171	3.4
	16	9	0.0	154	4.3	25	0.1	65	1.6	1	0.0	329	8.7	1	0.0	383	11
	18	10	0.1	485	11	27	0.1	174	4.0	1	0.0	948	25	1	0.0	1056	28
	20	19	0.2	2041	35	50	0.4	707	12	1	0.0	4228	110	1	0.0	4730	120
	22	17	0.1	9534	120	35	0.3	2954	41	1	0.0	20438	510	1	0.0	23675	540
	24	27	0.4	30502	360	40	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	18	1	0.0	387	12	1	0.0	452	14
	18	1	0.0	1447	48	1	0.0	1208	56	1	0.0	1100	34	1	0.0	1268	41
	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

Table 6. Test set P4

during the search. However, our method easily solves instances with up to 50 periods, both in term 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.

5 Conclusions

It was previously shown [24] 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 cost-based filtering. The wide test-bed considered shows the effectiveness of our approach under different parameter configurations and demand trends. The improvement is several orders of magnitude in almost every instance we analyzed. We are now able to solve to optimality problems of a realistic size with planning horizons of fifty and more periods, 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 many 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.

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. K. Apt. Principles of Constraint Programming. Cambridge University Press, Cambridge, UK, 2003.
2. R. G. Askin. A Procedure for Production Lot Sizing With Probabilistic Dynamic Demand. *AIIE Transactions* 13:132–137, 1981.
3. R. E. Bellman Dynamic Programming, Princeton University Press, Princeton, NJ. 1957
4. M. Florian, J. K. Lenstra, A. H. G. Rinooy Kan. Deterministic Production Planning: Algorithms and Complexity. *Management Science* 26(7):669–679, 1980.
5. J. H. Bookbinder, J. Y. Tan. Strategies for the Probabilistic Lot-Sizing Problem With Service-Level Constraints. *Management Science* 34:1096–1108, 1988.
6. S. C. Brailsford, C. N. Potts, B. M. Smith. Constraint Satisfaction Problems: Algorithms and Applications. *European Journal of Operational Research* 119:557–581, 1999.
7. T. Fahle, M. Sellmann. Cost-Based Filtering for the Constrained Knapsack Problem. *Annals of Operations Research* 115:73–93, 2002.
8. F. Focacci, A. Lodi, M. Milano. Cost-Based Domain Filtering. *Fifth International Conference on the Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* 1713, Springer Verlag, 1999, pp. 189–203.
9. L. Fortuin. Five Popular Probability Density Functions: a Comparison in the Field of Stock-Control Models. *Journal of the Operational Research Society* 31(10):937–942, 1980.
10. S. J. Gartska, R. J.-B. Wets. On Decision Rules in Stochastic Programming. *Mathematical Programming* 7:117–143, 1974.
11. S. K. Gupta, J. K. Sengupta. Decision Rules in Production Planning Under Chance-Constrained Sales. *Decision Sciences* 8:521–533, 1977.
12. R. B. Heady, Z. Zhu. An Improved Implementation of the Wagner-Whitin Algorithm. *Production and Operations Management* 3(1), 1994.
13. L. A. Johnson, D. C. Montgomery. Operations Research in Production Planning, Scheduling, and Inventory Control. Wiley, New York, 1974.
14. N. Jussien, R. Debruyne, P. Boizumault. Maintaining Arc-Consistency Within Dynamic Backtracking. *Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* 1894, 2000, pp. 249–261.
15. F. Laburthe and the OCRE project team. Choco: Implementing a CP Kernel. Bouygues e-Lab, France.
16. I. J. Lustig, J.-F. Puget. Program Does Not Equal Program: Constraint Programming and its Relationship to Mathematical Programming. *Interfaces* 31:29–53, 2001.
17. R. Peterson, E. Silver, D. F. Pyke. Inventory Management and Production Planning and Scheduling. John Wiley and Sons, New York, 1998.

18. E. L. Porteus. Foundations of Stochastic Inventory Theory. Stanford University Press, Stanford, CA, 2002.
19. R. Sedgewick. Algorithms. Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.
20. E. A. Silver. Inventory Control Under a Probabilistic Time-Varying Demand Pattern. *AIIE Transactions* 10:371–379, 1978.
21. D. Simchi-Levi, E. Simchi-Levi, P. Kaminsky. Designing and Managing the Supply Chain. McGraw-Hill, 2000.
22. J. Y. Tan. Heuristic for the Deterministic and Probabilistic Lot-Sizing Problems. MSc. thesis, Department of Management Sciences, University of Waterloo, 1983.
23. S. A. Tarim, B. G. Kingsman. The Stochastic Dynamic Production/Inventory Lot-Sizing Problem With Service-Level Constraints. *International Journal of Production Economics* 88:105–119, 2004.
24. S. A. Tarim, B. Smith. Constraint Programming for Computing Non-Stationary (*R,S*) Inventory Policies. *European Journal of Operational Research*. to appear.
25. S. A. Tarim, S. Manandhar, T. Walsh. Stochastic Constraint Programming: A Scenario-Based Approach. *Constraints* 11:53–80, 2006.
26. S. Vajda. Probabilistic Programming. Academic Press, New York, 1972.
27. H. M. Wagner, T. M. Whitin. Dynamic Version of the Economic Lot Size Model. *Management Science* 5:89–96, 1958.
28. P. H. Zipkin. Foundations of Inventory Management. McGraw-Hill/Irwin, Boston, Mass., 2000.
29. E. A. Silver, D. F. Pyke, R. Peterson. Inventory Management and Production Planning and Scheduling, John-Wiley and Sons, New York, 1998.
30. U. Wemmerlov. The Ubiquitous EOQ - Its Relation to Discrete Lot Sizing Heuristics. *Internat. J. of Operations and Production Management*, 1:161–179, 1981.
31. P. Van Hentenryck and J.-P. Carillon. Generality vs. specificity: an experience with AI and OR techniques. In National Conference on Artificial Intelligence (AAAI-88), 1988.
32. A. Charnes, W. W. Cooper. Chance-Constrainted Programming. *Management Science* 6(1):73–79, 1959.
33. J. R. Birge, F. Louveaux. Introduction to Stochastic Programming. Springer Verlag, New York , 1997.