Constraints (2008) 13:490–517
DOI 10.1007/s10601-007-9038-4

A Global Chance-Constraint for Stochastic Inventory Systems Under Service Level Constraints

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

Published online: 16 February 2008

Abstract We consider a class of production/inventory control problems that has a single product and a single stocking location, for which a stochastic demand with a known non-stationary probability distribution is given. Under the widely-known replenishment cycle policy the problem of computing policy parameters under service level constraints has been modeled using various techniques. Tarim and Kingsman introduced a modeling strategy that constitutes the state-of-the-art approach for solving this problem. In this paper we identify two sources of approximation in Tarim and Kingsman's model and we propose an exact *stochastic constraint programming* approach. We build our approach on a novel concept, *global chance-constraints*, which we introduce in this paper. Solutions provided by our exact approach are employed to analyze the accuracy of the model developed by Tarim and Kingsman.

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.

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

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

S. Armagan 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

Keywords Global chance-constraints · Stochastic inventory control · Non-stationary (R,S) policy · Uncertainty

1 Introduction

The study of lot-sizing began with Wagner and Whitin [34], and there is now a sizeable literature in this area extending the basic model to consider capacity constraints, multiple items, multiple stages, etc. However, most previous work on lot-sizing has been directed towards the deterministic case. For a general overview over deterministic lot-sizing problems the reader may refer to [14].

The practical problem is that in general many, if not all, of the future demands have to be forecasted. Point forecasts are typically treated as deterministic demands. However, the existence of forecast errors radically affects the behavior of the lot-sizing procedures based on assuming the deterministic demand situation. Forecasting errors lead both to stock-outs occurring with unsatisfied demands and to larger inventories being carried than planned. The introduction of safety stocks in turn generates even larger inventories and also more orders. It is reported by Davis [10] that a study at Hewlett-Packard revealed the fact that 60% of the inventory investment in their manufacturing and distribution system is due to demand uncertainty.

As pointed out in [15] one major theme in the continuing development of inventory theory is to incorporate more realistic assumptions about product demand into inventory models. In most industrial contexts, demand is uncertain and hard to forecast. Many demand histories behave like random walks that evolve over time with frequent changes in their directions and rates of growth or decline. Furthermore, as product life cycles get shorter, the randomness and unpredictability of these demand processes have become even greater. In practice, for such demand processes, inventory managers often rely on forecasts based on a time series of prior demand, such as a weighted moving average. Typically these forecasts are predicated on a belief that the most recent demand observations are the best predictors for future demand.

An interesting class of production/inventory control problems therefore considers the single-location, single-product case under non-stationary stochastic demand. This class has been widely studied because of its key role in practice. We assume a fixed procurement cost each time a replenishment order is placed, whatever the size of the order, and a linear holding cost on any unit carried over in inventory from one period to the next. Our objective is to minimize the expected total cost under a service level constraint, that is the probability that at the end of every time period the net inventory will not be negative. Early works in the area were heuristic (Silver [25] and Askin [2]). Bookbinder and Tan [7] proposed another heuristic, under the static–dynamic uncertainty strategy. In this strategy, the replenishment periods are fixed at the beginning of the planning horizon and the actual orders at future replenishment periods are determined only at those replenishment periods, depending upon the realized demand. The expected total cost is minimized under the minimal service-level constraint.

We focus on the work of Tarim and Kingsman [29], where the authors proposed a mathematical programming approach to compute near-optimal policy parameters

for the inventory control policy known as the *replenishment cycle policy* or (R,S) policy. A detailed discussion on the characteristics of (R,S) can be found in [11]. In this policy a replenishment is placed every R periods to raise the inventory level to the order-up-to-level S. This provides an effective means of damping planning instability (deviations in planned orders, also known as *nervousness* [12, 16]) and coping with demand uncertainty. As pointed out by Silver et al. ([26], pp. 236–237), (R,S) is particularly appealing when items are ordered from the same supplier or require resource sharing. In these cases all items in a coordinated group can be given the same replenishment period. In [17] Janssen and de Kok discuss a two-supplier periodic model where one supplier delivers a fixed quantity while the amount delivered by the other is governed by an (R,S) policy. In [27] Smits et al. consider a production-inventory problem with compound renewal item demand. The model consists of stock-points, one for each item, controlled according to (R,S)-policies and one machine which replenishes them. Periodic review also allows a reasonable prediction of the level of the workload on the staff involved, and is particularly suitable for advanced planning environments and risk management [28]. For these reasons (R,S) is a popular inventory policy. Under the assumption of non-stationary demand it takes the form (R^n,S^n) where R^n denotes the length of the n^{th} replenishment cycle and S^n the corresponding order-up-to-level.

Tarim and Kingsman's formulation operates under the assumption that negative orders are not allowed, so that if the actual stock exceeds the order-up-to-level for that review, this excess stock is carried forward and not returned to the supply source. This event is assumed to be rare, and therefore its effects are ignored. As a direct consequence of this, the model only computes suboptimal policy parameters and an approximate expected total cost.

In this paper we exploit *stochastic constraint programming*, a novel modeling framework introduced by Walsh [35], to fully model the original stochastic programming formulation for computing (R^n, S^n) policy parameters. In our approach we extend the original framework with a new concept, *global chance-constraints*, and we employ this to compute optimal (R^n, S^n) policy parameters and the exact expected total cost for a given parameter configuration. By using optimal solutions provided by our model we gauge the accuracy of the solutions provided by Tarim and Kingsman's approach for a set of instances. In our experiments we show that the assumption adopted in Tarim and Kingsman's model are justified and that their model constitutes a valid trade-off for computing near-optimal (R^n, S^n) policy parameters when a short computational time is required.

This paper is organized as follows. In Section 2 we provide some formal background about different modeling techniques employed in this paper: stochastic programming, constraint programming, stochastic constraint programming and inventory control models. In Section 3 we review the existing approaches developed in the literature to compute (R^n, S^n) policy parameters. In Section 4 we introduce *global chance-constraints* and we present a novel *stochastic constraint programming* approach, based on this new concept, to compute optimal (R^n, S^n) policy parameters. In Section 5 we compare results produced by our exact approach with those provided by the state-of-the-art MIP approach for computing near-optimal (R^n, S^n) policy parameters. In Section 6 we draw conclusions.

2 Formal Background

In this paper we employ and merge several different modeling techniques. In this section some formal background and references are given for each technique exploited.

2.1 Stochastic Programming

Stochastic programming [6] 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 application areas 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* [9], 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* [6], which optimizes the expected objective function subject to some expected constraints. Another method, *chance-constrained programming*, was pioneered by Charnes and Cooper [9] 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 [26]. 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. Some kind of 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.

2.2 Constraint Programming

A *Constraint Satisfaction Problem* (CSP) [1, 8, 20] is a triple $\langle V, C, D\rangle$, where V is a set of decision variables, D is a function mapping each element of V to a domain of potential values, and C is a set of constraints stating allowed combinations of values for subsets of variables in V. A *solution* to a CSP is simply a set of values of the variables such that the values are in the domains of the variables and all of the constraints are satisfied. 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. Alternatively, we can define a constraint as a mathematical function: $f : D_1 \times D_2 \times \ldots \times D_n \rightarrow \{0, 1\}$ such that $f(x_1, x_2, \ldots, x_n) = 1$ if and only if $C(x_1, x_2, \ldots, x_n)$ is satisfied. Using this functional notation, we can then define a

constraint satisfaction problem (CSP) as follows (see also [1]): given n domains D_1, $D_2, \ldots, D_n$ and m constraints $f_1, f_2, \ldots, f_m$ find $x_1, x_2, \ldots, x_n$ such that

$$f_k(x_1, x_2, \ldots, x_n) = 1, \quad 1 \leq k \leq m; \tag{1}$$

$$x_j \in D_j, \quad 1 \leq j \leq n. \tag{2}$$

The problem is only a feasibility problem, and no objective function is defined. Nevertheless, CSPs are also an important class of combinatorial optimization problems. Here the functions f_k do not necessarily have closed mathematical forms (for example, functional representations) and can be defined simply by providing the subset S of the set $D_1 \times D_2 \times \ldots \times D_n$, such that if $(x_1, x_2, \ldots, x_n) \in S$, then the constraint is satisfied.

We now recall some key concepts in *Constraint Programming* (CP): constraint filtering algorithm, constraint propagation and arc-consistency [22]. In CP a filtering algorithm is typically associated with every constraint. This algorithm removes values from the domains of the variables participating in the constraint that cannot belong to any solution of the CSP. These filtering algorithms are repeatedly called until no new deduction can be made. This process is called propagation mechanism. In conjunction with this process CP uses a search procedure (like a backtracking algorithm) where filtering algorithms are systematically applied when the domain of a variable is modified. One of the most interesting properties of a filtering algorithm is arc-consistency. We say that a filtering algorithm associated with a constraint establishes arc-consistency if it removes all the values from the domains of the variables involved in the constraint that are not consistent with the constraint. As a consequence of results in [23], where authors proved that any non-binary constraint can be translated into an equivalent binary one with additional variables, several studies on arc-consistency were limited to binary constraints. However modeling problems by means of binary constraints presents several drawbacks. Firstly these constraints are poor in term of expressiveness. Secondly the domain reduction achieved by the respective filtering algorithm associated is typically weak. In order to overcome both these problems constraints that capture a relation among a non-fixed number of variables were introduced. These constraints not only are more expressive than the respective aggregation of simple constraints, but they can be associated with more powerful filtering algorithms that take into account the simultaneous presence of simple constraints to further reduce the domains of the variables. These constraints are called *global constraints*. One of the most well known examples is the `alldiff` constraint [21], both because of its expressiveness and its efficiency in establishing arc-consistency.

2.3 Stochastic Constraint Programming

In [35] and [33] a *stochastic constraint satisfaction problem* (stochastic CSP) is defined as a 6-tuple $\langle V, S, D, P, C, \theta \rangle$, where V is a set of decision variables and S is a set of stochastic variables, D is a function mapping each element of V and each element of S to a domain of potential values. A decision variable in V is *assigned* a value from its domain. P is a function mapping each element of S to a probability distribution for its associated domain. C is a set of constraints. A constraint $h \in C$ that constrains at least one variable in S is a *chance-constraint*. θ_h is a threshold value in the interval

[0, 1], indicating the minimum satisfaction probability for chance-constraint h. Note that a chance-constraint with a threshold of 1 is equivalent to a hard constraint.

A stochastic CSP consists of a number of *decision stages*. Solving a stochastic CSP implies a two step process.

In the first step a *policy of response* has to be defined. A policy of response states the rules that decide when decision variables have to be set. There are two extreme policies: here-and-now and wait-and-see. The *here-and-now* policy sets all decision variables before observing the realization of the random variables. A solution can be therefore expressed as an assignment for decision variables in V. The *wait-and-see* policy delays as much as possible the assignment of a value to a decision variable. Therefore a decision variable $x_i \in V$ is set to a value only after the realizations of stochastic variables $y_1, \ldots, y_{i-1} \in S$ have been observed. Under this policy typically the solution of a stochastic CSP is represented by means of a *policy tree* [33]. A policy tree is a tree of decisions where each path represents a different possible scenario (set of values for the stochastic variables) and the values assigned to decision variables in this scenario. Hybrid policies can be defined by stating at which stage k, $1 \leq k \leq j$ a decision variable x_j has to be set. The solution for any policy that is not a pure *here-and-now* will be expressed in general as a policy tree.

In the second step we solve the stochastic CSP under the given policy by finding specific *policy parameters*. In a one-stage stochastic CSP, the decision variables are set before the stochastic variables and the chosen policy is *here-and-now*. Under any other policy, that is *wait-and-see* or hybrid, we have an m-stage stochastic CSP where V and S are partitioned into disjoint sets, $V_1, \ldots, V_m$ and $S_1, \ldots, S_m$. To solve an m-stage stochastic CSP an assignment to the variables in V_1 must be found such that, given random values for S_1, an assignment can be found for V_2 such that, given random values for $S_2 \ldots$, an assignment can be found for V_m so that, given random values for S_m the hard constraints are satisfied and the chance-constraints are satisfied in the specified fraction of all possible scenarios.

In [35] a policy based view of stochastic constraint programs is proposed. The semantics is based on a tree of decisions. Each path in a policy represents a different possible scenario (set of values for the stochastic variables), and the values assigned to decision variables in this scenario. To find satisfying policies, backtracking and forward checking algorithms, which explores the implicit AND/OR graph, are presented. Such an approach has been further investigated in [3]. An alternative semantics for stochastic constraint programs, which suggests an alternative solution method, comes from a scenario-based view [6]. In [33] the authors outline this solution method, which consists in generating a scenario-tree that incorporates all possible realizations of discrete random variables into the model explicitly. The great advantage of such an approach is that conventional constraint solvers can be used to solve stochastic CSP. Of course, there is a price to pay in this approach, as the number of scenarios grows exponentially with the number of stages and such a growth is particularly affected by random variables that contain a wide range of values in their domain. To deal with this problem the authors developed dedicated scenario-reduction techniques, which unfortunately affect the completeness of the approach when applied to improve performances of the search process. Another limit of the approaches in [35] and [33] is that they provide implementations only for a *wait-and-see* policy. The reason for this is that, when decision and random variables are split into disjoint sets $V_1, \ldots, V_m$ and $S_1, \ldots, S_m$ containing more than one element, the

computation required to find policy parameters usually is special purpose and it is unlikely to be performed by a general approach.

2.4 Inventory Control and (R^n,S^n) Policy

In this paper we consider the class of production/inventory control problems that refers to the single location, single product case under non-stationary stochastic demand. 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 that at the end of every period the net inventory will not be negative to be at least a given value α. 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 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 [26]. In what follows the problem described above will be solved adopting the replenishment cycle policy (R^n,S^n). We recall that R^n denotes the length of the nth replenishment cycle and S^n the respective order-up-to-level (Fig. 1). In this policy the actual order quantity X_n for replenishment cycle n is determined only after the demand in former periods has been realized. X_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 $n = \{1, \ldots, N\}$.

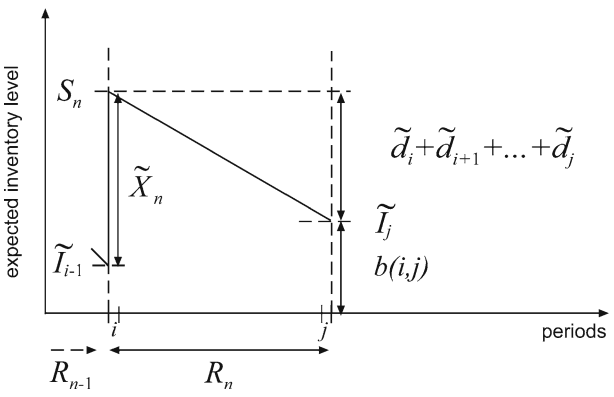

Fig. 1 (R^n,S^n) policy. $\tilde{d}_i + \tilde{d}_{i+1} + \ldots + \tilde{d}_j$ is the expected demand over R^n; $b(i, j)$ is the minimum buffer stock required to guarantee service level α; $\tilde{X}_n$ is the expected order quantity in period i for replenishment cycle n; $\tilde{I}_{i-1}$ and $\tilde{I}_j$ are respectively the expected closing-inventory-levels for periods $i-1$ and j

3 Existing Approaches

Early works in stochastic inventory control area adopted heuristic strategies such as those proposed by Silver [25], Askin [2] and Bookbinder and Tan [7]. The first complete (MIP) solution method, which operates under mild assumptions, was introduced for this problem by Tarim and Kingsman [29]. Tarim and Smith [30] introduced a more compact and efficient CP formulation for the same model. Dedicated *cost-based filtering* techniques for such a CP model were presented in [31] and [32]. This latter enhanced model proved to be able to solve real world problem instances considering up to a 50 periods planning horizon in a few seconds. In the following sections we discuss the assumptions adopted by Tarim and Kingsman and we propose a stochastic constraint programming approach in which these assumptions are dropped. By means of this approach we can compute optimal (R^n, S^n) policy parameters and the real associated expected total cost. Of course there is a price to pay for dropping Tarim and Kingsman's assumptions, in fact our approach is less efficient than the one proposed in [32].

3.1 Stochastic Programming Model

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 non-negative replenishment orders, X_t in period t, with the objective of minimizing the expected total cost $E\{TC\}$ over a finite planning horizon of N periods. The model is given below:

$$\min \quad E\{TC\} = \int_{d_1}\int_{d_2}\dots\int_{d_N}\sum_{t=1}^{N}(a\delta_t + h\cdot\max(I_t,0)) \\ g_1(d_1)g_2(d_2)\dots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\dots\mathrm{d}(d_N) \tag{3}$$

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

$$\delta_t = \begin{cases} 1, & \text{if} \quad X_t > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$I_t = I_0 + \sum_{i=1}^{t}\left(X_i - d_i\right) \tag{5}$$

$$\Pr\{I_t \geq 0\} \geq \alpha \tag{6}$$

$$I_t \in \mathbb{R}, \quad X_t \geq 0, \quad \delta_t \in \{0,1\}. \tag{7}$$

The demand d_t in each period is a continuous random variable with probability distribution function $g_t(d_t)$. 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 ($\delta_t = 0$). Chance-constraint (6) enforces the required service level, that is the probability α the net inventory will not be negative at the end

of each and every time period. The objective function (3) minimizes the expected total cost over the given planning horizon.

Although this stochastic programming approach fully models our production/inventory problem, a solution cannot be expressed before a *response policy* is chosen. We have already seen that a policy states the rules to decide when decision variables have to be set. By using the general approach proposed in [33] a solution can be found under *wait-and-see* policy. In this policy a replenishment decision X_k for period k is made only after all the outcomes for random variables associated with former periods $1, \ldots, k-1$ have been observed. The solution therefore is expressed as a policy tree, which can exponentially grow in dimension even for short planning horizons.

In order to avoid this intractable solution, approaches based on order-up-to-level strategies have typically been proposed for this model in the literature. Expressing replenishment decisions in terms of order-up-to-levels instead of order quantities is a convenient way to find optimal policy parameters without employing an exponential solution tree. An order-up-to-level for period k represents the level to which stocks have to be maintained at the beginning of such a period. Therefore at the beginning of each period k, $k = 1 \ldots, N$, in our planning horizon we can observe the actual inventory level and we can decide if an order has to be issued to bring the inventory up to the required level. There are two well-known order-up-to-level policies for the general model proposed.

The so-called (s^n,S^n) policy [26] is a pure *wait-and-see* policy where at the end of period k we observe the inventory level and if this level is below s^k, then an order is issued to raise stocks up to level S^k. It is easy to see that this policy is *wait-and-see* since every decision, placing or not an order and the actual size of the order, is taken at the very last moment, by observing the demands that have been realized in the former periods. Furthermore a solution under this policy can be expressed by using only N pairs (s^k,S^k), in contrast to the exponential solution tree required when the problem is modeled using order quantities.

A hybrid order-up-to-level policy is the so-called (R^n,S^n) policy [7], also known as replenishment cycle policy, which we described above. In this policy the inventory review times are set under a *here-and-now* strategy at the beginning of the planning horizon. These decisions are not affected by the actual demand realized in each period. On the other hand, for each inventory review we need to observe the actual demand realized in former periods to compute the actual order quantity. This makes the (R^n,S^n) policy hybrid, since the order quantity for each review is computed in a *wait-and-see* fashion only after previous demands have been realized. Also in this case the solution can be efficiently expressed. In fact we only require M $(\leq N)$ couples of values (R^k,S^k), $k = 1, \ldots, M$, where R^k is the length of the k-th replenishment cycle and S^k is the respective order-up-to-level.

From these considerations, and from the well known Jensen's inequality [6], it is easy to see that an (s^n,S^n) policy always has a lower expected total cost than an (R^n,S^n) policy. The optimality of the (s^n,S^n) policy has been presented in [24]. In what follows we will focus on the (R^n,S^n) policy. In fact, as already discussed, despite being suboptimal this policy presents several interesting aspects.

In the next section we will recall a CP model proposed by Tarim and Smith [30] and based on a *deterministic equivalent* mathematical programming (MIP) model originally introduced by Tarim and Kingsman in [29] to compute (R^n,S^n)

policy parameters. This model can only provide near-optimal policy parameters because it relies on assumptions that affect optimality. In the following section these assumptions are discussed.

3.2 Tarim and Kingsman's Approach

In this section we provide a description of the *deterministic equivalent* CP formulation for the (R^n,S^n) policy proposed by Tarim and Smith in [30] and based on the approach originally introduced by Tarim and Kingsman in [29]. It should be noted that this formulation is the discrete version of the model presented in Section 3.1. Since the normal distribution is the limiting case of a discrete binomial distribution $P_p(k|n)$[1] as the sample size n becomes large,[2] in the discrete model an uniformly distributed random demand with mean μ and variance σ^2 can be modeled as a discrete random variable following a binomial probability mass function $P_p(k|n)$, where $np = \mu$ and $np(1-p) = \sigma^2$.

The *deterministic equivalent* CP formulation for the (R^n,S^n) policy proposed in [30] is

$$\min \; E\{TC\} = \sum_{t=1}^{N} \left(a\delta_t + h\tilde{I}_t \right) \tag{8}$$

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

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} \geq 0 \tag{9}$$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} > 0 \Rightarrow \delta_t = 1 \tag{10}$$

$$\tilde{I}_t \geq b \left(\max_{j \in \{1..t\}} j \cdot \delta_j, t \right) \tag{11}$$

$$\tilde{I}_t \in \mathbb{Z}^+ \cup \{0\}, \quad \delta_t \in \{0, 1\} \tag{12}$$

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

$$b\,(i,\ j\,) = G^{-1}_{d_i+d_{i+1}+\ldots+d_j}(\alpha) - \sum_{k=i}^{j} \tilde{d}_k. \tag{13}$$

$G_{d_i+d_{i+1}+\ldots+d_j}$ 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. Unfortunately the computation of the binomial cumulative distribution function is time consuming. For this reason it is common to adopt an approximate approach that exploits the respective normal cumulative distribution function,[3] whose computation is much easier. In what follows we will adopt this approach not only for its efficiency,

[1] The binomial distribution gives the discrete probability distribution $P_p(k|n)$ of obtaining exactly k successes out of n Bernoulli trials [18].

[2] In which case $P_p(k|n)$ is normal with mean $\mu = np$ and variance $\sigma^2 = np(1-p)$.

[3] This approximation is a huge time-saver (exact calculations of $P_p(k|n)$ with large n are very onerous); it can be seen as a consequence of the central limit theorem [18] since $P_p(k|n)$ is a sum of n independent, identically distributed 0–1 indicator variables.

but also because it lets us comply in the discrete model with the original problem definition that assumes a normally distributed demand in each period. We will therefore compute buffer stock levels as

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

where $d_i, d_{i+1}, \ldots, d_j$ are normally distributed random variables. The term $G^{-1}_{d_i+d_{i+1}+\ldots+d_j}(\alpha)$ is rounded to the nearest integer—function round$(\cdot)$—according to the known concept of *continuity correction* (see [13]) in probability theory. For a detailed discussion on this CP model see [31]. Each decision variable $\tilde{I}_t$ represents the expected inventory level at the end of period t. It should be noted that the expected inventory level at the beginning of such a period is simply $\tilde{I}_t + \tilde{d}_t$ and if a replenishment is scheduled in t this latter value denotes the order-up-to-level (S^n) in period t. Each $\tilde{d}_t$ represents the expected demand in a given period t according to its probability mass 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 (8) 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 (9) 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 (10) 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. Constraint (11) enforces 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 every time period, the probability that the net inventory will not be negative is at least α. These buffer

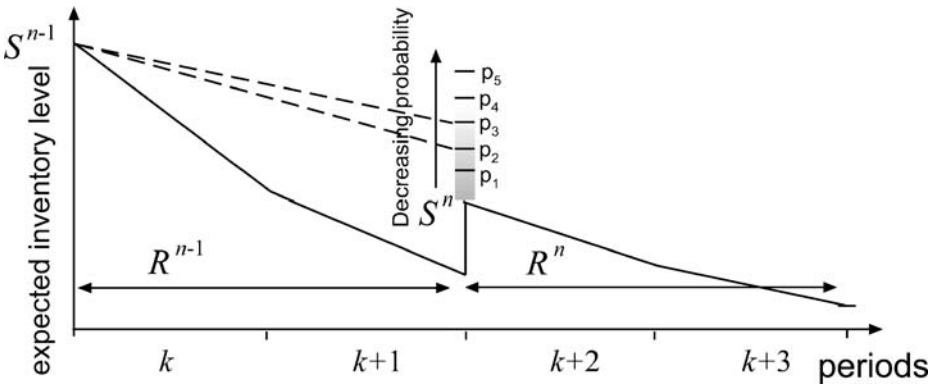

Fig. 2 In Tarim and Kingsman [29] the event that actual stock exceeds the order-up-to-level S^n for a given review R^n is assumed to be rare. In other words, in their model observing a low demand during R^{n-1} has negligible probability. This implies that probabilities $p_1, p_2, \ldots, p_m$ are assumed to be low

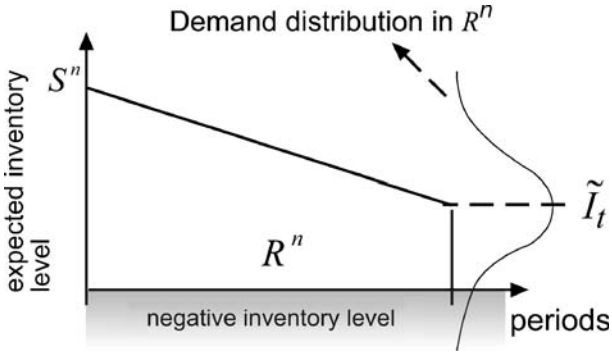

Fig. 3 Negative inventory levels

stocks, which are stored in matrix $b(\cdot, \cdot)$, are pre-computed following the approach originally suggested in [29].

The CP formulation operates under the assumption that negative orders are not allowed, so that if the actual stock exceeds the order-up-to-level for that review, this excess stock is carried forward and not returned to the supply source. However this event is assumed to be rare, therefore in the model it is ignored (Fig. 2).

Let us analyze the effects of this assumption on the solutions produced by the CP approach.

1. The cost of carrying excess stock as a consequence of a low demand before a given replenishment is ignored, therefore the actual cost of a policy can be higher than the one provided by the model.
2. The event of carrying excess stock as a consequence of low demand before a given replenishment can have an impact on the service level of next periods. In particular, when the probability of ending up with a stock level higher than the order-up-to-level fixed in a given replenishment period is sufficiently high, it could be possible to exploit excess stock to provide the required service level, keeping lower expected closing inventory levels in following periods.

Furthermore, the CP approach models holding cost by considering expected closing-inventory-level values $\tilde{I}_t$ in each period (Fig. 3), while in the original stochastic programming formulation negative inventories do not contribute to the actual overall expected holding cost, which may be therefore higher than the one computed by the CP model.

4 A Stochastic Constraint Programming Approach Based on Global Chance-Constraints

In this section we provide a novel CP approach to find optimal (R^n, S^n) policy parameters. Our approach avoids both the assumptions adopted in Tarim and Kingsman [29], therefore it considers the effect of excess stock on the service level of subsequent replenishment cycles and on the expected total cost of a given policy. It also considers the fact that a negative closing-inventory-level does not contribute to the overall holding cost. The core of our modeling strategy is the new concept of *global chance-constraints*. By means of this novelty we are able to dynamically

compute the exact service level provided by a given policy parameter configuration and the expected total cost associated with it.

4.1 Chance-Constraints and Policies

The techniques proposed in [35] and [33] for solving stochastic CSPs are general-purpose but limited to *wait-and-see* policies. Since in the inventory control problem presented we apply a hybrid policy, we adopt a different and specialized approach. By recalling that we can define a constraint as a mathematical function, in a similar fashion it is possible to define a *chance-constraint*, originally introduced by Charnes and Cooper [9], as a mathematical function. Depending on the chosen policy the domain of our function f will change. For instance if we restrict ourselves to a *here-and-now* policy, so that the solution for our stochastic CSP can be expressed as a simple assignment for the decision variables, the function will be $f : D(x_1) \times \ldots \times D(x_n) \rightarrow \{0, 1\}$, where $V = \{x_1, \ldots, x_n\}$, and $f(x_1, \ldots, x_n) = 1$ if and only if $x_1, \ldots, x_n$ is an assignment such that, given random values for $y_1, \ldots, y_n$, where $S = \{y_1, \ldots, y_n\}$ the hard constraints are satisfied and the chance-constraints are satisfied in the specified fraction of all possible scenarios. In a *wait-and-see* policy as we have seen $V_1 = \{x_1\}, \ldots, V_n = \{x_n\}$ and $S_1 = \{y_1\}, \ldots, S_n = \{y_n\}$. Therefore the function $f(x_1, x_2, \ldots, x_n)$ will map each possible *policy tree* in the solution space identified by our chance-constraint to the two possible values $\{0, 1\}$. $f(x_1, x_2, \ldots, x_n) = 1$ if and only if the assignment for the variable x_1 is such that, given a random value for y_1, an assignment can be found for variable x_2 such that, given a random value for $y_2 \ldots$, an assignment can be found for variable x_m so that, given a random value for y_m the hard constraints are satisfied and the chance-constraints are satisfied in the specified fraction of all possible scenarios. These functions can obviously be expressed in theory for any possible policy.

4.2 Global Chance-Constraints

We recalled a known concept in stochastic programming: chance-constraints. We also saw in former sections how CP can be extended to consider random variables and chance-constraints. This leads to what is called *stochastic constraint programming*. We now aim to extend stochastic constraint programming with a new concept in analogy to what has been done for CP. We already saw in Section 2 that in CP the simultaneous presence of several simple constraints, for efficiency and expressiveness, is typically modeled by means of *global constraints*. Also in *stochastic programming* we can identify simple chance-constraints of the form $\Pr\{D \geq r\} \geq \alpha$, typically involving a decision variable D and a random variable r. An example is given by the service level at period t in our inventory control problem, $\Pr\{I_t \geq 0\} \geq \alpha$. These simple chance-constraints in stochastic programming typically appear as a set. In our inventory model we enforce a service level constraint for every period in our planning horizon, that is we replicate $\Pr\{I_t \geq 0\} \geq \alpha$, for $t = 1, \ldots, N$. In a *stochastic constraint programming* framework it is therefore natural to group this set of simple chance-constraints and to define what we will call a *global chance-constraint* over a set of decision variables and a set of random variables. The general signature for a global chance-constraint will be

$$globalChanceConstraint(D_1, \ldots, D_N, r_1, \ldots, r_N, \alpha),$$

where $D_1, \ldots, D_N$ are decision variables $r_1, \ldots, r_N$ are random variables and α is a value in the interval [0, 1], indicating the minimum satisfaction probability for the chance-constraint. According to the probability distribution functions of random variables, the filtering algorithm of this constraint will prune values from domains of $D_1, \ldots, D_N$ that cannot guarantee the chance-constraints are satisfied at the required threshold probability. Depending on the given problem and on the response policy chosen, dedicated efficient filtering algorithms can be implemented (see the forward checking technique proposed by Walsh [35] for *wait-and-see* policies, and the improved algorithm in [3]).

This new concept defines much more than a notation extension. In fact it should be noted that stochastic programming is a very high level modeling framework. An apparently simple constraint like the one presented, $\Pr\{I_t \geq 0\}$, actually hides in the stochastic programming model interdependencies between several, and often all, decision variables and random variables in the problem. Usually evaluating these dependencies requires the computation of a convolution integral. Therefore in general it will not be possible to express a global chance-constraint in stochastic constraint programming as a set of simple and independent chance-constraints. An immediate example is given by Tarim and Smith's model [30]. Here the *chance-constraints* in the stochastic programming model are modeled as independent deterministic equivalent constraints according to the approach proposed by Tarim and Kingsman [29]. As discussed in the former sections this leads to several approximations, since many dependencies between decision and random variables are ignored. In the following sections we introduce a global chance-constraint able to model these dependencies.

4.3 A Global Chance-Constraint for (R^n,S^n) Policy

We focus on the (R^n,S^n) policy, which is hybrid and therefore cannot be solved by means of the approaches in [3, 33] that only cope with wait-and-see policies. As already discussed, by reasoning in terms of order-up-to-levels, under this policy a solution for our stochastic model can be efficiently expressed as an assignment for our decision variables, that is replenishment decisions and order-up-to-levels, and it does not require a tree representation. We developed a dedicated *global chance-constraint* that identifies feasible policy parameters for our inventory control problem. As in the case of hard constraints the function does not necessarily have closed mathematical form. In our case this function is defined by providing an algorithm able to identify feasible assignments for decision variables, i.e. policy parameters. Within the same constraint we also developed an algorithm to compute the expected total cost for a given policy parameter configuration. The signature of our global chance-constraint is as follows

$$serviceLevelRS(C, a, h, \tilde{I}, \delta, d, \alpha)$$

where C is a decision variable denoting the expected total cost, a is the fixed ordering cost, h is the holding cost per unit, $\tilde{I}$ and δ are arrays of decision variables, d is an array of discrete random variables d_t with probability mass function $g_t(d_t)$ and α is the required service level. This constraint ensures that, at the end of each time period, the probability that the net inventory will not be negative is at least α. It is therefore semantically equivalent to Constraint (6) for $t = \{1, \ldots, N\}$ and it can be used to express these constraints in a CP model. The decision variable C represents a

lower bound on the expected total cost (3) for a given partial assignment for decision variables $\tilde{I}$ and δ, and such a bound is tight when all the decision variables $\tilde{I}$ and δ are ground. It should be noted that the *global view* provided by this constraint allows us to consider joint probabilities during the search when service levels and the expected total cost are computed. These joint probabilities are ignored when the same condition is expressed by means of many independent constraints as in Tarim and Smith [30]. In the following sections we will describe the deterministic equivalent CP model that incorporates our global chance-constraint and the propagation logic for the constraint.

4.4 Deterministic Equivalent Model

The deterministic equivalent model that incorporates our constraint is

$$\min \; E\{TC\} = C \tag{14}$$

subject to

$$serviceLevelRS(C, a, h, \tilde{I}_{t\in\{1,\ldots,N\}}, \delta_{t\in\{1,\ldots,N\}}, d_{t\in\{1,\ldots,N\}}, \alpha) \tag{15}$$

and for $t = 1 \ldots N$,

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} \geq 0 \tag{16}$$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} > 0 \Rightarrow \delta_t = 1 \tag{17}$$

$$\tilde{I}_t, C \in \mathbb{Z}^+ \cup \{0\}, \quad \delta_t \in \{0, 1\}. \tag{18}$$

It is easy to see that the model is similar to the one proposed in [30] and presented in Section 3.2. Again we observe two sets of decision variables: the replenishment decision in period t, δ_t; and the expected closing-inventory-level in period t, $\tilde{I}_t$. The buffer stocks needed to provide the required service level α and the expected total cost C for a given policy are computed by the special purpose global chance-constraint.

4.5 Propagating the Service Level Global Chance-Constraint

In order to propagate our constraint and compute a feasible assignment for the expected closing-inventory-levels $\tilde{I}$, we will consider now a two-replenishment cycle case (Fig. 4) in a four-period planning horizon, then we will extend the idea in a recursive fashion to the case of M subsequent replenishment cycles $\{R^1, \ldots, R^M\}$ over N periods. Two consecutive replenishment cycles are planned over the planning horizon considered, let us call them R^1 and R^2. R^1 covers periods $\{1, 2\}$, R^2 periods $\{3, 4\}$. Let S^i be the opening inventory level for R^i and $\Pr\{d_i \leq D\}$ be the probability of the event "observing a demand in period i less than or equal to D", where d_i is a random variable that represents the distribution of the demand in period i. In a simple newsvendor problem [26] over one period with random demand d, the opening-inventory-level that provides a service level α can be computed as $G_d^{-1}(\alpha)$, where G_d^{-1} is the inverse cumulative distribution function of d. It is easy to see that $S^1 = G_{d_1+d_2}^{-1}(\alpha)$ and the correct minimum opening-inventory-level S^2 for R^2, which

guarantees the required service level α, can be computed from the following relation that mixes *scenario-based approach* and *chance-constrained programming*

$$\Pr\{d_1 + d_2 \geq S^1 - S^2\} \cdot G_{d_3+d_4}(S^2) + \sum_{i=0}^{S^1-S^2} \left(\Pr\{d_1 + d_2 = i\} \cdot G_{d_3+d_4}(S^1 - i)\right) \geq \alpha, \tag{19}$$

where $G_{d_i+d_{i+1}+\ldots+d_j}(\cdot)$ is the cumulative probability distribution function of $d_i + d_{i+1} + \ldots + d_j$. For the two replenishment cycles case, this can be rewritten using the following extended form

$$\left(1 - G_{d_1+d_2}(S^1 - S^2 - 1)\right) \cdot G_{d^3+d^4}(S^2) + \sum_{i=0}^{S^1-S^2} \left(G_{d_1+d_2}(i) - G_{d_1+d_2}(i-1)\right) \cdot G_{d_3+d_4}(S^1 - i) \geq \alpha. \tag{20}$$

Notice that if S^1 is smaller than S^2, obviously the former cycle has no influence on the computation of S^2 and Condition (19) becomes $G_{d_3+d_4}(S^2) \geq \alpha$. Furthermore, if the computed S^2 is such that $S^2 < S^1 - \tilde{d}_1$, we just set S^2 to the minimum value allowed, that is $S^1 - \tilde{d}_1$.

Finally observe that the term

$$\sum_{i=1}^{S^1-S^2} \left(G_{d_1+d_2}(i) - G_{d_1+d_2}(i-1)\right) \cdot G_{d_3+d_4}(S^1 - i)$$

in Condition (20) has to be multiplied by the normalization term

$$G_{d_1+d_2}(S^1 - S^2 - 1) \Big/ \sum_{i=0}^{S^1-S^2} \left(G_{d_1+d_2}(i) - G_{d_1+d_2}(i-1)\right)$$

in order to guarantee that the sum of all the event probabilities is one. In fact negative demands are disregarded, but the respective probabilities must be taken into account to cover the space of all possible events.

In order to propagate (Algorithm 1: `propagate`) this constraint in the case of M subsequent replenishment cycles over N periods, at each node of the search tree we look for the first M consecutive replenishment cycles (Algorithm 1, line 2) identified by the current partial assignment for decision variables δ. Two replenishment cycles R^m, R^{m+1} are consecutive if the last period of R^m is g and the first period of R^{m+1} is $g + 1$. A replenishment cycle R^k over periods $\{i, \ldots, j\}$ can be identified by a full assignment over $\delta_i, \ldots, \delta_{j+1}$ where δ_i, δ_{j+1} are set to 1 and $\delta_{i+1}, \ldots, \delta_j$ are set to 0 (Function `listCycles()`). The opening-inventory-level S^1 for the first replenishment cycle R^1 covering periods $\{1, \ldots, j\}$ can be easily computed as $G^{-1}_{d_1+\ldots+d_j}(\alpha)$. In what follows we will describe a recursive *scenario-based approach* [6] to compute the opening-inventory-level S^j required in replenishment cycle $j \in \{1, \ldots, M\}$. We will assume that opening-inventory-levels for $R^1, \ldots, R^{j-1}$ are known (Algorithm 1, line 8) and we will use a generalized version of Condition (19) to compute such a value (Algorithm 1, lines 19 to 21). A generalized version of (19) for the case of M replenishment cycles can be introduced by

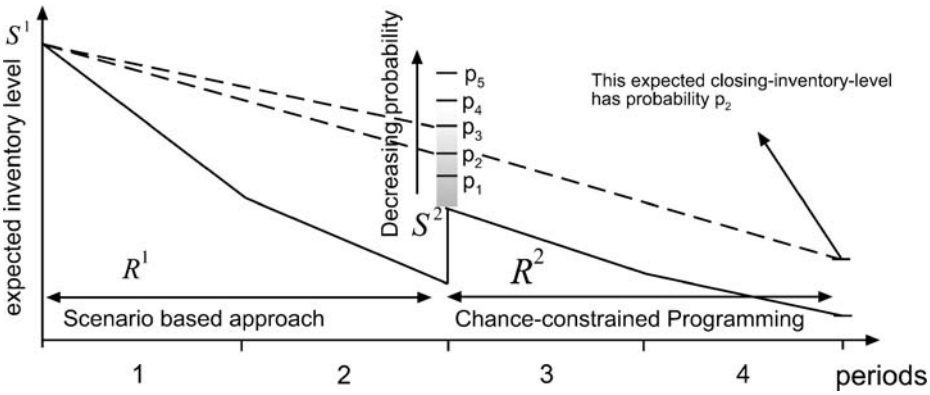

Fig. 4 Two replenishment cycle case

observing that S^j, $j \in \{1, \ldots, M\}$, the opening-inventory-level for opening-inventory-level for replenishment cycle R^j, is affected only by former replenishment cycles $\{R^i, \ldots, R^{j-1}\}$, where $i = \min\{v \in \{1, \ldots, j\} | (S^v \geq S^1) \wedge \ldots \wedge (S^v \geq S^{v-1})\}$. If $i = j$ no former replenishment cycle affects R^j. Now since we know the distribution of the demand in replenishment cycles $\{R^i, \ldots, R^j\}$ and under the assumption that former opening-inventory-levels $\{S^i, \ldots, S^{j-1}\}$ have been already set, it is easy to recursively compute the expected service level for replenishment cycle R^j by using a *scenario based approach*. We can therefore extend Condition (19) to compute S^j for R^j given that $\{R^i, \ldots, R^{j-1}\}$ are the former periods affecting service level of R^j.

Let $P_j(S^j)$ be the probability of observing an inventory level of S^j, that is the opening-inventory-level R^j, at the beginning of R^j.

Let $P_j(S^j, h)$ be the probability of observing an inventory level of $S^j + h$, that is h units higher than the opening-inventory-level of R^j, at the beginning of R^j.

Given $q \in \mathbb{Z}^+ \cup \{0\}$ and $k \in \{i, \ldots, M\}$, the probability associated with the event "observing a demand less or equal to q in replenishment cycle R^k" can be easily computed. Such a probability is in fact $G_{d_{R^k}}(q)$, where d_{R^k} is the demand distribution in replenishment cycle R^k, that is, if R^k covers periods $\{m, \ldots, n\}$, $d_{R^k} = d_m + \ldots + d_n$. Let $\widehat{G}_{d_{R^k}}(q)$ be the element of probability $G_{d_{R^k}}(q) - G_{d_{R^k}}(q-1)$.

– If $S^{j-1} \geq S^j$, then $P_j(S^j)$ is computed as

$$P_{j-1}(S^{j-1}) \cdot \left(1 - G_{d_{R^{j-1}}}(S^{j-1} - S^j - 1)\right) + \sum_{k=1}^{S^i - S^{j-1}} P_{j-1}(S^{j-1}, k) \cdot \left(1 - G_{d_{R^{j-1}}}(S^{j-1} - S^j + k - 1)\right) \quad (21)$$

that is $P_{j-1}(S^{j-1})$ multiplied by the probability of the event "observing a demand greater or equal to $S^{j-1} - S^j$ in replenishment cycle R^{j-1}", plus the summation, for $k = 1, \ldots, S^i - S^{j-1}$, of $P_{j-1}(S^{j-1}, k)$ multiplied by the probability of the event "in R^{j-1} we observe a demand greater or equal to $S^{j-1} - S^j + k$".

– If $S^{j-1} < S^j$, then $P_j(S^j)$ is computed as

$$P_{j-1}\left(S^{j-1}\right) + \sum_{k=1}^{S^j - S^{j-1}} P_{j-1}\left(S^{j-1}, k\right) + \sum_{k=1}^{S^i - S^j} P_{j-1}\left(S^{j-1}, S^j - S^{j-1} + k\right) \cdot \left(1 - G_{d_{R^{j-1}}}(k-1)\right) \tag{22}$$

– If $S^{j-1} \geq S^j + h$, then $P_j\left(S^j, h\right)$ is computed as

$$P_{j-1}\left(S^{j-1}\right) \cdot \widehat{G}_{d_{R^{j-1}}}\left(S^{j-1} - S^j - h\right) + \sum_{k=1}^{S^i - S^{j-1} - h} P_{j-1}\left(S^{j-1}, k\right) \cdot \widehat{G}_{d_{R^{j-1}}}\left(S^{j-1} - S^j - h + k\right) \tag{23}$$

– If $S^{j-1} < S^j + h$, then $P_j\left(S^j, h\right)$ is computed as

$$\sum_{k=S^j+h-S^{j-1}}^{S^i - S^{j-1}} P_{j-1}\left(S^{j-1}, k\right) \cdot \widehat{G}_{d_{R^{j-1}}}\left(k - S^j - h + S^{j-1}\right). \tag{24}$$

Obviously $P_i(S^i) = 1$ since, for the way R^i is chosen, no former replenishment cycle may affect its order-up-to-level S^i. By following a *dynamic programming* [4] scheme, S^j can be computed as the minimum value that satisfies

$$P_j(S^j) \cdot G_{d_{R^j}}(S^j) + \sum_{k=1}^{S^i - S^j} \left(P_j(S^j, k) \cdot G_{d_{R^j}}(S^j + k)\right) \geq \alpha. \tag{25}$$

Since this paper is not focused on efficiency issues, the dynamic programming algorithm developed to implement (25) simply employs a recursive code structured as the functional equation itself. Nevertheless we want to underline that the proposed recursion only aims to describe a correct functional equation to compute feasible assignments. As in every dynamic program, efficiency can be obtained by adopting a forward recursion and by trading memory and time to avoid computing the probability of a given scenario more than once.

In the recursive computation scenarios with negative demands are not considered, therefore we must normalize the probabilities of other events in order to ensure that their sum covers the whole space of the possible events. In other words we need to ensure that the probability associated with area A in Fig. 5 is one. This is a known approach in inventory control and it is usually justified since the distortion introduced by this normalization typically does not affect the quality of the solutions. A possible way to perform this normalization step is to divide the term

$$P_j(S^j) \cdot G_{d_{R^j}}(S^j) + \sum_{k=1}^{S^i - S^j} \left(P_j(S^j, k) \cdot G_{d_{R^j}}(S^j + k)\right)$$

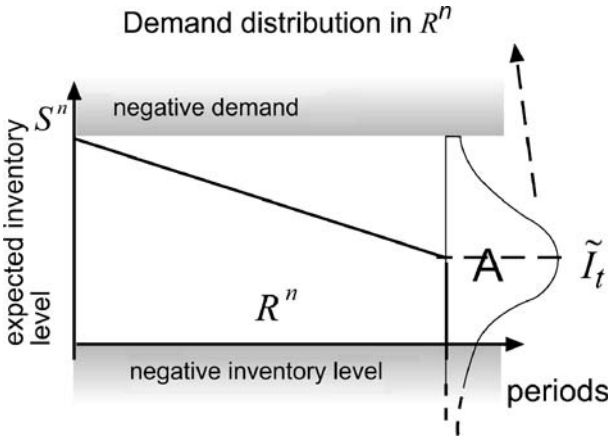

Fig. 5 Normalization

in Condition (25) by the following normalization term

$$P_j(S^j) + \sum_{i=k}^{S^i - S^j} P_j(S^j, k) \tag{26}$$

in order to guarantee that the sum of all the probabilities of the events considered in step j is one.

In order to speed up the search for the optimal opening-inventory-level associated with a given replenishment cycle R^k, recall that opening-inventory-levels computed as shown in [30] are always greater than or equal to optimal opening-inventory-level satisfying (25). Therefore an efficient strategy (Procedure `setBufferForCycle()`) for finding optimal opening-inventory-levels is to consider sequentially the first M replenishment cycles, R^k, $k \in \{1, \ldots, M\}$, identified by the current partial assignment for replenishment decisions δ. For each replenishment cycle R^k an upper-bound for the optimal opening-inventory-level can be computed as $\lceil G^{-1}_{d_{R^k}}(\alpha) \rceil$ (see [29]). Starting from this upper-bound we can decrease it and search for the minimum value that satisfies (25) (Procedure `setBufferForCycle()`, line 4). Opening-inventory-levels computed as in [29] are close to optimal because probabilities associated with negative order quantity scenarios are typically low, therefore this strategy requires only a few steps to reach the optimum levels.

4.6 Computing Holding Cost

In this section we address the problem of computing the correct holding cost for a given replenishment cycle R covering periods $\{i, \ldots, j\}$ when the expected closing-inventory-level $\tilde{I}_t$ for each period $t \in \{i, \ldots, j\}$ is given. We recall that $\tilde{I}_j$ denotes S^j minus the expected demand in replenishment cycle $j, \tilde{d}_{R^j}$. The problem of computing the exact holding cost arises from the fact that negative inventory levels do not contribute to the overall holding cost. Therefore the term $h\tilde{I}_t$ in the objective function of the model presented by Tarim and Kingsman is not a complete representation of this cost component. Once $\tilde{I}_j$ is known every other $\tilde{I}_k$, $k \in \{i, \ldots, j-1\}$ can be

easily computed as $\tilde{I}_k = \tilde{I}_j + \sum_{t=k+1}^{j} \tilde{d}_t$. Let $h(R, \tilde{I}_j)$ be the expected holding cost for replenishment cycle R when the expected closing-inventory-level $\tilde{I}_j$ is given. This cost component is made up of individual cost components for each period in our replenishment cycle R. Let us consider a given period $k \in \{i, \ldots, j\}$. The opening inventory level for R is $S^i = \tilde{I}_j + \sum_{t=i}^{j} \tilde{d}_t$. We recall that the probability of observing an overall demand r over the time span $\{i, \ldots, k\}$ is denoted by $\widehat{G}_{d_i+\ldots+d_k}(r)$. By letting r range from 0 to S^i we obtain every possible scenario for which a holding cost is incurred in period k. Therefore the expected holding cost for period k can be expressed as $h \sum_{r=0}^{S^i} (S_i - r) \cdot \widehat{G}_{d_i+\ldots+d_k}(r)$ and the expected holding cost for replenishment cycle R will be the sum of the contributions from every period $k \in \{i, \ldots, j\}$.

Algorithm 1: `propagate`

input : C, $\delta_1, \ldots, \delta_N$, $\tilde{I}_1, \ldots, \tilde{I}_N$, α, a, h, $d_1, \ldots, d_N$, N

begin
  $cycles \leftarrow listCycles(\delta_1, \ldots, \delta_N, \tilde{I}_1, \ldots, \tilde{I}_N, N)$;
  $n \leftarrow$ # elements in $cycles$;
  if $n = 0$ **then**
    return;
  $cost \leftarrow a \cdot n$;
  $condition \leftarrow true$;
  for *each element* e *in cycles* **do**
    let $\{i, \ldots, j\}$ be the span covered by e;
    if *no decision variable* $\tilde{I}_i, \ldots, \tilde{I}_j$ *is assigned* **then**
      $condition \leftarrow false$;
    else if $\exists k \mid$ *decision variable* $\tilde{I}_k$, $i \leq k \leq j$ *is assigned* **then**
      $S^i \leftarrow$ cycle opening inventory level of e, linearly dependent on $\tilde{I}_k$;
      $holdingCost \leftarrow$ cycle holding cost of e with opening inventory level S^i (27);
      $cost \leftarrow cost + holdingCost$;
  if $condition$ **then**
    $C \leftarrow cost$;
  else
    $setBufferForCycle(cycles, d_1, \ldots, d_N, \alpha)$;
    let e be the last element in $cycles$, a replenishment cycle over $\{i, \ldots, j\}$;
    $S^i \leftarrow$ cycle opening inventory level of e, linearly dependent on $\tilde{I}_j$;
    $holdingCost \leftarrow$ cycle holding cost of e with opening inventory level S^i (27);
    $cost \leftarrow cost + holdingCost$;
    $Inf(C) \leftarrow cost$;
end

4.7 Computing the Objective Function

In order to compute the expected total cost for a given replenishment plan, or a lower bound for such a cost associated with a given partial assignment for replenishment decisions δ, we look again for the first M consecutive replenishment cycles identified by the current partial assignment for decision variables δ. Therefore we will assume that $R^1, \ldots, R^M$ are known (Algorithm 1, line 8) and we will follow a reasoning similar to the one developed to satisfy our chance-constraints.

Procedure `setBufferForCycle`$(cycles, d_1, \ldots, d_N, \alpha)$

input: $cycles, d_1, \ldots, d_N, \alpha$

begin
  let R be the last element in $cycles$, a replenishment cycle over $\{i, \ldots, j\}$;
  $S \leftarrow \lceil G^{-1}_{d_i+\ldots+d_j}(\alpha) \rceil$;
  decrease S to the min value that satisfies (25), with former cycles as listed in $cycles$;
  $\tilde{I}_j \leftarrow x - \tilde{d}_i - \ldots - \tilde{d}_j$;
end

Function `listCycles`$(\delta_1, \ldots, \delta_N, \tilde{I}_1 \ldots, \tilde{I}_N, N)$

input : $\delta_1, \ldots, \delta_N, N$
output: $cycles$

begin
  $cycles \leftarrow \{\}$;
  $lastCycle \leftarrow null$;
  $pointer \leftarrow 1$;
  for *each* δ_i, $i = 2, \ldots, N$ **do**
    if δ_i *is not assigned* **then**
      return $cycles$;
    else if $lastCycle \neq null$ **then**
      let $\{i, \ldots, j\}$ be the span covered by $lastCycle$;
      if *no variable* $\tilde{I}_i, \ldots, \tilde{I}_j$ *is assigned* **then**
        return $cycles$;
    if δ_i *is assigned to* 1 **then**
      $lastCycle \leftarrow$ a replenishment cycle over $\{pointer, ..., i-1\}$;
      add $lastCycle$ to $cycles$;
      $pointer \leftarrow i$;
  $lastCycle \leftarrow$ a replenishment cycle over $\{pointer, ..., N\}$;
  add $lastCycle$ to $cycles$;
  return $cycles$;
end

The expected holding cost for replenishment cycle R^j, $j \in \{1, \ldots, M\}$, is affected only by former replenishment cycles $\{R^i, \ldots, R^{j-1}\}$, where $i = \min \{v \in \{1, \ldots, j\} | (S^v \geq S^1) \wedge \ldots \wedge (S^v \geq S^{v-1})\}$. If $i = j$ no former replenishment cycle affects R^j. Now since we know the distribution of the demand in replenishment cycles $\{R^i, \ldots, R^j\}$ and since we assume that former opening-inventory-levels $\{S^i, \ldots, S^{j-1}\}$ have been already set, it is easy to recursively compute the expected holding cost for replenishment cycle R^j by using a *scenario based approach*.

The expected holding cost (HC) for R^j given that $\{R^i, \ldots, R^{j-1}\}$ are the earlier periods affecting R^j can be computed as

$$E\{HC_{R^j}\} = P_j(S^j) \cdot h(R^j, \tilde{I}_j) + \sum_{k=1}^{S^i - S^j} \left(P_j(S^j, k) \cdot h(R^j, \tilde{I}_j + i) \right). \tag{27}$$

Also in this case, since negative demands are not considered in the summation, event probabilities must be normalized accordingly using the term given in (26) as shown before.

A valid lower bound (Algorithm 1, line 24) for the expected total cost of a given partial assignment involving decision variables δ—tight when the assignment is complete (Algorithm 1, line 17)—can be computed by considering a fixed ordering cost for each replenishment cycle R^i identified by the assignment (Algorithm 1, line 6), plus the expected holding cost for the first M consecutive replenishment cycles $R^1, \ldots, R^M$ computed as explained above (Algorithm 1, lines 14 and 22).

4.8 Cost-Based Filtering

In order to improve the search process we employed a cost-based filtering method similar to the one proposed in [31]. We will not describe in detail the whole method. We will rather try to give a high level description of it. The reader may refer to [31] for further details.

Firstly we recall that, in Tarim and Kingsman's model [29], upper bounds for decision variables $\tilde{I}_i$, $i = \{1, \ldots, N\}$ can be computed by considering a single replenishment cycle covering the whole planning horizon. The buffer stock required to guarantee the required service level is $b(1, N)$, as defined in (13). Since $b(i, j)$ is an increasing function [30], it directly follows that the maximum value for the domain of $\tilde{I}_N$ is obviously $b(1, N)$ and that for every other decision variable $\tilde{I}_i$, $i = \{1, \ldots, N-1\}$ the maximum value in the domain is $b(1, N) + \sum_{k=i+1}^{N} \tilde{d}_k$. These bounds are still valid in our model. In fact the effect of excess stocks from former periods may only decrease a buffer stock needed to provide a given service level.

A lower bound for the cost of an optimal policy associated with a given partial assignment can be computed as shown in [31]. In this work the authors solve in polynomial time, by using a shortest path algorithm, a relaxation of the original problem where inventory conservation constraints between subsequent replenishment cycles are relaxed. This means that negative order quantities are allowed in this relaxed model. The bound is dynamically computed during the search process and it takes into account partial assignments for both decision variables δ_t and inventory levels $\tilde{I}_t$, by respectively forbidding or forcing stated nodes in the optimal path to reflect assignments for δ_t variables, and by modifying costs in the connection matrix to reflect assignments for $\tilde{I}_t$ variables.

A similar approach can be adopted in our case by noticing that Tarim and Kingsman's approach underestimates holding cost in each period. Firstly because it considers the contribution of negative inventory levels on the holding cost. Secondly because it does not consider the effect of excess stocks from former periods not only in the service level computation, but also in the cost computation. This means that Tarim and Kingsman's model always computes a cost that is less than or equal to the actual cost associated with a given policy. On the other hand, as seen, such a model overestimates buffer stocks.

In our cost-based filtering approach we relax not only the inventory conservation constraints, as in [31], but also the constraints that force buffer stocks at the end of each replenishment cycle. Therefore we simply solve a deterministic production planning problem under fixed ordering cost and linear holding cost. The same algorithm proposed in [31] can be employed to efficiently solve this problem. Since we do not take into account buffer stocks, and from the former considerations on the cost structure, this relaxed Tarim and Kingsman model provides a lower bound for the cost provided by our exact model. Also in our cost-based filtering approach this bound is dynamically computed during the search process and it takes into account partial assignments for both decision variables δ_t and inventory levels $\tilde{I}_t$ as discussed above.

5 Comparison with Tarim and Kingsman's Approach

In this section we compare the results obtained by the approach presented in [31] with the exact solutions provided by the new model.

The following assumptions are valid for the rest of this section. We assume that the demand in each period is normally distributed about the forecast value with the same coefficient of variation τ. Thus the standard deviation of demand in period t is $\sigma_t = \tau \cdot \tilde{d}_t$. In all cases, initial inventory levels, delivery lead-times and salvage values are set to zero.

All experiments here presented were performed on an Intel(R) Centrino(TM) CPU 1.50GHz with 500Mb RAM. The solver used for our test is Choco [19], an open-source solver developed in Java.

Firstly we consider a decreasing demand pattern over a 5-period planning horizon. The planning horizon considered is short since this demand pattern is particularly hard to treat.

The forecasts for the demand in each period are given in Table 1. As input parameters we considered $a \in \{1, 100, 200\}$, $\tau \in \{0.15, 0.25\}$ and $\alpha \in \{0.95, 0.75\}$. The holding cost h is fixed and equal to 1 for all the instances, since replenishment decisions are affected only by the ratio between ordering cost and holding cost. In Table 2 experimental results are presented. For each instance considered "Exact $E\{TC\}$" is the expected total cost of the optimal solution (i.e. set of policy parameters: replenishment cycle lengths and order-up-to-levels) obtained using the complete

Table 1 Expected values for a decreasing demand pattern

	Period	1	2	3	4	5
Decreasing	$\tilde{d}_t$	400	130	150	60	35

Table 2 Decreasing demand pattern

	Parameters			Total Cost						
	a	τ	α	T&K				Exact		
				$E\{TC\}$	$\widehat{E}\{TC\}$	gap(%)	sec	$E\{TC\}$	gap(%)	sec
1	1	0.25	0.95	324	370	12.4	1	358	3.35	469
2	100	0.25	0.95	773	814	5.04	1	799	1.88	254
3	200	0.25	0.95	1, 152	1, 189	3.11	1	1, 176	1.11	165
4	1	0.15	0.95	197	205	3.90	1	200	2.50	372
5	100	0.15	0.95	637	644	1.09	1	640	0.63	249
6	200	0.15	0.95	984	990	0.61	1	985	0.51	30
7	1	0.25	0.75	135	178	24.1	1	172	3.49	219
8	100	0.25	0.75	573	613	6.53	1	607	0.99	161
9	200	0.25	0.75	886	910	2.64	1	907	0.33	22
10	1	0.15	0.75	83	101	17.8	1	100	1.00	282
11	100	0.15	0.75	517	535	3.36	1	534	0.19	181
12	200	0.15	0.75	797	810	1.60	1	809	0.12	8

Columns "$E\{TC\}$" are the expected total cost computed by Tarim and Kingsman's approximate approach (T&K) and by our exact approach (Exact). In order to compute T&K $E\{TC\}$ we employed the efficient CP approach proposed in [31]. In columns "sec" we report, in seconds, the time performance for each model. Since T&K provides an approximate expected total cost, in column "$\widehat{E}\{TC\}$" we report the actual expected total cost of such a solution, which is computed by simulating demands according to the given distribution in each period and by observing the realized total cost over 10,000 runs. The two columns "gap" for T&K and Exact report respectively: the difference between T&K $E\{TC\}$ and T&K $\widehat{E}\{TC\}$, in percentage on T&K $E\{TC\}$, and the difference between T&K $\widehat{E}\{TC\}$ and Exact $E\{TC\}$ in percentage on Exact $E\{TC\}$. Holding cost h is set to 1 for every instance

approach we presented. "T&K $E\{TC\}$" is the approximate expected total cost of the solution obtained by using the model proposed in [31], which adopts Tarim and Kingsman's approach. "T&K $\widehat{E}\{TC\}$" is the actual expected total cost of the solution obtained using the model proposed in [31]. This actual expected total cost has been computed by simulation. Notice that for some parameter configurations the solution obtained with the approach in [30] differs from the optimal one, while for other cases the approximate approach produces a solution close to the optimal one. The reasons are different depending on the particular parameter configuration.

Instance (1) has a low ordering cost a, therefore we expect to order frequently. The expected total holding cost and the buffer stock levels required to provide service level α are affected by the negative trend of the demand and by excess stocks carried from former replenishment cycle as a consequence of this trend (Fig. 6). Since the model in [31] does not take into account these effects the expected total cost of the optimal solution it provides (T&K $\widehat{E}\{TC\}$) differs from the actual optimum (Exact $E\{TC\}$).

Instances (10), (11) and (12) have a low service level α and coefficient of variation τ. In this case the policy parameters computed by the approach in [31] are optimal, in fact T&K $\widehat{E}\{TC\}$ is close to Exact $E\{TC\}$. The effect of excess stocks is so low that it can actually be ignored, but the approximate expected total cost computed by the approach in [31] (T&K $E\{TC\}$) differs from the exact one (T&K $\widehat{E}\{TC\}$) by respectively 17.8%, 3.36% and 1.60%, since negative inventory levels affect the expected total cost of the policy. This follows from the fact that we require a low

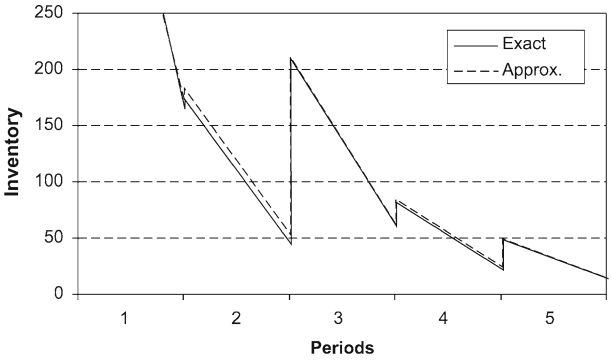

Fig. 6 Comparison between inventory levels computed by the exact and the approximate approach

service level and we keep low buffer stock levels, therefore the probability of ending up with negative inventory levels becomes high and the effect of negative inventory levels on the expected holding cost increases as the length of the replenishment cycles decreases.

It should be noted that the computational effort required by our exact approach to compute policy parameters is directly affected by the number of replenishment cycles in our plan. This is the reason why we observe higher run times when the ratio between ordering cost and holding cost is low. This is true in general also for the instances that will be considered below.

We will now consider three other demand patterns that typically arise in practice. These patterns were originally proposed by Berry in [5] and they were also adopted for the experiments in [29]. The patterns are presented in Table 3. We did not consider a constant demand pattern, which is instead included in Berry's test bed, since it is obvious that for this pattern the solutions provided by our approach would not differ from the ones provided by Tarim's and Kingsman approach. In these cases as input parameters we considered $a \in \{1, 50, 100\}$, $\tau \in \{0.2, 0.3\}$ and $\alpha \in \{0.95, 0.75\}$. In Table 4 experimental results for these three further demand patterns are presented. Similar considerations to those just introduced indicate why also for these demand patterns in some cases the results provided by our exact approach may differ substantially from those obtained with the approximate one. Typically such a difference is due to the combined effect of excess stocks and/or negative inventory levels as already discussed.

From our experiments it is clear that the approximate expected total cost computed by Tarim and Kingsman's model (T&K $E\{TC\}$) may substantially underestimate the exact expected total cost (T&K $\widehat{E}\{TC\}$) associated with a given solution, which can be easily computed by simulation or by using our exact model.

Table 3 Expected values for seasonal, life cycle and erratic demand patterns

	Period	1	2	3	4	5	6	7	8
Seasonal	$\tilde{d}_t$	50	75	90	75	50	25	10	25
Life cycle	$\tilde{d}_t$	20	25	30	35	40	25	20	10
Erratic	$\tilde{d}_t$	50	30	70	15	60	10	30	15

Table 4 Experimental results for seasonal (13, . . . , 24), life cycle (25, . . . , 36) and erratic (37, . . . , 48) demand patterns

	Parameters			Total Cost						
	a	τ	α	T&K				Exact		
				$E\{TC\}$	$\widehat{E}\{TC\}$	gap(%)	sec	$E\{TC\}$	gap(%)	sec
13	1	0.3	0.95	205	213	3.76	1	207	2.90	2, 774
14	50	0.3	0.95	566	570	0.70	1	564	1.06	478
15	100	0.3	0.95	858	864	0.69	1	859	0.58	104
16	1	0.2	0.95	139	140	0.71	1	139	0.72	1, 412
17	50	0.2	0.95	498	499	0.20	1	498	0.20	180
18	100	0.2	0.95	771	772	0.13	1	766	0.78	66
19	1	0.3	0.75	88	108	18.5	1	106	1.89	908
20	50	0.3	0.75	440	458	3.93	1	458	0.00	165
21	100	0.3	0.75	696	710	1.97	1	709	0.14	56
22	1	0.2	0.75	61	73	16.4	1	72	1.39	603
23	50	0.2	0.75	411	422	2.61	1	420	0.48	109
24	100	0.2	0.75	658	666	1.20	1	665	0.15	51
25	1	0.3	0.95	109	110	0.91	1	110	0.00	48
26	50	0.3	0.95	441	443	0.45	1	438	1.14	8
27	100	0.3	0.95	634	634	0.00	1	630	0.63	4
28	1	0.2	0.95	76	77	1.30	1	77	0.00	34
29	50	0.2	0.95	393	393	0.00	1	392	0.26	6
30	100	0.2	0.95	574	574	0.00	1	570	0.70	4
31	1	0.3	0.75	49	58	15.5	1	56	3.57	30
32	50	0.3	0.75	355	362	1.93	1	357	1.40	6
33	100	0.3	0.75	529	535	1.12	1	531	0.75	4
34	1	0.2	0.75	35	41	14.6	1	40	2.50	27
35	50	0.2	0.75	333	338	1.48	1	334	1.20	6
36	100	0.2	0.75	503	507	0.79	1	503	0.80	4
37	1	0.3	0.95	175	195	10.2	1	188	3.72	554
38	50	0.3	0.95	492	494	0.40	1	489	1.02	33
39	100	0.3	0.95	692	692	0.00	1	689	0.44	14
40	1	0.2	0.95	110	122	9.84	1	119	2.52	381
41	50	0.2	0.95	418	418	0.00	1	417	0.24	25
42	100	0.2	0.95	618	619	0.16	1	617	0.32	10
43	1	0.3	0.75	64	90	28.8	1	85	5.88	277
44	50	0.3	0.75	360	370	2.70	1	369	0.27	18
45	100	0.3	0.75	560	570	1.75	1	569	0.18	9
46	1	0.2	0.75	45	59	23.7	1	56	5.36	225
47	50	0.2	0.75	332	339	2.06	1	339	0.00	19
48	100	0.2	0.75	532	539	1.30	1	536	0.56	8

This is particularly evident in the erratic demand case, where for instances 43 and 46 the approximate expected total cost predicted by Tarim and Kingsman's model (T&K $E\{TC\}$) is respectively 28.8 and 23.7% less costly than the exact expected total cost associated with the policy parameter configuration in the respective solution (T&K $\widehat{E}\{TC\}$). Although Tarim and Kingsman's model underestimates cost—T&K $E\{TC\}$ is on average 5.26% lower than T&K $\widehat{E}\{TC\}$—over the whole test bed the average difference between T&K $\widehat{E}\{TC\}$ and Exact $E\{TC\}$ is only 1.25%.

This means that the approximate approach in [29] actually computes near-optimal parameters for (R^n,S^n) policy, reorder points and the respective order-up-to-levels, regardless of the underestimated cost. Nevertheless for some instances, i.e. (29), (30), (31) etc., T&K $E\{TC\}$ is equal to T&K $\widehat{E}\{TC\}$, which means that for these instances the assumptions adopted by Tarim and Kingsman are valid. In summary these results suggest that Tarim and Kingsman's model can actually compute near-optimal policy parameters, although the approximate expected total cost predicted can often differ significantly from the actual expected total cost associated with these reorder points and respective order-up-to-levels.

As we may notice from the run-times reported in columns "sec", the approach proposed in [31] always outperforms our exact method and runs efficiently for every instance considered. Further results presented in [31] suggest that such an approach can efficiently handle large scale instances. Since our results suggest that the exact solution in the average case differs only slightly from the one provided by Tarim and Kingsman's approximate approach, when efficiency is an issue, their approach remains a valid alternative to our exact model.

6 Conclusions

We identified two sources of approximation in Tarim and Kingsman's model for computing (R^n,S^n) policy parameters under service level constraint. We proposed an exact *stochastic constraint programming* approach based on a novel concept—*global chance-constraints*—which extends the original stochastic constraint programming framework proposed by Walsh. We described a dedicated global chance-constraint that computes optimal inventory levels to meet the required service level and the expect total cost associated with them. We analyzed the accuracy of the approximate solutions provided by the model developed by Tarim and Kingsman over four different demand patterns and over several different input parameter configurations. We also provided insights into for which kind of instances the assumptions adopted by Tarim and Kingsman may affect the quality of the solution provided by their model. Our results suggest that their modeling strategy is a good trade-off between quality of the solution and efficiency of the search process.

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. Balafoutis, T., & Stergiou, K. (2006). Algorithms for stochastic csps. In *Proceedings of the 12th international conference on the principles and practice of constraint programming. Lecture notes in computer science* (No. 4204, pp. 44–58). Springer Verlag.
4. Bellman, R. E. (2003). *Dynamic programming*. Dover Publications, Incorporated.
5. Berry, W. L. (1972). Lot sizing procedures for requirements planning systems: A framework for analysis. *Production and Inventory Management Journal, 13*, 19–34.
6. Birge, J. R., & Louveaux, F. (1997). *Introduction to stochastic programming*. New York: Springer.
7. Bookbinder, J. H., & Tan, J. Y. (1988). Strategies for the probabilistic lot-sizing problem with service-level constraints. *Management Science, 34*, 1096–1108.
8. Brailsford, S. C., Potts, C. N., & Smith, B. M. (1999). Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research, 119*, 557–581.

9. Charnes, A., & Cooper, W. W. (1959). Chance-constrainted programming. *Management Science, 6*(1), 73–79.
10. Davis, T. (1993). *Effective supply chain management*. Sloan Management Review.
11. de Kok, A. G. (1991). *Basics of inventory management: Part 2 the* (*R,S*)*-model*. *Research memorandum, FEW 521, 1991*. Tilburg, The Netherlands: Department of Economics, Tilburg University.
12. de Kok, T., & Inderfurth, K. (1997). Nervousness in inventory management: Comparison of basic control rules. *European Journal of Operational Research, 103*, 55–82.
13. Devore, J. L. (1995). *Probability and statistics for engineering and the sciences* (4th ed.). Duxbury Press
14. Florian, M., Lenstra, J. K., & Rinooy Kan, A. H. G. (1980). Deterministic production planning: Algorithms and complexity. *Management Science, 26*(7), 669–679.
15. Graves, S. C. (1999). A single-item inventory model for a non-stationary demand process. *Manufacturing & Service Operations Management, 1*, 50–61.
16. Heisig, G. (2002). *Planning stability in material requirements planning systems*. New York: Springer.
17. Janssen, F., & de Kok, T. (1999). A two-supplier inventory model. *International Journal of Production Economics, 59*, 395–403.
18. Jeffreys, H. (1961). *Theory of probability*. Oxford, UK: Clarendon Press.
19. Laburthe, F., & the OCRE project team. (1994). Choco: Implementing a cp kernel. Technical report. France: Bouygues e-Lab.
20. Lustig, I. J., & Puget, J. F. (2001). Program does not equal program: Constraint programming and its relationship to mathematical programming. *Interfaces, 31*, 29–53.
21. Regin, J.-C. (1994). A filtering algorithm for constraints of difference in csps. In *Proceedings of the national conference on artificial intelligence* (*AAAI-94*) (pp. 362–367). Seattle, WA, USA.
22. Regin, J.-C. (2003). Global constraints and filtering algorithms. In M. Milano (Ed.), *Constraints and integer programming combined*. Kluwer.
23. Rossi, F., Petrie, C., & Dhar, V. (1990). On the equivalence of constraint satisfaction problems. In *Proceedings of the 9th ECAI. European conference on artificial intelligence* (pp. 550–556). Stockholm, Sweden: Pitman Publishing.
24. Scarf, H. (1960). The optimality of (s,S) policies in dynamic inventory problem. In K. J. Arrow, S. Karlin, & P. Suppes (Eds.), *Mathematical methods in social sciences*. Stanford Unversity Press.
25. Silver, E. A. (1978). Inventory control under a probabilistic time-varying demand pattern. *AIIE Transactions, 10*, 371–379.
26. Silver, E. A., Pyke, D. F., & Peterson, R. (1998). *Inventory management and production planning and scheduling*. New York: Wiley.
27. Smits, S. R., Wagner, M., & de Kok, T. G. (2004). Determination of an order-up-to policy in the stochastic economic lot scheduling model. *International Journal of Production Economics, 90*, 377–389.
28. Tang, C. S. (2006). Perpectives in supply chain risk management. *International Journal of Production Economics, 103*, 451–488.
29. 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.
30. Tarim, S. A., & Smith, B. (2008). Constraint programming for computing non-stationary (*R,S*) Inventory Policies. *European Journal of Operational Research* (to appear).
31. 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. Lecture Notes in Artificial Intelligence* (No. 4651, pp. 169–183). Springer.
32. Tarim, S. A., Hnich, B., Rossi, R., & Prestwich, S. (2008). Cost-based filtering techniques for stochastic inventory control under service level constraints. *Constraints* (to appear).
33. Tarim, S. A., Manandhar, S., & Walsh, T. (2006). Stochastic constraint programming: A scenario-based approach. *Constraints, 11*(1), 53–80.
34. Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management Science, 5*, 89–96.
35. Walsh, T. (2002). Stochastic constraint programming. In *Proceedings of the 15th ECAI. European conference on artificial intelligence*. IOS Press.