A State Space Augmentation Algorithm for the Replenishment Cycle Inventory Policy

Roberto Rossi [a,*] S. Armagan Tarim [b] Brahim Hnich [c]
Steven Prestwich [d]

[a] *Logistics, Decision and Information Sciences, Wageningen UR, the Netherlands*
[b] *Department of Management, Hacettepe University, Turkey*[1]
[c] *Faculty of Computer Science, Izmir University of Economics, Izmir, Turkey*[1]
[d] *Cork Constraint Computation Centre, University College, Cork, Ireland*

Abstract

In this work we propose an efficient Dynamic Programming approach for computing Replenishment Cycle policy parameters under non-stationary stochastic demand and service level constraints. The Replenishment Cycle policy is a popular inventory control policy typically employed for dampening planning instability. The approach proposed in this work achieves a significant computational efficiency and it can solve any relevant size instance in trivial time. Our method exploits the well known concept of State Space Relaxation. A filtering procedure and an augmenting procedure for the state space graph are proposed. Starting from a relaxed state space graph our method tries to remove provably suboptimal arcs and states (filtering) and then it tries to efficiently build up (augmenting) a reduced state space graph representing the original problem. Our experimental results show that the filtering procedure and the augmenting procedure often generate a small filtered state space graph, which can be easily processed using Dynamic Programming in order to produce a solution for the original problem.

Key words: Inventory Control; Non-stationary Stochastic Demand; Replenishment Cycle Policy; Dynamic Programming; State Space Relaxation; State Space Filtering; State Space Augmentation

* **Corresponding author**. LDI, Wageningen UR, Hollandseweg 1, 6706 KN, Wageningen, The Netherlands. Tel. +31 (0) 317 482321, Fax. +31 (0)317 485646.
Email addresses: roberto.rossi@wur.nl (Roberto Rossi), armtar@yahoo.com (S. Armagan Tarim), brahim.hnich@ieu.edu.tr (Brahim Hnich), s.prestwich@4c.ucc.ie (Steven Prestwich).

[1] **Acknowledgments:** B. Hnich and A. Tarim are supported by the Scientific and

Technological Research Council of Turkey under Grant No. SOBAG-108K027. A. Tarim is supported by Hacettepe University-BAB.

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 and service level constraints. Such a problem has been widely studied because of its key role in practice.

Different inventory control policies can be adopted for the above mentioned problem. For a discussion of inventory control policies see [18]. One of the possible policies that can be adopted is the replenishment cycle policy, (R, S). A detailed discussion on the characteristics of (R,S) can be found in [7]. In this policy an order is placed every R periods to raise the inventory level to the order-up-to-level S. This provides an effective means of dampening planning instability (deviations in planned orders, also known as *nervousness* [8,11]) and coping with demand uncertainty. As pointed out by Silver et al. ([18], 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. 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 [19].

Under the non-stationary demand assumption the replenishment cycle policy takes the form (R^n, S^n) where R^n denotes the length of the nth replenishment cycle and S^n the respective order-up-to-level. In this policy, the actual order quantity for replenishment cycle n is determined after the demand in previous periods has been observed. The order quantity 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 $\{R^n | n = 1, \ldots, M\}$ and $\{S^n | n = \{1, \ldots, M\}$, where M denotes the number of replenishment cycles scheduled over a finite planning horizon of N periods.

The problem of populating these sets has been solved to optimality only recently, due to the complexity involved in the modeling of uncertainty and of the policy-of-response. As Silver points out, computing replenishment cycle policy parameters under non-stationary stochastic demand is a computationally hard task [17]. Early works in this area adopted heuristic strategies such as those proposed by Silver [17], Askin [1] and Bookbinder & Tan [5]. Under some mild assumptions, the first complete solution method for this problem was introduced by Tarim & Kingsman [22], who proposed a deterministic equivalent Mixed Integer Programming (MIP) formulation for computing (R^n, S^n)

policy parameters. Tempelmeier extended Tarim & Kingsman's MIP formulation in order to consider different service level measures [25], such as the "fill rate". Nevertheless, empirical results showed that Tarim & Kingsman's model is unable to solve large instances. Tarim & Smith [24] therefore introduced a more compact and efficient Constraint Programming formulation of the same problem that showed a significant computational improvement over the MIP formulation. The Constraint Programming formulation has been further enhanced by means of dedicated cost-based filtering algorithms developed by Tarim et al. in [21]. A Stochastic Constraint Programming [23] approach for computing optimal (R^n, S^n) policy parameters is proposed in [15]. In this work the authors drop the mild assumptions originally introduced by Tarim & Kingsman and compute optimal (R^n, S^n) policy parameters. Of course, there is a price to pay for dropping Tarim & Kingsman's assumptions, in fact this latter approach is less efficient than the one in [24]. Finally, Pujawan and Silver recently proposed a novel and effective heuristic approach [13].

In this paper, we build on Tarim & Kingsman's modeling assumptions and we develop a state-of-the-art algorithm for computing optimal (R^n,S^n) policy parameters. Two existing techniques — Dynamic Programming and State Space Relaxation — are combined in order to obtain an effective approach for computing (R^n,S^n) policy parameters. Dynamic Programming (DP) is an optimization procedure that solves optimization problems by decomposing them into a nested family of subproblems. DP is based on the *principle of optimality* [2,9] and it has been applied to solve a wide variety of combinatorial optimization problems, as well as optimal control problems. State Space Relaxation (SSR) considers the DP formulation of a combinatorial optimization problem, and modifies this formulation to obtain a different — and possibly more compact — DP formulation whose optimal solution is a lower bound for the original problem. Proposed by Christofides et al. in [6], SSR has been successfully applied to constrained variants of routing problems (see e.g. [12,10]). Roughly speaking, SSR maps the original State Space Graph to a new state space graph having a smaller number of vertices, and whose shortest path represents a lower bound for the cost of the shortest path in the original State Space Graph.

In this work, we enhance these known approaches with a novel strategy: we introduce a filtering procedure for the State Space Graph and an augmenting procedure that is able to build a reduced State Space Graph for the original problem starting from a filtered state space graph for the relaxed problem. The concept of State Space Augmentation [4] is known in the Operations Research literature. A dual approach to State Space Augmentation also exists and is known as Decremental SSR [14]. Nevertheless, the idea of filtering a relaxed state space graph is, to the best of our knowledge, a novel contribution. Our experimental results prove the effectiveness of such an approach for computing optimal (R^n,S^n) policy parameters.

The paper is structured as follows. In Section 2 we introduce the problem definition and the modeling assumptions adopted in this work. In Section 3 we describe a DP reformulation for Tarim and Kingsman's model. An SSR for this reformulation is presented in Section 4. A procedure for filtering the relaxed State Space Graph is presented in Section 5. An augmenting procedure for the relaxed State Space Graph is described in Section 6. An example that demonstrates the algorithm proposed is given in Section 7. Our computational experience and a comparison with the state-of-the-art approaches for computing Replenishment Cycle policy parameters are discussed in Section 8. In Section 9 we draw conclusions.

2 Problem definition and modeling assumptions

The single-location, single-product production/inventory control problem under non-stationary stochastic demand and service level constraints is formulated in this paper by using the following inputs and assumptions.

We consider a planning horizon of N periods and a demand d_t for each period $t \in \{1, \ldots, N\}$, which is a non-negative random variable with known probability density function and expected value $\tilde{d}_t$. We assume that the demand occurs instantaneously at the beginning of each time period. The demand is non-stationary, that is it can vary from period to period, demands in different periods are assumed to be independent. Demands occurring when the system is out of stock are assumed to be back-ordered and satisfied as soon as the next replenishment order arrives. The sell-back of excess stock is not allowed, if the actual stock exceeds the order-up-to-level for a given review, this excess stock is carried forward and it is not returned to the supply source. However, as in [5,22,24,25] such occurrences are regarded as rare events and accordingly the cost of carrying this excess stock and its effect on the service levels of subsequent periods are ignored.

A fixed delivery cost a is incurred for each order. A linear holding cost h is incurred for each unit of product carried in stock from one period to the next. Our aim is to find a replenishment plan that minimizes the expected total cost, which is composed of ordering costs and holding costs, over the N-period planning horizon, satisfying the service level constraints. As a service level constraint we require that, with a probability of at least a given value α, at the end of each period the net inventory will be non-negative. As pointed out in [25], since period demands are random, the net inventory may become negative. However, the number of stock-outs is restricted by the service level constraints enforced. While computing holding costs, we will assume, as in [5,22,24,25], that the service level is set large enough to ensure that the net inventory will be a good approximation of the inventory on hand.

3 A DP Formulation for the Deterministic Equivalent Problem

We hereby introduce a deterministic equivalent DP formulation for computing optimal (R^n,S^n) policy parameters.

Definition: A replenishment cycle, $T(i,j)$, is the time span between two consecutive orders/productions occurring in periods i and $j+1$, $j \geq i$.

Definition: The cycle buffer stock, $b(i,j)$, denotes the minimum expected buffer stock level required to satisfy the required non-stock-out probability during $T(i,j)$.

We define $b(i,j)$, $i = 1,\ldots,N$, $j = i,\ldots,N$, as

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

where $G_{d_i+d_{i+1}+\ldots+d_j}$ 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. It should be noted that it is possible to consider different service level measures — for instance the "fill rate" — simply by introducing a different definition for the cycle buffer stock (see also [25]).

Since N is the number of periods in our planning horizon, this will also be the number of steps in the system. A state s_k at step k represents a possible expected closing-inventory-level, $\tilde{I}_k$, at the end of period k. The decision x_k to be taken at step k is to place an order in such a period or not; if an order is placed, x_k also indicates how many subsequent periods this order should cover.

Let $X_k(s_{k-1})$ denote the set of possible feasible decisions x_k at period k, when the expected closing inventory level at period $k-1$ is s_{k-1}. This set may comprise: the decision of not placing an order ($x_k = 0$), the decision of covering 1 period with the order placed ($x_k = k$), the decision of covering 2 periods with the order placed ($x_k = k+1$), ..., and the decision of covering $N-k+1$ periods with the order placed ($x_k = N$). In other words, if $x_k = 0$, no order is placed in period k; if $k \leq x_k \leq N$, x_k schedules a replenishment cycle $T(k, x_k)$. However, one should note that the decision $x_k = 0$ is only allowed if

$$b(v,k) \leq s_{k-1} - \tilde{d}_k,$$

where $v = \max\{t | 1 \leq t \leq k, x_t > 0\}$. Intuitively, we can decide not to place an order at the beginning of period k if and only if we have sufficient stocks to guarantee the required service level at least for this period.

Given a pair $\langle s_k, x_k \rangle$ the cost function $p_k(s_k, x_k)$ is clearly given by the sum of the fixed ordering cost a, which is charged if x_k states that an order should be placed, and of the inventory holding cost at the end of the period, which is equal to the expected closing-inventory-level s_k, multiplied by the per-unit holding cost h. A per-unit purchase/production cost may also be considered, this will be briefly discussed in Section 6.

The state transition function, $s_k = t_k(s_{k-1}, x_k)$, is as follows

$$s_k = \begin{cases} s_{k-1} - \tilde{d}_k & \text{if } x_k = 0, \\ \max(s_{k-1} - \tilde{d}_k, b(k, x_k) + \sum_{i=k+1}^{x_k} \tilde{d}_i) & \text{if } k \leq x_k \leq N. \end{cases} \tag{2}$$

S_k, the set of feasible expected closing-inventory-levels at the end of period k, is obtained recursively from the state transition functions $t_1, t_2, \ldots, t_k$, by assuming $s_0 = 0$ and, therefore, that an order should be always placed at period 1 in order to cover one or more following periods. In other words, $X_1(s_0)$ does not include the option of not placing an order.

The objective function is

$$z = \min \left\{ \sum_{k=1}^{N} p_k(s_k, x_k) \right\}. \tag{3}$$

To determine the value of z, DP solves a set of problems $i = 1, \ldots, N$, each corresponding to a system composed by i steps and characterized by the state s_i at the end of step i. The recursive formulation of the cost function at step i is:

$$f_i(s_i) = \min_{x_i \in X_i(s_{i-1})} \{ f_{i-1}(s_{i-1}) + p_i(s_i, x_i) \}, \tag{4}$$

where $s_i = t_i(s_{i-1}, x_i)$. In addition, we have the following boundary condition:

$$f_1(s_1) = \min_{x_1 \in X_1(s_0)} \{ p_1(s_1, x_1) \}, \tag{5}$$

where $s_1 = t_1(s_0, x_1)$.

Clearly, a mere recursive approach would immediately generate a very large State Space Graph that would certainly be unmanageable. For this reason, in the following sections we will propose an effective strategy for limiting the size of the State Space Graph.

4 A State Space Relaxation for the Deterministic Equivalent Problem

Intuitively, the first way of keeping the State Space Graph compact consists in employing a relaxation that clusters states together. More specifically, in order to do so we will employ a relaxation proposed by Tarim in [20].

The core observation in Tarim's relaxation lies in the fact that, if we relax the constraint which enforces non-negative order quantities — i.e. we give the opportunity to sell back items in excess to the supplier at the beginning of a given replenishment cycle — then the model proposed can be reduced to a Shortest Path Problem on a state space graph having a number of nodes and arcs polynomial in the number N of periods.

In this relaxation, since the inventory conservation constraint is relaxed between replenishment cycles, each replenishment cycle can be treated independently and its expected total cost can be computed *a priori*. In fact, given a replenishment cycle $T(i,j)$, we recall that $b(i,j)$, as defined above, denotes the minimum expected buffer stock level required to satisfy a given service level constraint during the replenishment cycle $T(i,j)$. It directly follows that $\tilde{I}_j = b(i,j)$. Furthermore for each period $t \in \{i,\ldots,j-1\}$ the expected closing-inventory-level is $\tilde{I}_t = b(i,j) + \sum_{k=t+1}^{j} \tilde{d}_k$. Since all the $\tilde{I}_t$ for $t \in \{i,\ldots,j\}$ are known it is easy to compute the expected total cost for $T(i,j)$, which is by definition the sum of the ordering cost and of the holding cost components, $a + h\sum_{t=i}^{j} \tilde{I}_t$.

We now have a set $\mathcal{S}$ of $N(N+1)/2$ possible different replenishment cycles and their respective costs. Our new problem is to find an optimal set $\mathcal{S}^* \subset \mathcal{S}$ of consecutive disjoint replenishment cycles that covers our planning horizon at the minimum cost.

We shall now show that the optimal solution to this relaxation is given by the shortest path in a *state space graph* from a given initial node to a final node (boundary condition) where each arc represents a replenishment cycle cost. If N is the number of periods in the planning horizon of the original problem, we introduce $N+1$ nodes. Since we assume that an order is always placed at period 1, we take node 1, which represents the beginning of the planning horizon, as the initial node. Node $N+1$ represents the end of the planning horizon.

Definition: The cycle cost, $c(i,j)$, denotes the expected cost of the optimal policy for $T(i,j)$. It can be expressed as

$$c(i,j) = a + h(j-i+1)b(i,j) + h\sum_{t=i}^{j}(t-i)\tilde{d}_t. \tag{6}$$

The cycle cost is the sum of two components. A fixed ordering cost a, that is charged at the beginning of the cycle when an order is placed, and a variable holding cost h_t charged at the end of each time period within the replenishment cycle and proportional to the amount of stock held in inventory.

For each possible replenishment cycle $T(i, j-1)$ such that $i, j \in \{1, \ldots, N+1\}$ and $i < j$, we introduce an arc (i, j) with associated cost $c(i, j-1)$ (Fig. 1). Since we are dealing with a one-way temporal feasibility problem [26], when

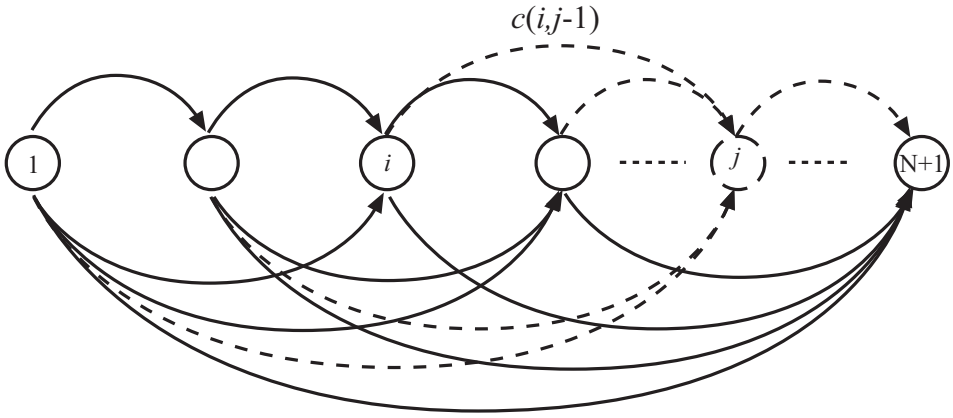

Fig. 1. Shortest path problem graph

$i \geq j$, we introduce no arc. As shown in [20], 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. A shortest path can be efficiently found by applying Dijkstra's algorithm that runs in $O(n^2)$ time, where n is the number of nodes in the graph. Details on efficient implementations of Dijkstra's algorithm can be found in [16].

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

5 A Filtering Procedure for the Relaxed State Space Graph

In the previous section we presented a known relaxation for the deterministic equivalent formulation of the (R^n,S^n) policy. In this relaxation we solve a shortest path problem over a given graph in order to find a lower bound for the cost of the optimal solution for the original problem.

We now aim to reduce *a priori* as much as possible the number of arcs in the graph we defined in the previous section. To do so we exploit a reduction procedure based on an upper bound for replenishment cycle lengths that was originally presented by Tarim and Smith in [24].

Definition: Cycle opening inventory level, $R(i,j)$, denotes the minimum opening inventory level in period i to meet demand until period $j+1$ and $R(i,j) = b(i,j) + \sum_{t=i}^{j} \tilde{d}_t$.

Let us assume now that period i is a replenishment period. It is not generally possible, prior to obtaining the optimal solution to an instance of the problem, to determine the length of the optimum replenishment cycle for a particular replenishment period; however, an upper bound on the length can be determined using Proposition 1.

Proposition 1 (Tarim and Smith [24]) *If* $\forall k \in \{i,...,j-1\}$, $\Big(c(i,k) + c(k+1,j) > c(i,j)\Big) \vee \Big(b(i,k) > R(k+1,j)\Big)$ *and* $\exists k \in \{i,...,j\}$ $\Big(c(i,k) + c(k+1,j+1) \leq c(i,j+1)\Big) \wedge \Big(b(i,k) \leq R(k+1,j+1)\Big)$ *then for period* i *the optimum length replenishment cycle is* $T(i,p)^*$ *where* $i \leq p \leq j$, *and* j *indicates an upper bound.*

Since we have an upper bound j for the length of an optimum replenishment cycle starting at period i, we can remove from our graph every arc (i,t), where $t > j+1$.

6 An Augmenting Procedure for the Relaxed State Space Graph

Once the shortest path problem on the graph constructed as shown above is solved, we can easily verify if every relaxed constraint is satisfied by the solution found, that is, if no expected negative replenishment quantity is scheduled in the optimal replenishment plan. In this case, the solution found is feasible and optimal for the original problem. If, on the other hand, the solution is not feasible for the original model and it schedules expected negative replenishment quantities, we can augment the graph with additional nodes and arcs in such a way that the shortest path on the augmented graph is guaranteed

to provide a feasible and optimal solution for the original problem. In what follows we shall show how to augment the graph and efficiently compute an optimal solution for the original problem.

Algorithm 1: `Augmenting Procedure`

input : a relaxed and filtered state space graph $RSG(S,T)$
output: an augmented state space graph $ASG(S',T')$

```
1  begin
2      i' = N + 1;
3      ASG(S',T') ← RSG(S,T);
4      for each node i = 1,...,N in S' do
5          for each arc (p,i) in T' do
6              let b* be the buffer stock associate to (p,i);
7              for each arc (i,j) in T' do
8                  if b* > R(i,j-1) then
9                      i' = i' + 1;
10                     create a new node i' in S';
11                     introduce arc (p,i') in T' with associated buffer stock b*;
12                     remove arc (p,i) from T';
13                     let t > i be the minimum index for which
                       b* ≤ R(i,t-1) ≤ R(i,t) ≤ ... ≤ R(i,N);
14                     introduce arc (i',t) T' with buffer stock b(i,t-1);
15                     for each arc (i,k), k = t+1,...,N+1 in T' do
16                         introduce arc (i',k) in T' with associated buffer stock
                           b(i,k-1);
17                     let t-1 > i be the maximum index for which
                       b* > ... ≥ R(i,t-2);
18                     introduce arc (i',t-1) in T'with associated buffer stock
                       b* - Σ_{k=i}^{t-1} d̃_k;
19 end
```

For convenience, instead of associating a cost $c(i,j-1)$ to each arc (i,j) in the graph, we will now associate the respective cycle buffer stock, $b(i,j-1)$, as defined above. From the definitions given, it is easy to see that, once this expected buffer stock level is fixed, also the cost $c(i,j-1)$ is uniquely defined.

Let $RSG(S,T)$ be a relaxed state space graph built according to the discussion in Section 4 and filtered according to the discussion in Section 5. Let S denote the set of nodes and T the set of arcs in the graph. The pseudo-code for the proposed augmenting procedure is presented in Algorithm 1. The procedure eventually generates an augmented state space graph $ASG(S',T')$, where S' is the set of nodes and T' is the set of arcs in the augmented graph.

Algorithm 1 initially creates a copy $ASG(S', T')$ of $RSG(S, T)$ (line 3). Then it considers each node in S' in order (line 4), starting from node 1 up to node N. Note that node $N+1$ has no outbound arcs, so we do not have to consider it. The process is repeated for each node i, therefore we will only describe the steps performed on a single node.

We consider every inbound arc at node i (line 5) and we operate in the following fashion. Given an inbound arc (p, i) with associated buffer stock b^* (line 6), for each outbound arc (i, j) in the graph (line 7) we check that $b^* \leq R(i, j-1)$. If this condition is satisfied for every outbound arc, then we preserve the inbound arc (p, i) at node i with the associated buffer stock b^* (Fig. 2). Otherwise, if $b^* > R(i, j-1)$ (line 8), for a subsequent pair of re-

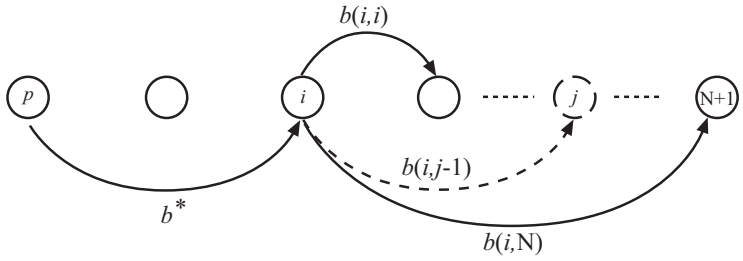

Fig. 2. Feasible node point

plenishment cycles a negative order quantity is scheduled. In order to resolve this infeasibility we perform the following transformation (lines 10...18). We introduce a new node i' in the graph. We remove arc (p, i) and we introduce a new arc (p, i') with associated buffer stock b^* (Fig. 3). Then we connect this new node the following way.

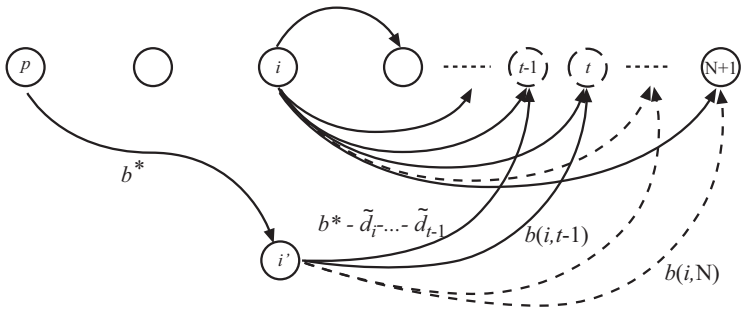

Fig. 3. Infeasible node point

Let $t > i$ be the minimum index for which $b^* \leq R(i, t-1) \leq R(i, t) \leq \ldots \leq R(i, N)$. We introduce arc (i', t) with buffer stock $b(i, t-1)$. Then, for each arc $(i, t+1), \ldots, (i, N+1)$ *in the graph*, we also introduce $(i', t+1), \ldots, (i', N+1)$ with buffer stock, respectively, $b(i, t), \ldots, b(i, N)$. It should be noted that some of the arcs $(i, t+1), \ldots, (i, N+1)$ may have been removed by the filtering described in Section 5.

Let $t-1 > i$ be the maximum index for which $b^* > \ldots \geq R(i, t-2)$. We introduce arc $(i', t-1)$ with buffer stock $b^* - \sum_{k=i}^{t-1} \tilde{d}_k$. Obviously arcs $(i', t-2), (i', t-3), \ldots$ are suboptimal and should not be introduced, since the inventory carried on from the previous replenishment cycle is enough to cover subsequent periods up to $t-1$.

Note that, when the process is iterated on subsequent nodes $i+1, \ldots, N$, the new inbound arcs that may have been introduced must also be considered among all the possible ones for a given node.

By starting from node 1 and by iterating this process for each node i, $1 \leq i \leq N$, we obtain an augmented graph. By construction the cost of the shortest path in this augmented graph is the optimal solution cost for our original problem since every possible negative order quantity scenario has been considered and replaced with the respective feasible possible courses of action. Nevertheless, as a consequence of the original filtering performed on the relaxed graph, the augmented graph will typically feature a very limited number of node and arcs. This will be shown in the following sections.

Before demonstrating our method on a simple numerical example, it is worth mentioning the following. Our model, for the sake of simplicity, assumes a zero unit purchase/production cost, also in line with the model in [24]. Nevertheless, the extension of our algorithm to the case of a non-zero unit production/purchasing cost is quite straightforward. In fact, as shown in [22] pp.113, the total unit variable cost can be reduced to a function of the expected closing-inventory-level of the very last period N. Therefore, considering such an effect in our algorithm is easy, since it only requires us to modify, in the graph connection matrix, the costs that appear in the rightmost column, which represents every possible replenishment cycle that ends in period N.

7 An Example

We shall consider here a simple example in detail, to show how in practice it is possible to apply the procedure described.

A single problem over a 5-period planning horizon is considered and the expected values for period demand are $[100, 125, 25, 40, 30]$. We assume an initial null inventory level and a normally distributed demand for every period with a coefficient of variation $\sigma_t/\tilde{d}_t = 0.3$ for each $t \in \{1, \ldots, N\}$, where σ_t denotes the standard deviation of the demand in period t. We consider an ordering cost value $a = 50$ and a holding cost $h = 1$ per unit per period. The non-stock-out probability in each period is set to $\alpha = 0.95$.

Firstly we build the connection matrix for the relaxed problem as described in Section 4. In Fig. 4 we show the connection matrix with the respective expected

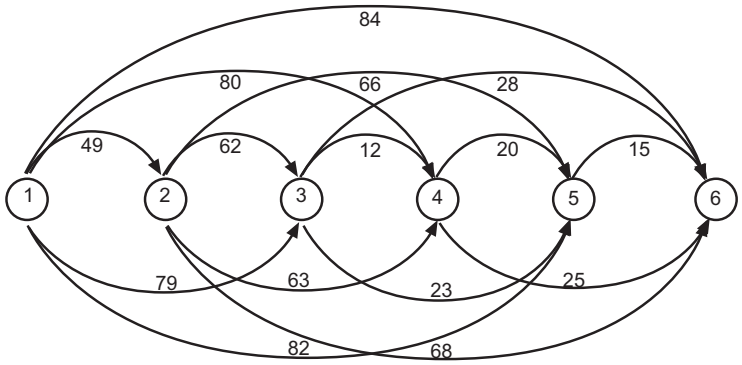

Fig. 4. Connection matrix with expected buffer stock levels

buffer stock level $b(i, j-1)$ associated with each arc (i, j).[1] In Fig. 5 instead

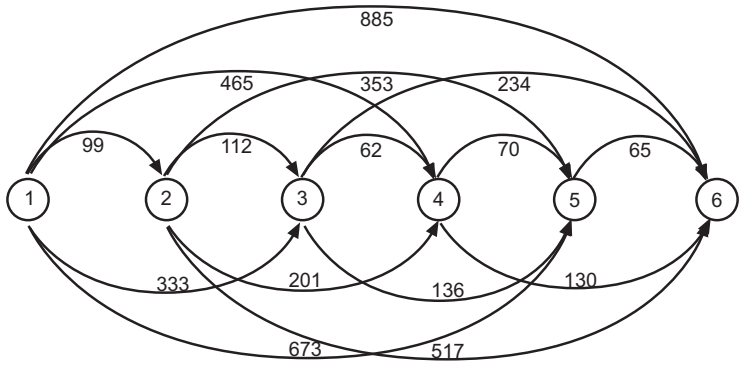

Fig. 5. Connection matrix with expected cycle costs

with each arc (i, j) we associate the respective expected cycle cost $c(i, j-1)$. It should be noted that the two representations are equivalent, since the expected cycle cost can be easily computed once the expected buffer stock level for a given cycle is fixed. In Fig. 6 the connection matrix is filtered according

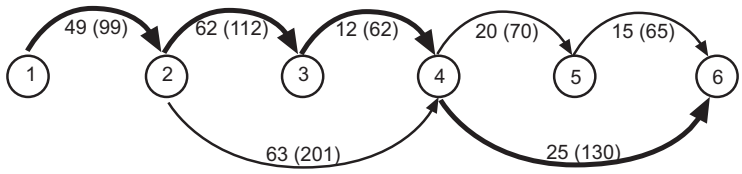

Fig. 6. Filtered connection matrix. Expected buffer stock levels and expected cycle costs (in parentheses) are shown for each arc. The shortest path is highlighted

to the procedure presented in Section 5. Expected buffer stock levels and expected cycle costs (in parentheses) are indicated for each arc that has not been removed by the filtering. The shortest path in this reduced network has a cost

[1] For clarity, in order to keep the graphical presentation as compact as possible, the expected buffer stock levels have been rounded to the nearest integer value.

of 403. The order periods and the order quantities are respectively $[1, 2, 3, 4]$ and $[149, 138, -25, 83]$. This assignment is infeasible for the non-relaxed problem since the expected order quantity in period 3 is -25, therefore its cost is a lower bound for the optimal solution cost of our original problem. According to the procedure described in Section 6 we augment the filtered graph and we obtain the new graph in Fig. 7. The shortest path in this augmented network

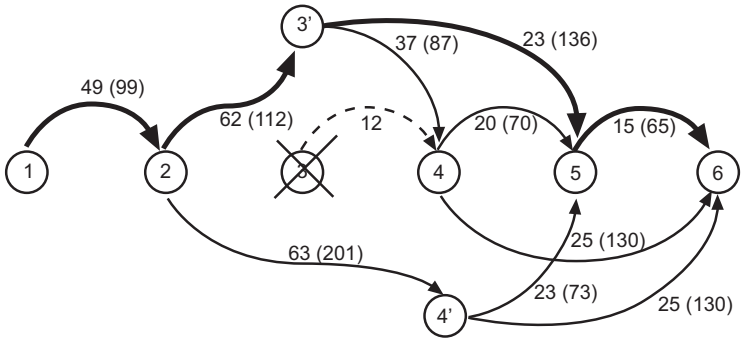

Fig. 7. Augmented connection matrix. Expected buffer stock levels and expected cycle costs (in parentheses) are shown for each arc. The shortest path is highlighted. Node 3 (and obviously arc $(3, 4)$) has been removed from the network since the augmenting procedure removed all its inbound arcs

has a cost of 412 and represents the optimal solution cost of our original problem. The replenishment periods in this optimal solution can be obtained from the indexes of the nodes in the shortest path. The respective order quantities can also be easily obtained from the expected buffer stock levels associated with each arc in the shortest path. The order periods and the order quantities are therefore respectively $[1, 2, 3, 5]$ and $[149, 138, 26, 22]$.

8 Experimental Results

We compared the results obtained with our approach with the results obtained with the state-of-the-art Constraint Programming (CP) approach in [21], based on the set of instances originally proposed in [3]. All the experiments presented in this section were performed on an Intel(R) Centrino(TM) CPU 1.50GHz with 500Mb RAM. As in [21], the demand in each period is assumed to be normally distributed and we also assume that its coefficient of variation remains sufficiently low (i.e. less or equal to 1/3) to ensure that negative demand values can be ignored. We recall that in [21] 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) three others are also considered:

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

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

For almost all these instances our DP approach is either better — in terms of run time — than the CP approach or equivalent, with some exceptions for the smallest instances. When the number of periods considered in the planning horizon grows, our DP approach clearly scales better than the CP approach. The maximum improvement observed reaches a factor of 24. Nevertheless, for this set of instances the CP approach remains competitive and achieves reasonable run times of a few seconds also for the largest instances.

In what follows, we aim to highlight the limits of the CP approach and we want to show that our DP approach remains very effective even for those instances for which the CP approach performs poorly. In order to do so, we consider the following set of instances (Test Set P5). The expected period demands, $\tilde{d}_t$, are generated as uniformly distributed random numbers in $[0, 100]$. Empirically, in fact, we observed that generating random sequences of demands rather than seasonal patterns or trends makes the problem harder to solve. Again we consider four different ordering cost values $a \in \{25, 50, 100, 200\}$ and two different $\sigma_t/\tilde{d}_t \in \{1/3, 1/6\}$. The planning horizon length takes the values $\{50, 75\}$. The holding cost used in these tests is $h = 1$ per unit per period. Again we consider two different service levels $\alpha = 0.95$ ($z_{\alpha=0.95} = 1.645$) and $\alpha = 0.99$ ($z_{\alpha=0.99} = 2.326$). Table 1 compares the CP and the DP approach for this new set of instances. In our test results, the heading "CP" refers to the state-of-the-art CP approach in [21], while "DP" refers to our novel DP approach. For the CP approach we report the number of nodes explored ("Nod") and the run time in seconds ("Sec"); for our DP approach we report the size of the state space graph generated ("Graph") and the run time in seconds ("Sec"). The size of the state space graph is described as a pair $\langle N; A \rangle$, where N is the number of nodes and A is the number of arcs. When a field is empty in the table, this means that the CP approach and the DP approach are equivalent, since for that particular instance the CP approach was able to prove optimality at the root node in polynomial time using the DP relaxation originally proposed in [20].

It is immediately clear that for low a/h ratios (that is for the lowest ordering costs considered), the CP approach has to explore a large search space and requires a long time to prove optimality, while our DP approach still generates small state space graphs and achieves fast runtimes. As the ratio a/h increases, the CP approach performs better and, for some instance, it is equivalent to our DP approach.

In the last set of instances considered (Test Set P6) we aim to show that our approach is effective even when the planning horizon is significantly longer, and that the computation is not affected by the magnitude of the demands considered. The planning horizon length now ranges up to 250 periods, in order to show that our approach scales well in the number of periods. The expected period demands $\tilde{d}_t$ are generated as uniformly distributed random numbers in $[0, 10000]$, in order to show that large values for the expected demands do not affect the scalability of our approach. Once more, we consider four different ordering cost values $a \in \{2500, 5000, 10000, 20000\}$ and two different $\sigma_t/\tilde{d}_t \in \{1/3, 1/6\}$. The planning horizon length takes the following values $\{75, 110, 145, 180, 215, 250\}$. The holding cost used in these tests is $h = 1$ per unit per period. Also in this case, we consider two different service levels $\alpha = 0.95$ ($z_{\alpha=0.95} = 1.645$) and $\alpha = 0.99$ ($z_{\alpha=0.99} = 2.326$). The computational results in Table 2 show that the graphs generated are still extremely compact and that the run times are mostly under one second even if a long planning horizon and large demands are considered.

9 Conclusions

We proposed a novel DP approach for computing (R^n,S^n) policy parameters. Our experimental results show that our approach, based on the described filtering algorithm for the State Space Graph and on the State Space Graph augmenting procedure, can solve instances over planning horizons comprising hundreds of periods. State Space Relaxation and State Space Augmentation are two known strategies in Operations Research, nevertheless, the idea of filtering a relaxed state space graph is, to the best of our knowledge, a novel contribution. As our computational experience shows, our DP reformulation performs significantly better than the original MIP approach proposed by Tarim and Kingsman and it also beats the state-of-the-art reformulations proposed by Tarim and Smith and by Tarim et al. . Furthermore our results are not affected by the magnitude of the demand considered in each period.

Acknowledgments: We wish to thank the anonymous reviewers who significantly contributed to improving the quality of this manuscript.

References

[1] R. G. Askin. A procedure for production lot sizing with probabilistic dynamic demand. *AIIE Transactions*, 13(2):132–137, 1981.

[2] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[3] W. L. Berry. Lot sizing procedures for requirements planning systems: A framework for analysis. *Production and Inventory Management Journal*, 13(2):19–34, 1972.

[4] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Oper. Res. Lett.*, 34(1):58–68, 2006.

[5] J. H. Bookbinder and J. Y. Tan. Strategies for the probabilistic lot-sizing problem with service-level constraints. *Management Science*, 34(9):1096–1108, 1988.

[6] N. Christofides, A. Mingozzi, and P. Toth. State space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11(2):145–164, 1981.

[7] A. G. de Kok. Basics of inventory management: part 2 The (R,S)-model. Research memorandum, FEW 521, 1991. Department of Economics, Tilburg University, Tilburg, The Netherlands.

[8] A. G. de Kok and K. Inderfurth. Nervousness in inventory management: Comparison of basic control rules. *European Journal of Operational Research*, 103(1):55–82, 1997.

[9] S. B. Dreyfus and A. M. Law. *The Art And Theory of Dynamic Programming*. Academic Press, New York, 1989.

[10] F. Focacci and M. Milano. Connections and integrations of dynamic programming and constraint programming. In *Proceedings of the International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems CP-AI-OR 2001*, 2001.

[11] G. Heisig. *Planning Stability in Material Requirements Planning Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[12] A. Mingozzi, L. Bianco, and S. Ricciardelli. Dynamic programming strategies for travelling salesan problem with time windows and precedence constraints. *Operations Research*, 45(3):365–377, 1997.

[13] I. N. Pujawan and E. A. Silver. Augmenting the lot sizing order quantity when demand is probabilistic. *European Journal of Operational Research*, 127(3):705–722, August 2008.

[14] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, 2008.

[15] R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich. A global chance-constraint for stochastic inventory systems under service level constraints. *Constraints*, 13(4):490–517, 2008.

[16] R. Sedgewick. *Algorithms (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.

[17] E. A. Silver. Inventory control under a probabilistic time-varying demand pattern. *AIIE Transactions*, 10(4):371–379, 1978.

[18] E. A. Silver, D. F. Pyke, and R. Peterson. *Inventory Management and Production Planning and Scheduling*. John-Wiley and Sons, New York, 1998.

[19] C. S. Tang. Perpectives in supply chain risk management. *International Journal of Production Economics*, 103(2):451–488, 2006.

[20] S. A. Tarim. *Dynamic Lotsizing Models for Stochastic Demand in Single and Multi-Echelon Inventory Systems*. PhD thesis, Lancaster University, 1996.

[21] S. A. Tarim, B. Hnich, R. Rossi, and S. Prestwich. Cost-based filtering techniques for stochastic inventory control under service level constraints. *Constraints*, 14(2):137–176, 2009.

[22] S. A. Tarim and B. G. Kingsman. The stochastic dynamic production/inventory lot-sizing problem with service-level constraints. *International Journal of Production Economics*, 88(1):105–119, 2004.

[23] S. A. Tarim, S. Manandhar, and T. Walsh. Stochastic constraint programming: A scenario-based approach. *Constraints*, 11(1):53–80, 2006.

[24] S. A. Tarim and B. Smith. Constraint Programming for Computing Non-Stationary (R,S) Inventory Policies. *European Journal of Operational Research*, 189(3):1004–1021, 2008.

[25] H. Tempelmeier. On the stochastic uncapacitated dynamic single-item lotsizing problem with service level constraints. *European Journal of Operational Research*, 181(1):184–194, 2007.

[26] H. M. Wagner and T. M. Whitin. Dynamic version of the economic lot size model. *Management Science*, 5(3):89–96, 1958.

		$\sigma_t/\tilde{d}_t = 1/3$							
		$\alpha = 0.95$				$\alpha = 0.99$			
		CP		DP		CP		DP	
a	N	Nod	Sec	Graph	Sec	Nod	Sec	Graph	Sec
25	50	857	45	$\langle 64; 76\rangle$	0.33	2474	170	$\langle 67; 82\rangle$	0.30
	75	41386	5400	$\langle 102; 125\rangle$	0.38	180000*	20000*	$\langle 106; 133\rangle$	0.37
50	50	441	17	$\langle 66; 84\rangle$	0.34	1242	170	$\langle 69; 88\rangle$	0.33
	75	23805	2400	$\langle 104; 133\rangle$	0.37	180000*	20000*	$\langle 108; 139\rangle$	0.37
100	50			$\langle 51; 87\rangle$	0.21	104	5	$\langle 73; 109\rangle$	0.34
	75			$\langle 76; 134\rangle$	0.24	329	30	$\langle 113; 167\rangle$	0.21
200	50			$\langle 51; 139\rangle$	0.23			$\langle 51; 131\rangle$	0.24
	75			$\langle 76; 212\rangle$	0.26			$\langle 76; 200\rangle$	0.28

		$\sigma_t/\tilde{d}_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$			
		CP		DP		CP		DP	
a	N	Nod	Sec	Graph	Sec	Nod	Sec	Graph	Sec
25	50	22	1	$\langle 58; 65\rangle$	0.34	325	17	$\langle 61; 70\rangle$	0.33
	75	245	35	$\langle 90; 103\rangle$	0.2	10118	970	$\langle 98; 116\rangle$	0.20
50	50			$\langle 51; 69\rangle$	0.20	70	3	$\langle 63; 80\rangle$	0.42
	75			$\langle 76; 106\rangle$	0.14	155	14	$\langle 100; 126\rangle$	0.37
100	50			$\langle 51; 103\rangle$	0.13			$\langle 51; 94\rangle$	0.12
	75			$\langle 76; 161\rangle$	0.26			$\langle 76; 145\rangle$	0.25
200	50			$\langle 51; 158\rangle$	0.22			$\langle 51; 184\rangle$	0.15
	75			$\langle 76; 242\rangle$	0.27			$\langle 76; 226\rangle$	0.26

Table 1

Test set P5. A figure marked with * means that the instance could not be solved in the given limit of 20000 seconds (5.55 hours)

		$\sigma_t/\tilde{d}_t = 1/3$				$\sigma_t/\tilde{d}_t = 1/6$			
		$\alpha = 0.95$		$\alpha = 0.99$		$\alpha = 0.95$		$\alpha = 0.99$	
a	N	Graph	Sec	Graph	Sec	Graph	Sec	Graph	Sec
2500	75	$\langle 102; 125\rangle$	0.44	$\langle 106; 133\rangle$	0.38	$\langle 90; 103\rangle$	0.25	$\langle 98; 116\rangle$	0.26
	110	$\langle 153; 194\rangle$	0.36	$\langle 160; 208\rangle$	0.36	$\langle 134; 157\rangle$	0.31	$\langle 144; 175\rangle$	0.28
	145	$\langle 197; 246\rangle$	0.41	$\langle 208; 266\rangle$	0.41	$\langle 174; 202\rangle$	0.41	$\langle 185; 222\rangle$	0.41
	180	$\langle 242; 301\rangle$	0.48	$\langle 255; 327\rangle$	0.49	$\langle 211; 245\rangle$	0.46	$\langle 225; 268\rangle$	0.47
	215	$\langle 284; 350\rangle$	0.96	$\langle 297; 376\rangle$	0.93	$\langle 248; 285\rangle$	0.93	$\langle 263; 310\rangle$	0.88
	250	$\langle 329; 407\rangle$	0.79	$\langle 346; 439\rangle$	0.79	$\langle 287; 330\rangle$	0.75	$\langle 305; 361\rangle$	0.77
5000	75	$\langle 104; 133\rangle$	0.20	$\langle 108; 139\rangle$	0.48	$\langle 76; 107\rangle$	0.13	$\langle 100; 126\rangle$	0.22
	110	$\langle 155; 202\rangle$	0.57	$\langle 162; 214\rangle$	0.29	$\langle 111; 152\rangle$	0.18	$\langle 146; 185\rangle$	0.28
	145	$\langle 199; 255\rangle$	0.36	$\langle 210; 272\rangle$	0.40	$\langle 146; 198\rangle$	0.21	$\langle 187; 234\rangle$	0.62
	180	$\langle 245; 317\rangle$	0.46	$\langle 258; 337\rangle$	0.55	$\langle 181; 250\rangle$	0.30	$\langle 230; 295\rangle$	0.49
	215	$\langle 287; 366\rangle$	0.85	$\langle 300; 386\rangle$	0.94	$\langle 216; 296\rangle$	0.75	$\langle 268; 338\rangle$	0.49
	250	$\langle 332; 426\rangle$	0.76	$\langle 349; 450\rangle$	0.74	$\langle 251; 347\rangle$	0.61	$\langle 312; 399\rangle$	0.95
10000	75	$\langle 76; 134\rangle$	0.13	$\langle 116; 174\rangle$	0.34	$\langle 76; 162\rangle$	0.15	$\langle 76; 147\rangle$	0.04
	110	$\langle 170; 270\rangle$	0.29	$\langle 171; 256\rangle$	0.19	$\langle 111; 230\rangle$	0.22	$\langle 111; 211\rangle$	0.10
	145	$\langle 216; 344\rangle$	0.62	$\langle 224; 332\rangle$	0.35	$\langle 146; 300\rangle$	0.26	$\langle 146; 280\rangle$	0.19
	180	$\langle 271; 439\rangle$	0.51	$\langle 279; 422\rangle$	0.69	$\langle 181; 377\rangle$	0.34	$\langle 181; 354\rangle$	0.25
	215	$\langle 317; 517\rangle$	0.85	$\langle 324; 493\rangle$	0.61	$\langle 216; 448\rangle$	0.70	$\langle 216; 423\rangle$	0.58
	250	$\langle 365; 593\rangle$	1.02	$\langle 375; 569\rangle$	0.99	$\langle 251; 512\rangle$	0.62	$\langle 251; 485\rangle$	0.67
20000	75	$\langle 76; 212\rangle$	0.14	$\langle 76; 201\rangle$	0.15	$\langle 76; 242\rangle$	0.15	$\langle 76; 228\rangle$	0.17
	110	$\langle 111; 306\rangle$	0.21	$\langle 111; 292\rangle$	0.13	$\langle 111; 352\rangle$	0.17	$\langle 111; 332\rangle$	0.16
	145	$\langle 146; 408\rangle$	0.27	$\langle 146; 388\rangle$	0.24	$\langle 146; 460\rangle$	0.39	$\langle 146; 437\rangle$	0.26
	180	$\langle 181; 514\rangle$	0.35	$\langle 181; 485\rangle$	0.49	$\langle 181; 575\rangle$	0.38	$\langle 181; 546\rangle$	0.54
	215	$\langle 216; 617\rangle$	0.67	$\langle 216; 585\rangle$	0.44	$\langle 216; 685\rangle$	0.50	$\langle 216; 652\rangle$	0.47
	250	$\langle 251; 713\rangle$	0.68	$\langle 251; 675\rangle$	0.60	$\langle 251; 787\rangle$	0.80	$\langle 251; 750\rangle$	0.79

Table 2
Test set P6