Dynamic Programming for Computing Non-Stationary (R,S) Policy Parameters

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

[1]Cork Constraint Computation Centre, University College Cork, Ireland
[2]Department of Management, Hacettepe University, Ankara, Turkey
[3]Faculty of Computer Science, Izmir University of Economics, Izmir, Turkey

15th International Symposium on Inventories, Budapest, Hungary

Outline

1. **Introduction**
   - Preliminaries
2. **Formal Background**
   - A modified Dynamic Programming algorithm with filtering
   - The stochastic production/inventory control problem
3. **The algorithm**
   - A State Space Relaxation
   - A filtering procedure for the relaxed State Space Graph
   - An augmenting procedure for the relaxed State Space Graph
   - A simple example
   - Experiments
4. **Related Works**
   - Related Works
5. **Conclusions**
   - Conclusions

Abstract

- In this work we propose an efficient **Dynamic Programming** approach for computing (R,S) policy parameters under **non-stationary stochastic demand** and **service level constraints**

Abstract

- In this work we propose an efficient **Dynamic Programming** approach for computing (R,S) policy parameters under **non-stationary stochastic demand** and **service level constraints**
- State Space Relaxation (Christofides et al. [4])

Abstract

- In this work we propose an efficient **Dynamic Programming** approach for computing (R,S) policy parameters under **non-stationary stochastic demand** and **service level constraints**
- State Space Relaxation (Christofides et al. [4])
- **State Space Filtering**

Abstract

- In this work we propose an efficient **Dynamic Programming** approach for computing (R,S) policy parameters under **non-stationary stochastic demand** and **service level constraints**
- State Space Relaxation (Christofides et al. [4])
- **State Space Filtering**
- State Space Augmentation (Boland et al. [2])

Abstract

- In this work we propose an efficient **Dynamic Programming** approach for computing (R,S) policy parameters under **non-stationary stochastic demand** and **service level constraints**
- State Space Relaxation (Christofides et al. [4])
- **State Space Filtering**
- State Space Augmentation (Boland et al. [2])
- `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.`

Abstract

- In this work we propose an efficient **Dynamic Programming** approach for computing (R,S) policy parameters under **non-stationary stochastic demand** and **service level constraints**
- State Space Relaxation (Christofides et al. [4])
- **State Space Filtering**
- State Space Augmentation (Boland et al. [2])
- `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 approach significantly outperforms earlier works

Dynamic Programming (DP)

- optimization procedure that solves optimization problems by decomposing them into a **nested family** of subproblems

Dynamic Programming (DP)

- optimization procedure that solves optimization problems by decomposing them into a **nested family** of subproblems
- based on the *principle of optimality* (Bellman [1])

Dynamic Programming (DP)

- optimization procedure that solves optimization problems by decomposing them into a **nested family** of subproblems
- based on the *principle of optimality* (Bellman [1])

- a problem P

Dynamic Programming (DP)

- optimization procedure that solves optimization problems by decomposing them into a **nested family** of subproblems
- based on the *principle of optimality* (Bellman [1])

- a problem P
- a *state space graph* $SG = (S, T)$

Dynamic Programming (DP)

- optimization procedure that solves optimization problems by decomposing them into a **nested family** of subproblems
- based on the *principle of optimality* (Bellman [1])

- a problem P
- a *state space graph* $SG = (S, T)$
  - $s \in S$ is a state

Dynamic Programming (DP)

- optimization procedure that solves optimization problems by decomposing them into a **nested family** of subproblems
- based on the *principle of optimality* (Bellman [1])

- a problem P
- a *state space graph* $SG = (S, T)$
  - $s \in S$ is a state
  - $t \in T$ represents a feasible transition between two states

Dynamic Programming (DP)

- optimization procedure that solves optimization problems by decomposing them into a **nested family** of subproblems
- based on the *principle of optimality* (Bellman [1])

- a problem P
- a *state space graph* $SG = (S, T)$
  - $s \in S$ is a state
  - $t \in T$ represents a feasible transition between two states
- *The original problem is solved by solving a shortest path problem in the state space graph from an initial state to a final state (boundary condition)*

Dynamic Programming (DP)

- optimization procedure that solves optimization problems by decomposing them into a **nested family** of subproblems
- based on the *principle of optimality* (Bellman [1])

- a problem P
- a *state space graph* $SG = (S, T)$
  - $s \in S$ is a state
  - $t \in T$ represents a feasible transition between two states
- *The original problem is solved by solving a shortest path problem in the state space graph from an initial state to a final state (boundary condition)*

As the authors remark in [7], this definition of DP is somehow restrictive, but it will be sufficient for the discussion in this paper

Dynamic Programming (DP)

- optimization procedure that solves optimization problems by decomposing them into a **nested family** of subproblems
- based on the *principle of optimality* (Bellman [1])

- a problem P
- a *state space graph* $SG = (S, T)$
  - $s \in S$ is a state
  - $t \in T$ represents a feasible transition between two states
- *The original problem is solved by solving a shortest path problem in the state space graph from an initial state to a final state (boundary condition)*

If the original problem is NP-hard, the correspondent state space graph will have an exponential number of nodes

State Space Relaxation (SSR)

Since the number of vertices in the state space graph is exponential, the concept of SSR has been widely studied in the Operations Research community, see (Christofides et al. [4])

State Space Relaxation (SSR)

A problem P with *state space graph* $SG = (S, T)$

- we build a new state space graph $RSG = (RS, RT)$ s.t.

State Space Relaxation (SSR)

A problem P with *state space graph* $SG = (S, T)$

- we build a new state space graph $RSG = (RS, RT)$ s.t.
  - a function $w(s)$ exists mapping every state $s \in S$ into a state $w(s) \in RS$

State Space Relaxation (SSR)

A problem P with *state space graph* $SG = (S, T)$

- we build a new state space graph $RSG = (RS, RT)$ s.t.
  - a function $w(s)$ exists mapping every state $s \in S$ into a state $w(s) \in RS$
  - for every arc $(s_i, s_j) \in T$, $(w(s_i), w(s_j))$ is an arc in RT

State Space Relaxation (SSR)

A problem P with *state space graph* $SG = (S, T)$

- we build a new state space graph $RSG = (RS, RT)$ s.t.
  - a function $w(s)$ exists mapping every state $s \in S$ into a state $w(s) \in RS$
  - for every arc $(s_i, s_j) \in T$, $(w(s_i), w(s_j))$ is an arc in RT
  - the cost of $(w(s_i), w(s_j))$ is less or equal to the cost of (s_i, s_j)

State Space Relaxation (SSR)

A problem P with *state space graph* $SG = (S, T)$

- we build a new state space graph $RSG = (RS, RT)$ s.t.
  - a function $w(s)$ exists mapping every state $s \in S$ into a state $w(s) \in RS$
  - for every arc $(s_i, s_j) \in T$, $(w(s_i), w(s_j))$ is an arc in RT
  - the cost of $(w(s_i), w(s_j))$ is less or equal to the cost of (s_i, s_j)

The relaxation represented by $RSG = (RS, RT)$ is, of course, useful if the number of vertices in the new graph is polynomial instead of exponential

Filtering the State Space Graph

A filtering procedure $\mathcal{F}$ is an algorithm that is able to detect provably suboptimal arcs in the state space graph $SG = (S, T)$

Filtering the State Space Graph

A filtering procedure $\mathcal{F}$ is an algorithm that is able to detect provably suboptimal arcs in the state space graph $SG = (S, T)$

A **filtering procedure** over a state space graph $SG = (S, T)$ can also be applied to a reduced state space graph $RSG = (RS, RT)$ for SG if such a strategy satisfies the following

Filtering the State Space Graph

A filtering procedure $\mathcal{F}$ is an algorithm that is able to detect provably suboptimal arcs in the state space graph $SG = (S, T)$

A **filtering procedure** over a state space graph $SG = (S, T)$ can also be applied to a reduced state space graph $RSG = (RS, RT)$ for SG if such a strategy satisfies the following

Property: An arc $a \in RT$ is necessarily preserved by the filtering procedure if there exists at least on arc $(s_i, s_j) \in T$ such that $(w(s_i), w(s_j)) = a$ and (s_i, s_j) is a candidate optimal arc in $SG = (S, T)$

Augmenting the Relaxed State Space Graph (Boland et al. [2])

Let

- $SG = (S, T)$ be a state space graph
- $RSG = (RS, RT)$ be a relaxed SSG for SG

Augmenting the Relaxed State Space Graph (Boland et al. [2])

Let

- $SG = (S, T)$ be a state space graph
- $RSG = (RS, RT)$ be a relaxed SSG for SG

an **augmenting procedure** $\mathcal{A}$ is a procedure that is able to augment RSG by introducing additional arcs and/or nodes in order to obtain a new graph $RSG^{\uparrow} = (RS^{\uparrow}, RT^{\uparrow})$, where $|RS^{\uparrow}| > |RS|$ and/or $|RT^{\uparrow}| > |RT|$, which respects the following

Augmenting the Relaxed State Space Graph (Boland et al. [2])

Let

- $SG = (S, T)$ be a state space graph
- $RSG = (RS, RT)$ be a relaxed SSG for SG

an **augmenting procedure** $\mathcal{A}$ is a procedure that is able to augment RSG by introducing additional arcs and/or nodes in order to obtain a new graph $RSG^{\uparrow} = (RS^{\uparrow}, RT^{\uparrow})$, where $|RS^{\uparrow}| > |RS|$ and/or $|RT^{\uparrow}| > |RT|$, which respects the following

Property: A function $w^{\uparrow}(s)$ exists mapping every state $s \in S$, for which a state $w(s) \in RS$ exists, into a state $w^{\uparrow}(s) \in RS^{\uparrow}$, and every arc $(s_i, s_j) \in T$, for which an arc $(w(s_i), w(s_j)) \in RT$ exists, into an arc in $(w^{\uparrow}(s_i), w^{\uparrow}(s_j)) \in RT^{\uparrow}$

A Modified Dynamic Programming Algorithm with Filtering

Algorithm 1: `Relax - Filter - Augment`

input : a problem instance P
input : a relaxed state space graph $RSG(RS, RT)$ for P
input : a sound filtering algorithm $\mathcal{F}$ for the state space graph of P
input : an augmenting procedure $\mathcal{A}$ for the relaxed state space graph $RSG(RS, RT)$ of P
output: the optimal solution cost C

begin
  $RSG(\widehat{RS}, \widehat{RT}) \leftarrow RSG(RS, RT)$ filtered using $\mathcal{F}$;
  solve a shortest path problem (SPP) over $RSG(\widehat{RS}, \widehat{RT})$;
  if *the shortest path in* $RSG(\widehat{RS}, \widehat{RT})$ *produces a feasible solution for* P **then**
    $C \leftarrow$ cost of the shortest path in $RSG(\widehat{RS}, \widehat{RT})$;
  else
    $RSG(\widehat{RS}^{\uparrow}, \widehat{RT}^{\uparrow}) \leftarrow RSG(\widehat{RS}, \widehat{RT})$ augmented using $\mathcal{A}$;
    solve a shortest path problem (SPP) over $RSG(\widehat{RS}^{\uparrow}, \widehat{RT}^{\uparrow})$;
    $C \leftarrow$ cost of the shortest path in $RSG(\widehat{RS}^{\uparrow}, \widehat{RT}^{\uparrow})$;
end

Problem definition

INSTANCE Number $N \in \mathbb{Z}^+$ of periods, for each period i, $1 \leq i \leq N$, a random demand $d_i \in \mathbb{Z}_0^+$ with probability mass function $g_i(d_i)$. A production set-up cost $a \in \mathbb{Z}^+$, an inventory holding cost coefficient $h \in \mathbb{Z}^+$, a non-stock-out probability $0 < \alpha < 1$.
QUESTION Production amounts $X_i \in \mathbb{Z}_0^+$, and associated closing inventory levels $I_i = \sum_{t=1}^{i}(X_t - d_t)$, $1 \leq i \leq N$, such that $\Pr\{I_i \geq 0\} \geq \alpha$, and $E\left[\sum_{i=1}^{N}(h_i I_i) + \sum_{X_i>0} a_i\right]$, where $E[\cdot]$ is the expectation operator, is minimized?

Problem definition

Assumptions

- an (R^n, S^n) policy is adopted
- assumptions on negative orders as in (Tarim & Kingsman [15])

Problem definition

Assumptions

- an (R^n, S^n) policy is adopted
- assumptions on negative orders as in (Tarim & Kingsman [15])

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

Problem definition

Assumptions

- an (R^n, S^n) policy is adopted
- assumptions on negative orders as in (Tarim & Kingsman [15])

Definition: A replenishment cycle, $T(i,j)$, is the time span between two consecutive orders 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 $T(i,j)$.

Problem definition

The (R^n,S^n) policy

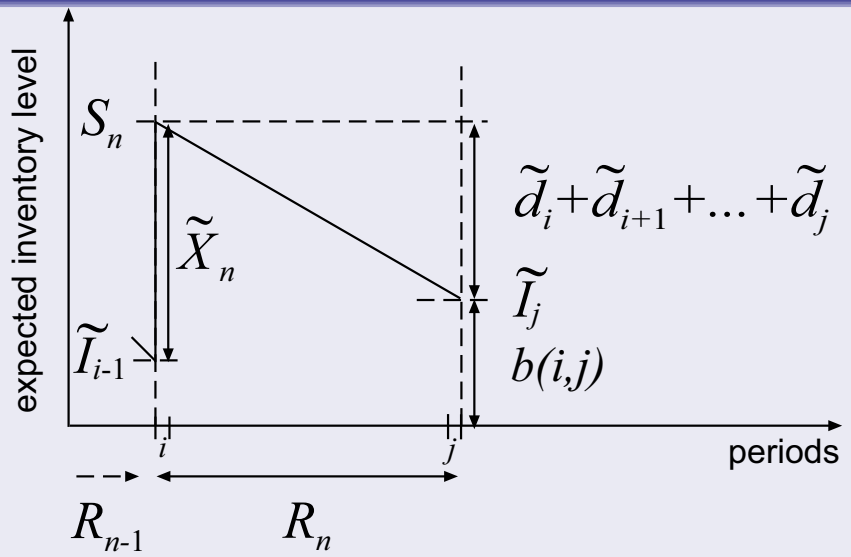


Figure: (R^n,S^n) policy. $\tilde{d}_i + \tilde{d}_{i+1} + \ldots + \tilde{d}_j$ is the expected demand over R^n; $b(i,j)$ is the minimum buffer stock required to guarantee service level α; $\tilde{X}_n$ is the expected order quantity in period i for replenishment cycle n; $\tilde{I}_{i-1}$ and $\tilde{I}_j$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

Problem definition

Deterministic equivalent (Tarim & Kingsman [15])

INSTANCE Number $N \in \mathbb{Z}^+$ of periods, for each period i, $1 \leq i \leq N$, an expected demand $\tilde{d}_i \in \mathbb{Z}_0^+$. A production set-up cost $a \in \mathbb{Z}^+$, an inventory cost coefficient $h \in \mathbb{Z}^+$, and an $N \times N$ buffer stock matrix $b(\cdot, \cdot)$

Problem definition

Deterministic equivalent (Tarim & Kingsman [15])

INSTANCE Number $N \in \mathbb{Z}^+$ of periods, for each period i, $1 \leq i \leq N$, an expected demand $\tilde{d}_i \in \mathbb{Z}_0^+$. A production set-up cost $a \in \mathbb{Z}^+$, an inventory cost coefficient $h \in \mathbb{Z}^+$, and an $N \times N$ buffer stock matrix $b(\cdot, \cdot)$
QUESTION Set of ordering decisions $\delta_i \in \{0, 1\}$, expected production amounts $\tilde{X}_i \in \mathbb{Z}_0^+$, and associated expected closing inventory levels $\tilde{I}_i = \sum_{t=1}^{i}(\tilde{X}_t - \tilde{d}_t)$, $1 \leq i \leq N$, such that $(\tilde{X}_i > 0) \Rightarrow (\delta_i = 1)$,

$$\tilde{I}_t \geq b(\max\{i | i \leq t \wedge \delta_i = 1\}, t),$$

and $\sum_{i=1}^{N} \left(a_i \delta_i + h_i \tilde{I}_i \right)$ is minimized.

Problem definition

A Dynamic Programming Formulation

- N is the number of steps in the system

Problem definition

A Dynamic Programming Formulation

- N is 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

Problem definition

A Dynamic Programming Formulation

- N is 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 states how many subsequent periods this order should cover

Problem definition

A Dynamic Programming Formulation

- N is 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 states how many subsequent periods this order should cover
- 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

Problem definition

A Dynamic Programming Formulation

- N is 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 states how many subsequent periods this order should cover
- 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

Problem definition

A Dynamic Programming Formulation

- N is 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 states how many subsequent periods this order should cover
- 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
  - 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

Problem definition

A Dynamic Programming Formulation

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

Problem definition

A Dynamic Programming Formulation

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

$$s_k = \begin{cases} s_{k-1} - \tilde{d}_k & \text{if } x_k \text{ states that no order is placed,} \\ \max(s_{k-1} - \tilde{d}_k, b(k,j) + \sum_{i=k+1}^{j} \tilde{d}_i) & \text{if } x_k \text{ states that an order is placed to cover next periods till } j. \end{cases} \quad (1)$$

Problem definition

A Dynamic Programming Formulation

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

$$s_k = \begin{cases} s_{k-1} - \tilde{d}_k & \text{if } x_k \text{ states that no order is placed,} \\ \max(s_{k-1} - \tilde{d}_k, b(k,j) + \sum_{i=k+1}^{j} \tilde{d}_i) & \text{if } x_k \text{ states that an order is placed to cover next periods till } j. \end{cases} \quad (1)$$

- X_k, the set of possible decisions at period k, comprises: the decision of not placing an order, the decision of covering 1 period with the order placed, the decision of covering 2 periods with the order placed, . . ., the decision of covering $N - k + 1$ periods with the order placed.

Problem definition

A Dynamic Programming Formulation

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

$$s_k = \begin{cases} s_{k-1} - \tilde{d}_k & \text{if } x_k \text{ states that no order is placed,} \\ \max(s_{k-1} - \tilde{d}_k, b(k,j) + \sum_{i=k+1}^{j} \tilde{d}_i) & \text{if } x_k \text{ states that an order is placed to cover next periods till } j. \end{cases} \quad (1)$$

- X_k, the set of possible decisions at period k, comprises: the decision of not placing an order, the decision of covering 1 period with the order placed, the decision of covering 2 periods with the order placed, ..., the decision of covering $N - k + 1$ periods with the order placed.
- 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 that an order should be always placed at period 1

Problem definition

A Dynamic Programming Formulation

The following objective function is minimized:

$$z = \min \left\{ \sum_{k=1}^{n} p_k(s_k, x_k) \right\}. \tag{2}$$

Problem definition

A Dynamic Programming Formulation

The following objective function is minimized:

$$z = \min \left\{ \sum_{k=1}^{n} p_k(s_k, x_k) \right\}. \tag{2}$$

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.

Problem definition

A Dynamic Programming Formulation

The following objective function is minimized:

$$z = \min \left\{ \sum_{k=1}^{n} p_k(s_k, x_k) \right\}. \tag{2}$$

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:

$$f_i(s_i) = \min_{x_i \in X_i} \left\{ \min_{s_i \in S_{i-1}} \{ f_{i-1}(s_{i-1}) + p_i(s_i, x_i) \} \right\} \tag{3}$$

where $s_i = t_i(s_{i-1}, x_i)$.

Problem definition

A Dynamic Programming Formulation

The following objective function is minimized:

$$z = \min \left\{ \sum_{k=1}^{n} p_k(s_k, x_k) \right\}. \tag{2}$$

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:

$$f_i(s_i) = \min_{x_i \in X_i} \left\{ \min_{s_i \in S_{i-1}} \{ f_{i-1}(s_{i-1}) + p_i(s_i, x_i) \} \right\} \tag{3}$$

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

$f_1(s_1) = \min_{x_1 \in X_1} \{p_1(s_1, x_1)\}$, where $s_1 = t_1(s_0, x_1)$.

A State Space Relaxation (Tarim [13])

- The model proposed can be reduced to a Shortest Path Problem if we relax the assumption on positivity of the expected order quantities.

A State Space Relaxation (Tarim [13])

- The model proposed can be reduced to a Shortest Path Problem if we relax the assumption on positivity of the expected order quantities.
- This corresponds to allowing negative replenishment quantities, or the ability to sell stock back to the supplier.

A State Space Relaxation (Tarim [13])

- The model proposed can be reduced to a Shortest Path Problem if we relax the assumption on positivity of the expected order quantities.
- This corresponds to allowing negative replenishment quantities, or the ability to sell stock back to the supplier.
- We now have a set $\mathcal{S}$ of $N(N+1)/2$ possible different replenishment cycles and the respective costs. Our new problem is to find an optimal set $\mathcal{S}^* \subset \mathcal{S}$ of consecutive disjoint replenishment cycles that covers our planning horizon at the minimum cost.

A State Space Relaxation (Tarim [13])

- The model proposed can be reduced to a Shortest Path Problem if we relax the assumption on positivity of the expected order quantities.
- This corresponds to allowing negative replenishment quantities, or the ability to sell stock back to the supplier.
- We now have a set $\mathcal{S}$ of $N(N+1)/2$ possible different replenishment cycles and the respective costs. Our new problem is to find an optimal set $\mathcal{S}^* \subset \mathcal{S}$ of consecutive disjoint replenishment cycles that covers our planning horizon at the minimum cost.

Definition: Cycle cost, $c(i,j)$, denotes the expected cost of the optimal policy for $T(i,j)$.

A State Space Relaxation (Tarim [13])

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 $c(i, j-1)$

A State Space Relaxation (Tarim [13])

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 $c(i, j-1)$

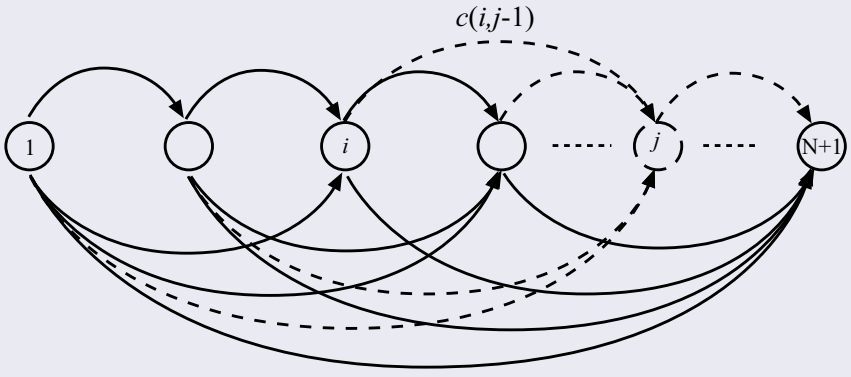


Figure: Shortest path problem graph

A State Space Relaxation (Tarim [13])

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 $c(i, j-1)$

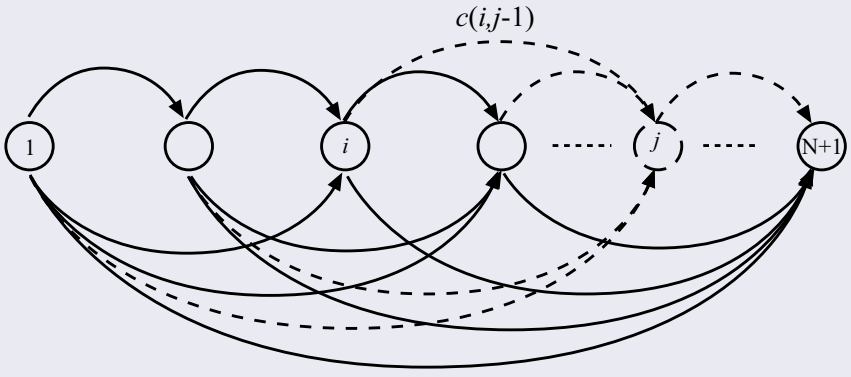


Figure: Shortest path problem graph

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.

Filtering procedure for the relaxed State Space Graph (Tarim & Smith [16])

We now aim to reduce *a priori* as much as possible the number of arcs in the graph we defined in the former section.

Filtering procedure for the relaxed State Space Graph (Tarim & Smith [16])

We now aim to reduce *a priori* as much as possible the number of arcs in the graph we defined in the former section.

- **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 $R(i,j) = b(i,j) + \sum_{t=i}^{j} \tilde{d}_t$.

Filtering procedure for the relaxed State Space Graph (Tarim & Smith [16])

We now aim to reduce *a priori* as much as possible the number of arcs in the graph we defined in the former section.

- **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 $R(i,j) = b(i,j) + \sum_{t=i}^{j} \tilde{d}_t$.
- **Proposition (Tarim and Smith [16])** 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.

Filtering procedure for the relaxed State Space Graph (Tarim & Smith [16])

We now aim to reduce *a priori* as much as possible the number of arcs in the graph we defined in the former section.

- **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 $R(i,j) = b(i,j) + \sum_{t=i}^{j} \tilde{d}_t$.
- **Proposition (Tarim and Smith [16])** 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 $t > j+1$.

Augmenting procedure for the relaxed State Space Graph

- This process is repeated for each node $i \in \{1, \ldots, N+1\}$, therefore we will only describe the steps performed on a single node

Augmenting procedure for the relaxed State Space Graph

- This process is repeated for each node $i \in \{1, \ldots, N+1\}$, therefore we will only describe the steps performed on a single node
- consider every inbound arc (p, i) at node i with associated buffer b^*
- for each outbound arc (i, j) in the graph 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 b^*

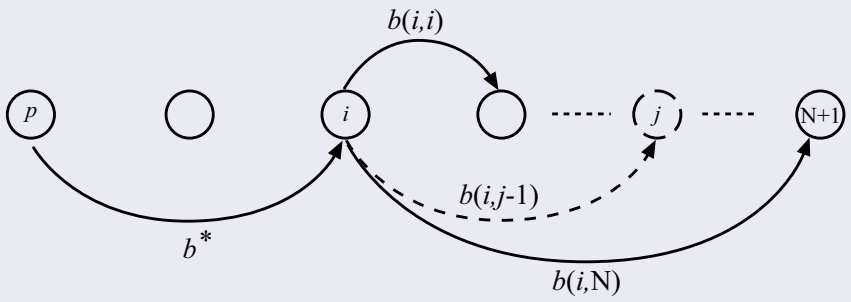


Figure: Feasible node point

Augmenting procedure for the relaxed State Space Graph

- If for some pair of subsequent replenishment cycles a negative order quantity is scheduled, in order to resolve this infeasibility we operate the following transformation:
- 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 b^*

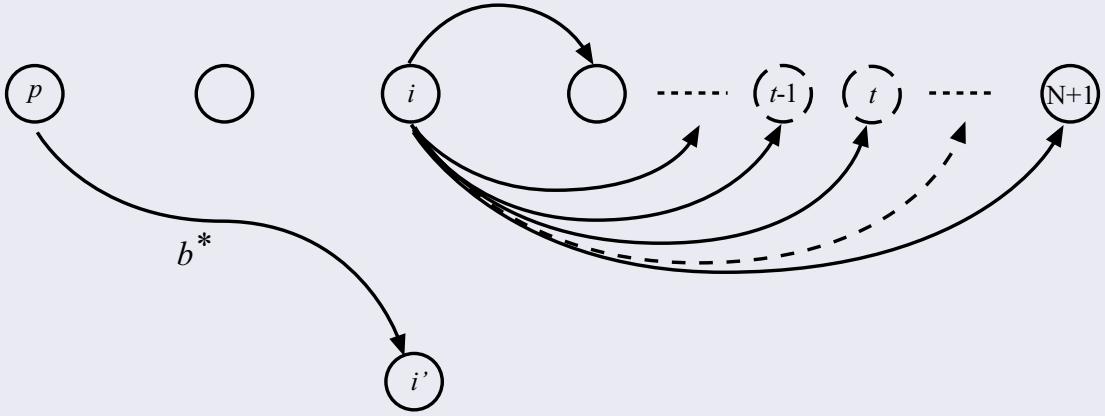


Figure: Infeasible node point

Augmenting procedure for the relaxed State Space Graph

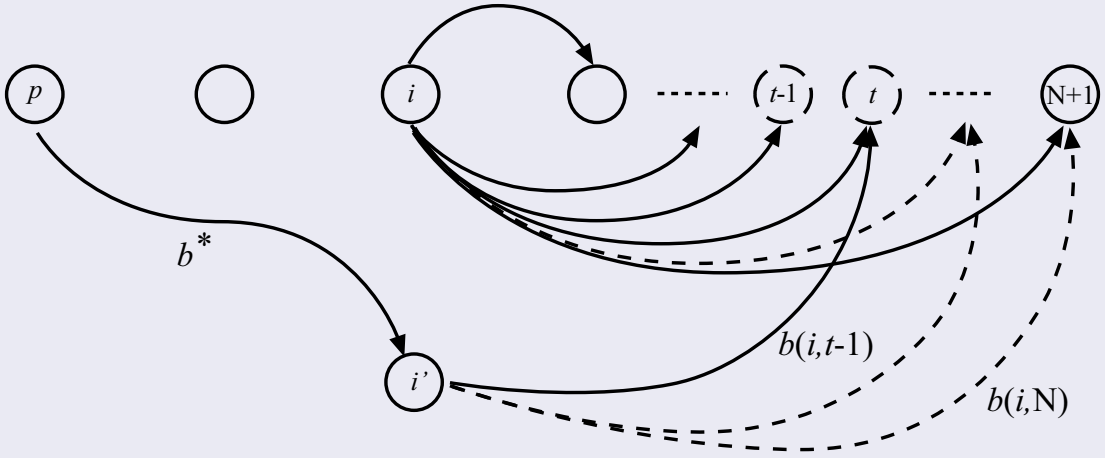


Figure: Infeasible node point

Then we connect this new node the following way:
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 $b(i, t-1)$. 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 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

Augmenting procedure for the relaxed State Space Graph

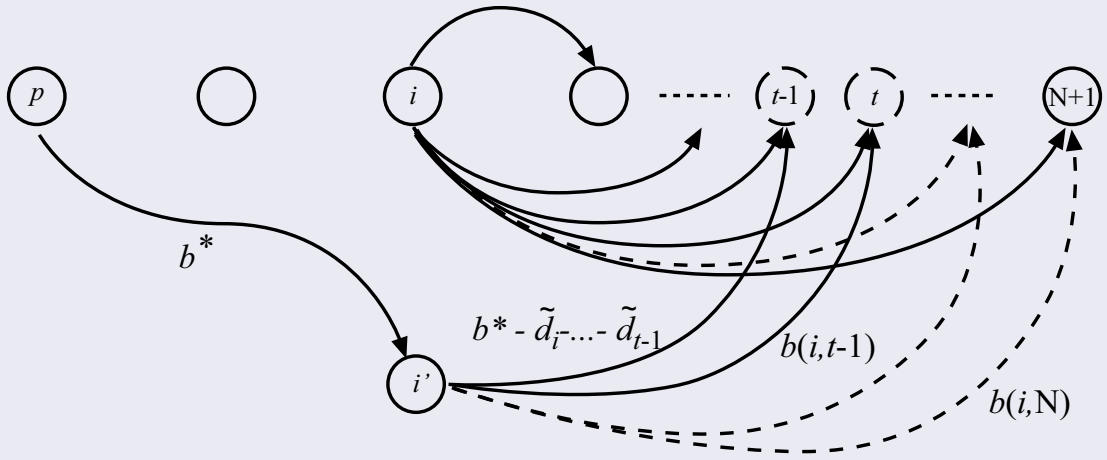


Figure: Infeasible node point

Then we connect this new node the following way:
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 $b^* - \sum_{k=i}^{t-1} \tilde{d}_k$. Obviously arcs $(i', t-2), (i', t-3), \ldots$ are suboptimal and should not be introduced, since the inventory carried on from the former replenishment cycle is enough to cover subsequent periods up to $t-1$

Input data

- 5-period planning horizon

Input data

- 5-period planning horizon
- expected values for period demand are $[100, 125, 25, 40, 30]$

Input data

- 5-period planning horizon
- expected values for period demand are $[100, 125, 25, 40, 30]$
- initial null inventory level

Input data

- 5-period planning horizon
- expected values for period demand are $[100, 125, 25, 40, 30]$
- initial null inventory level
- normally distributed demand for every period with a coefficient of variation $\sigma_t / \tilde{d}_t = 0.3$

Input data

- 5-period planning horizon
- expected values for period demand are $[100, 125, 25, 40, 30]$
- initial null inventory level
- normally distributed demand for every period with a coefficient of variation $\sigma_t / \tilde{d}_t = 0.3$
- ordering cost value $a = 50$

Input data

- 5-period planning horizon
- expected values for period demand are $[100, 125, 25, 40, 30]$
- initial null inventory level
- normally distributed demand for every period with a coefficient of variation $\sigma_t / \tilde{d}_t = 0.3$
- ordering cost value $a = 50$
- holding cost $h = 1$

Input data

- 5-period planning horizon
- expected values for period demand are $[100, 125, 25, 40, 30]$
- initial null inventory level
- normally distributed demand for every period with a coefficient of variation $\sigma_t / \tilde{d}_t = 0.3$
- ordering cost value $a = 50$
- holding cost $h = 1$
- non-stock-out probability in each period set to $\alpha = 0.95$

Relax & Filter

Firstly we build the connection matrix for the relaxed problem...

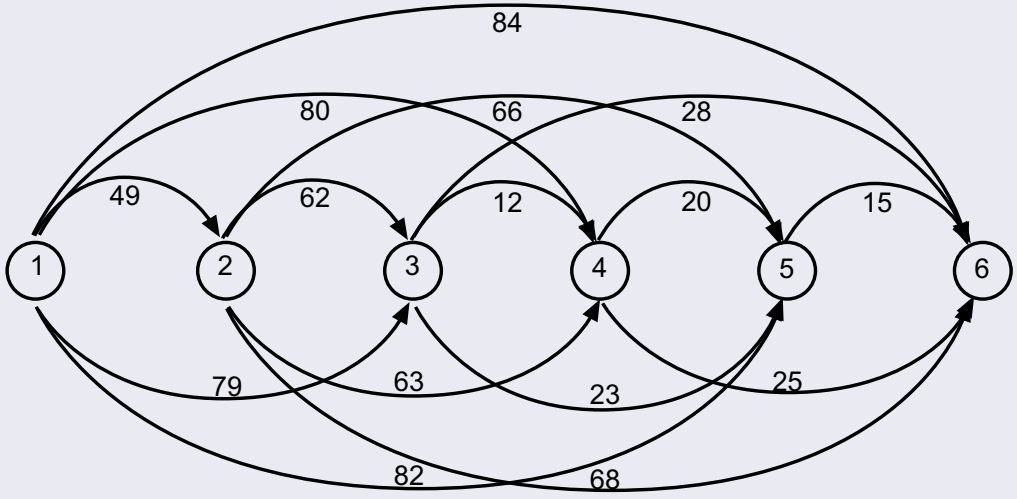


Figure: Connection matrix with expected buffer stock levels

Relax & Filter

Firstly we build the connection matrix for the relaxed problem...

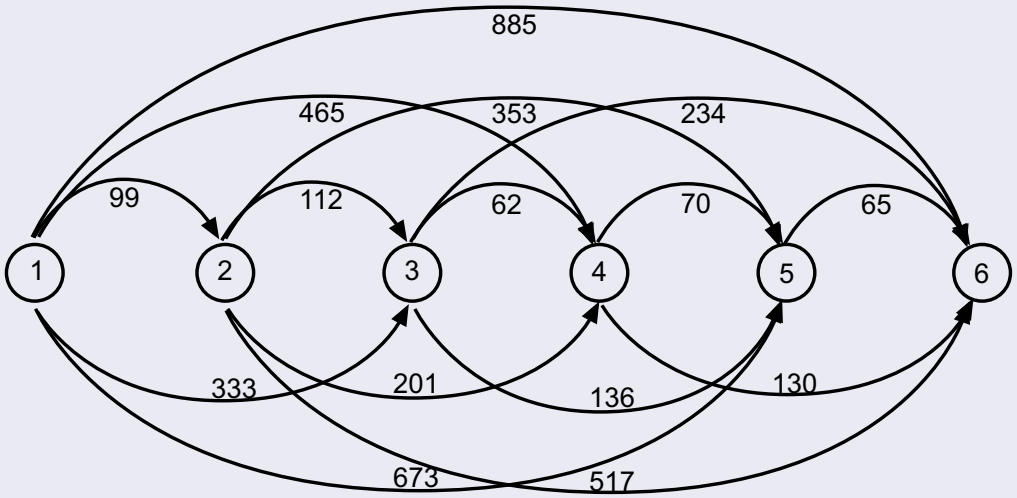


Figure: Connection matrix with expected cycle costs

Relax & Filter

Firstly we build the connection matrix for the relaxed problem...
...and we filter it by using the procedure already described...

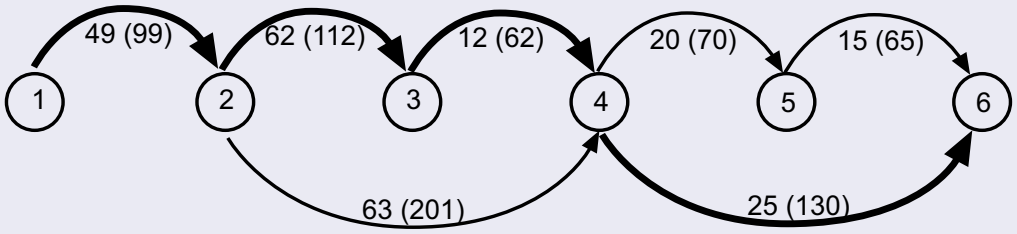


Figure: Filtered connection matrix. Expected buffer stock levels and expected cycle costs (in parenthesis) are shown for each arc. The shortest path is highlighted

The shortest path in this reduced network has a cost of 403. The order periods and the expected order quantities are respectively $[1, 2, 3, 4]$ and $[149, 138, -25, 83]$

Relax & Filter

Firstly we build the connection matrix for the relaxed problem...
...obtaining the following plan...

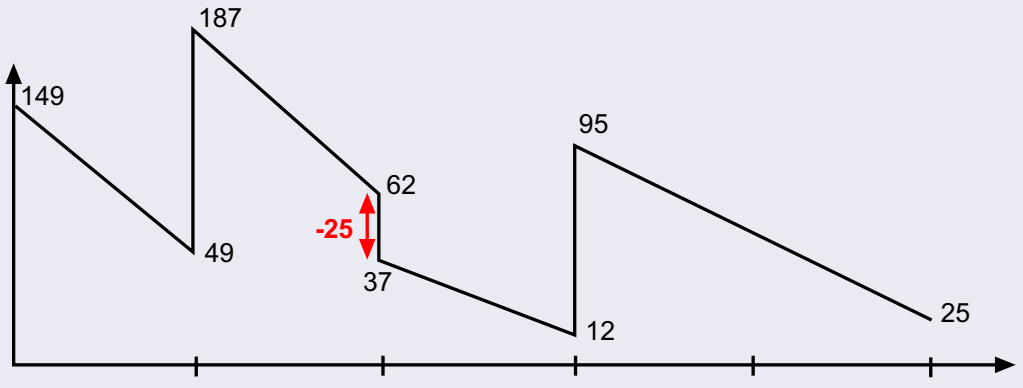
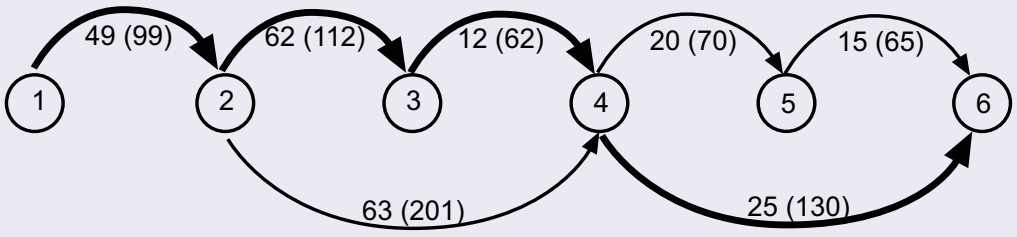
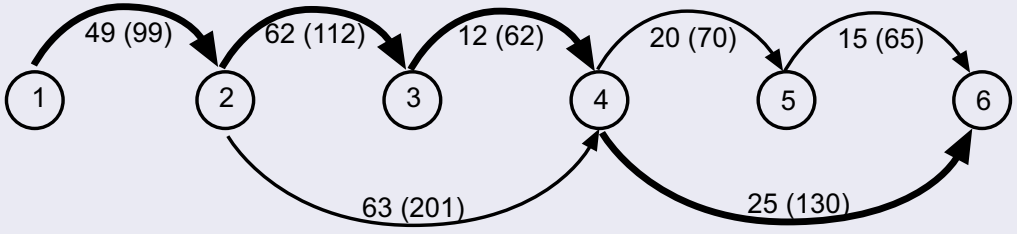


Figure: The respective plan

Augment

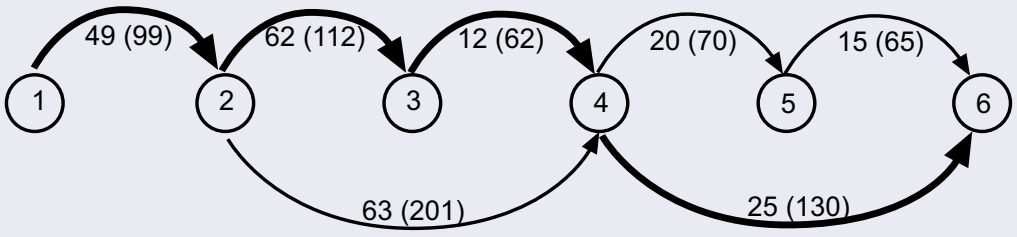


Augment

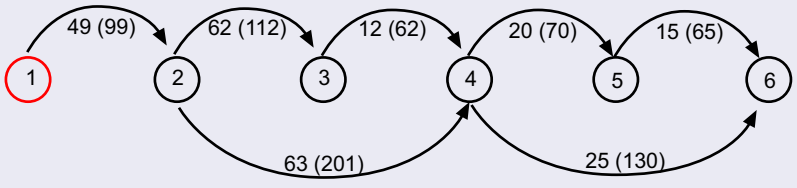


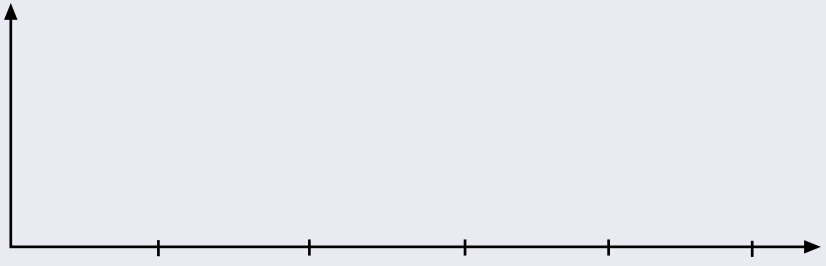
According to the procedure described we augment the filtered graph...

Augment

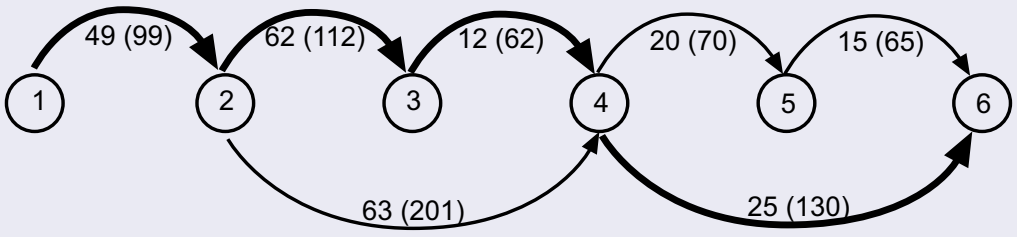


According to the procedure described we augment the filtered graph...

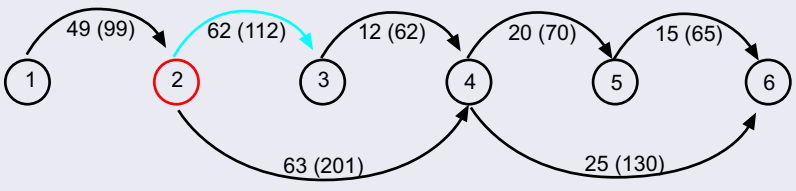


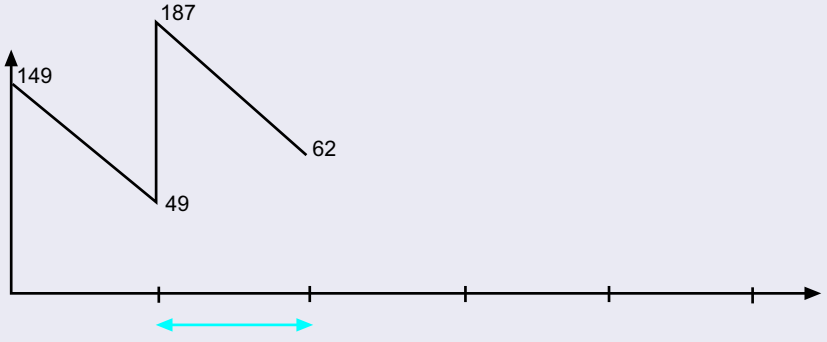


Augment

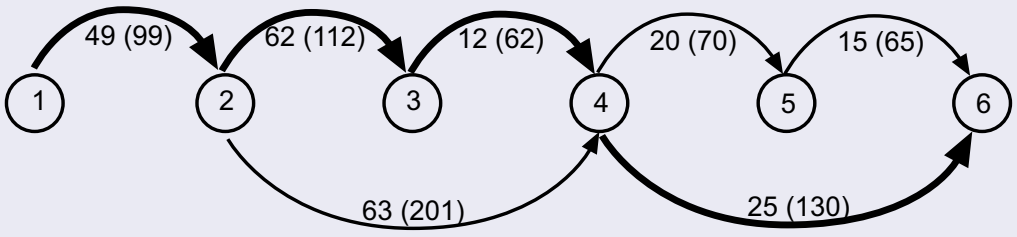


According to the procedure described we augment the filtered graph...

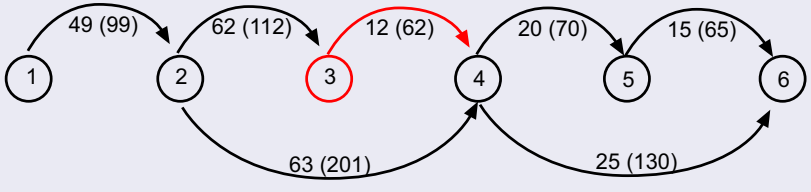


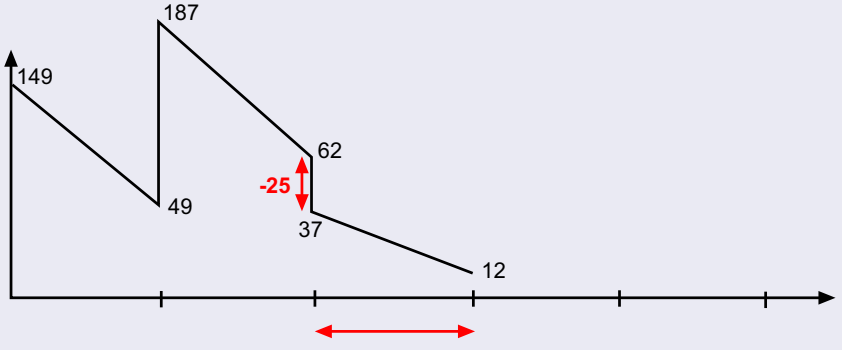


Augment

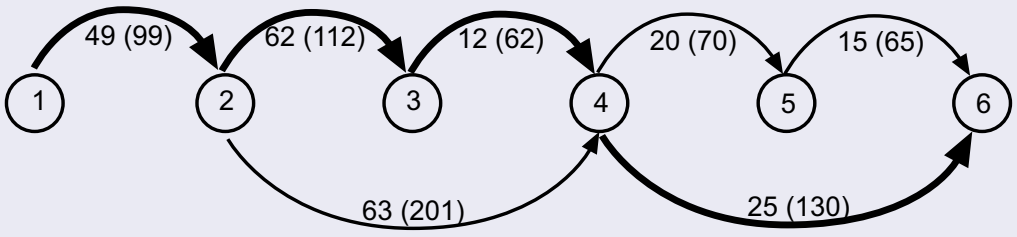


According to the procedure described we augment the filtered graph...

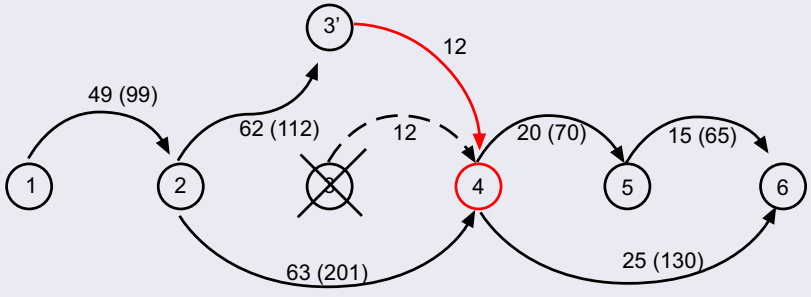


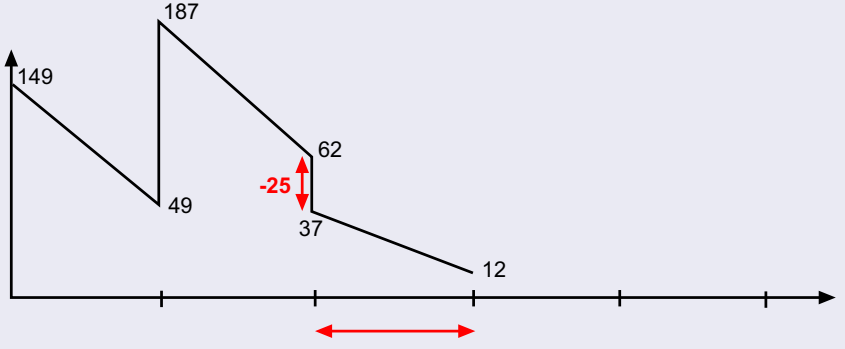


Augment

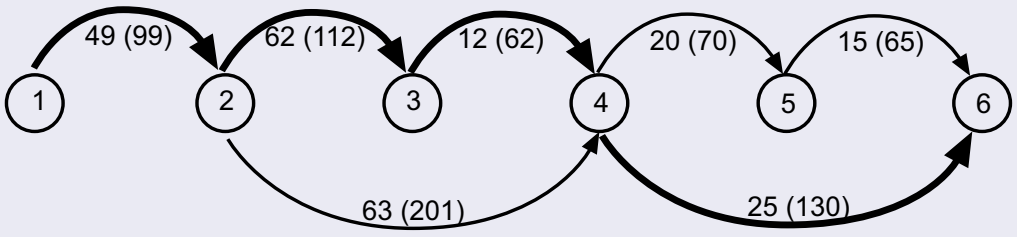


According to the procedure described we augment the filtered graph...

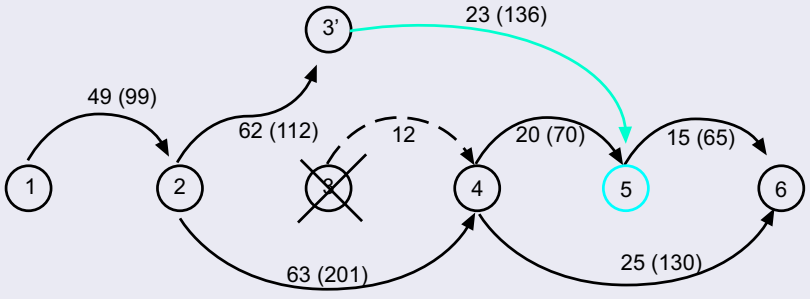


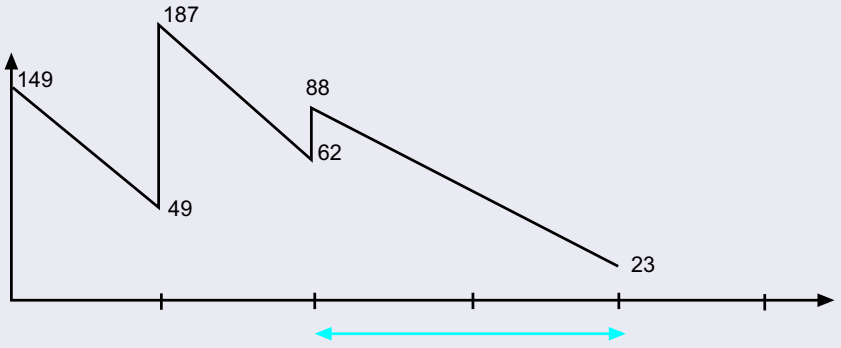


Augment

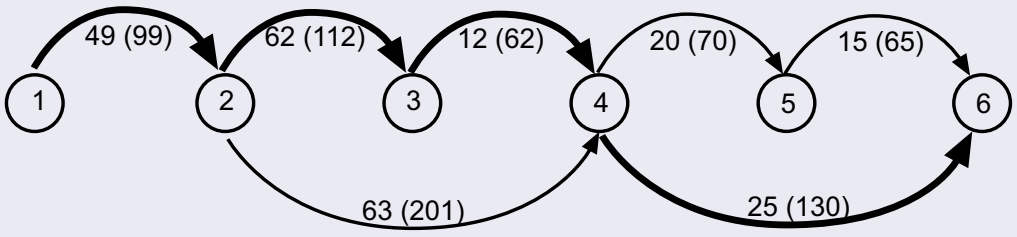


According to the procedure described we augment the filtered graph...

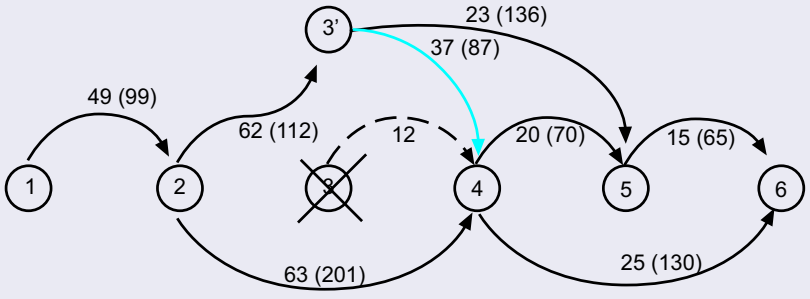


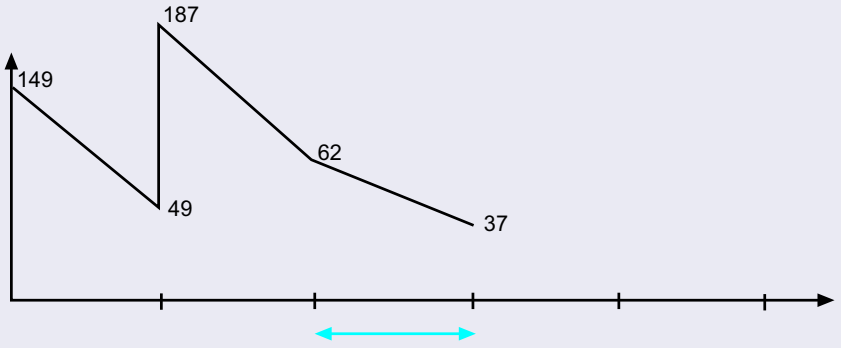


Augment

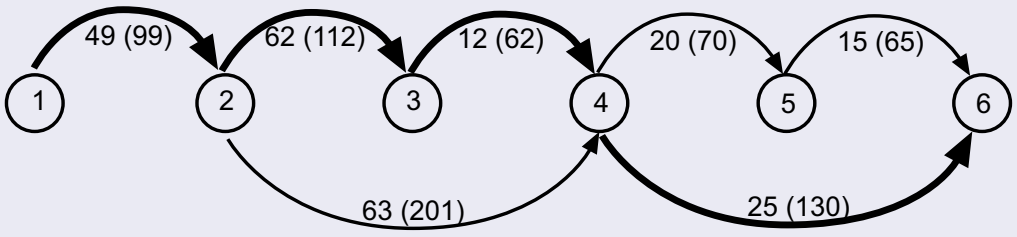


According to the procedure described we augment the filtered graph...

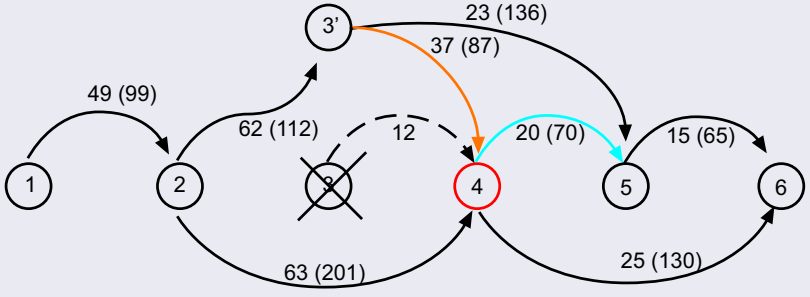


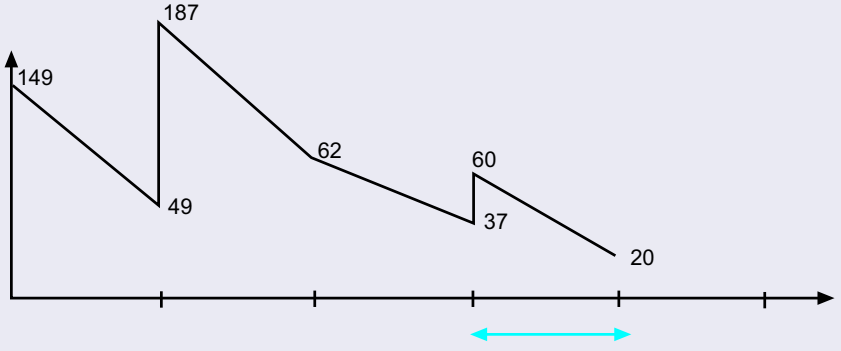


Augment

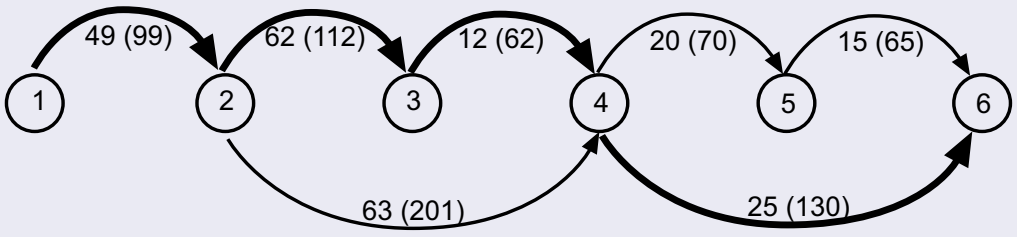


According to the procedure described we augment the filtered graph...

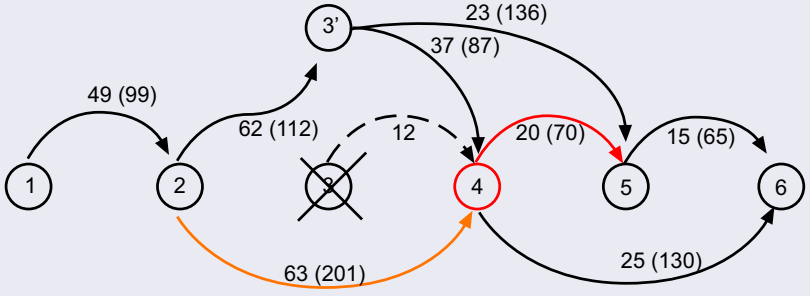


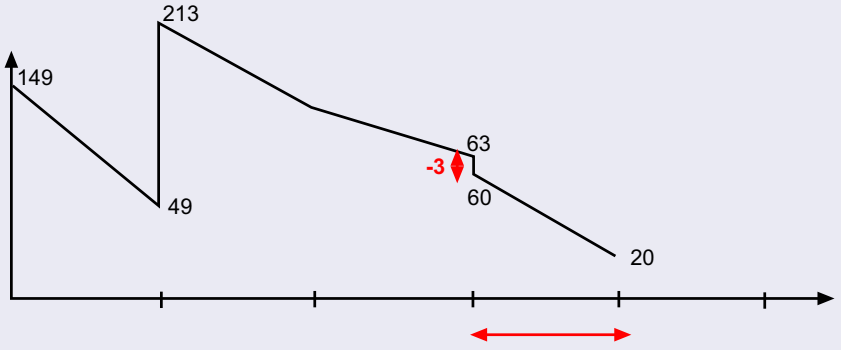


Augment

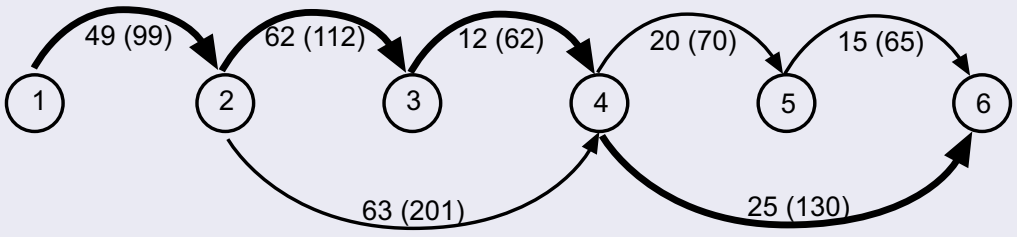


According to the procedure described we augment the filtered graph...

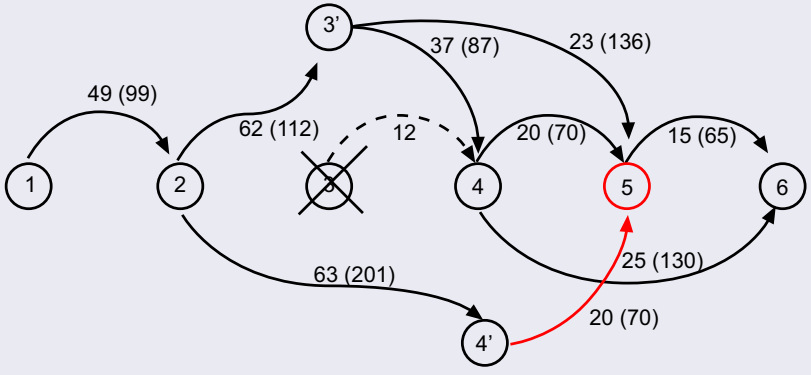


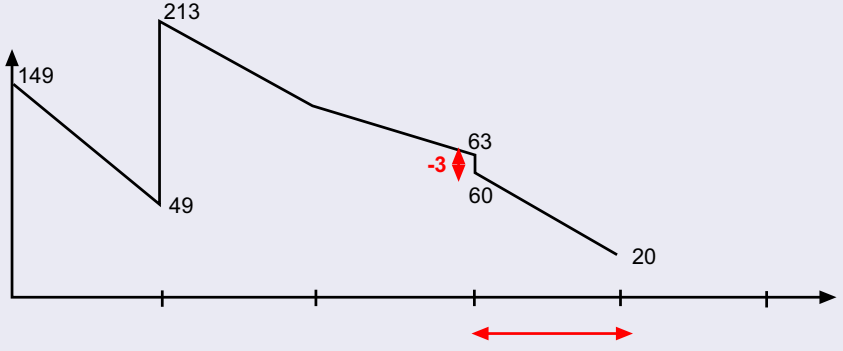


Augment

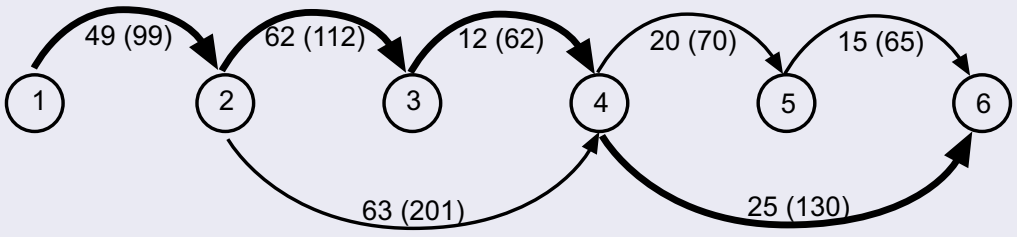


According to the procedure described we augment the filtered graph...

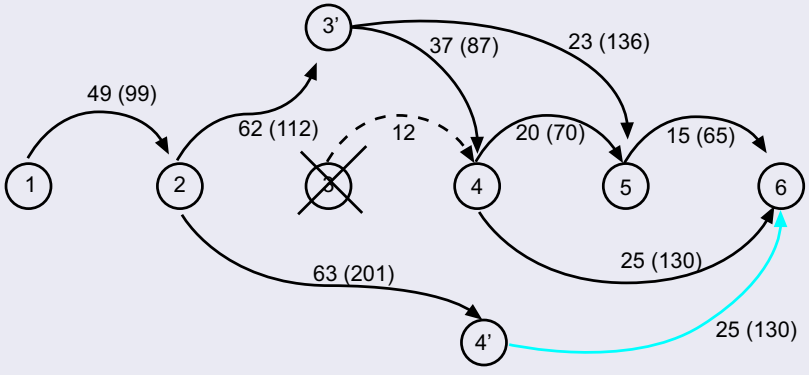


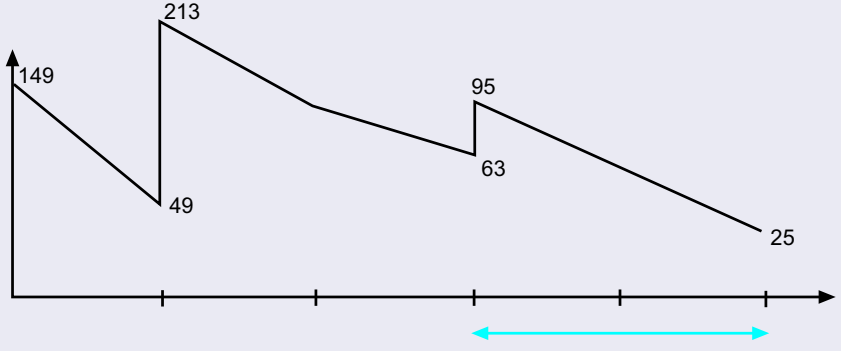


Augment

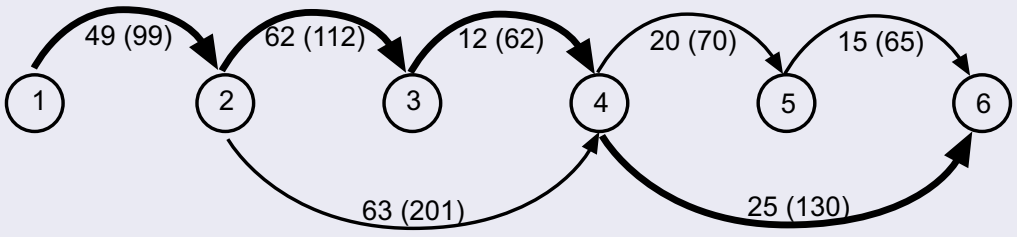


According to the procedure described we augment the filtered graph...

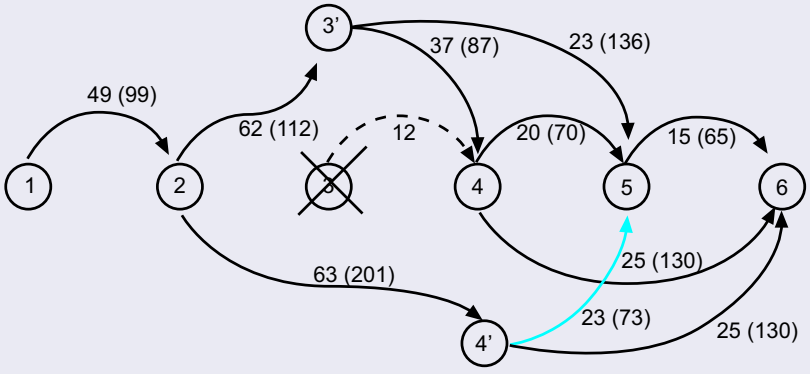


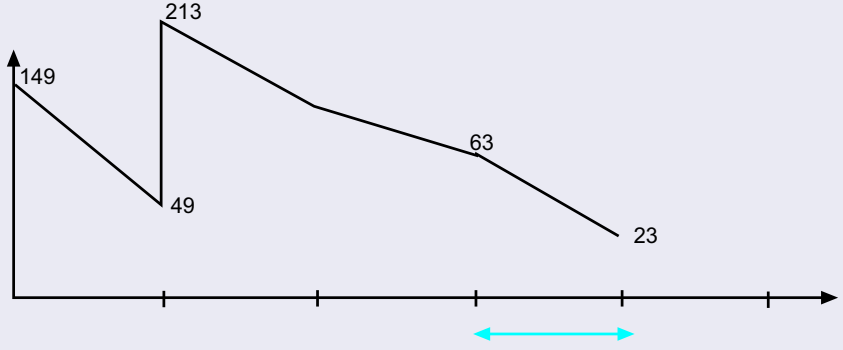


Augment

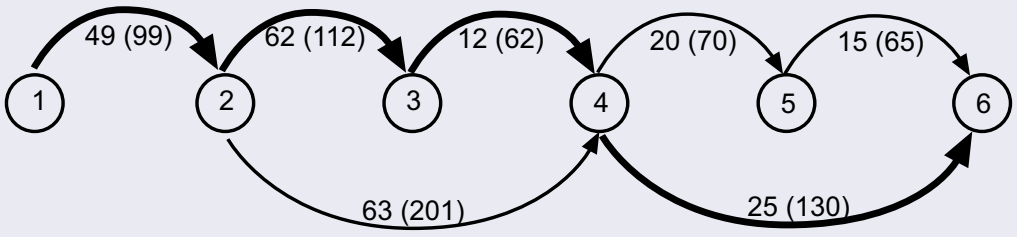


According to the procedure described we augment the filtered graph...

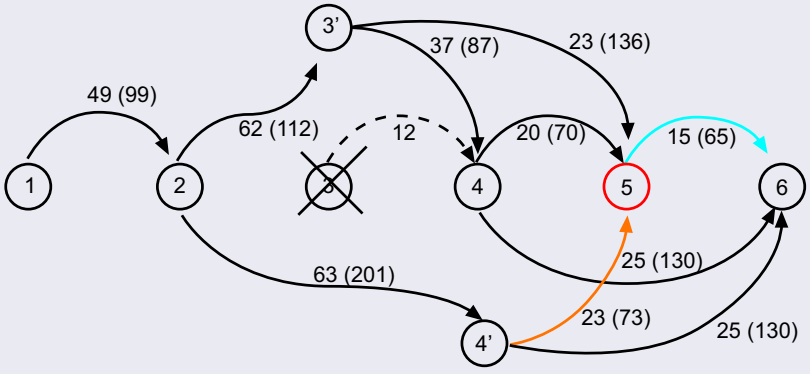


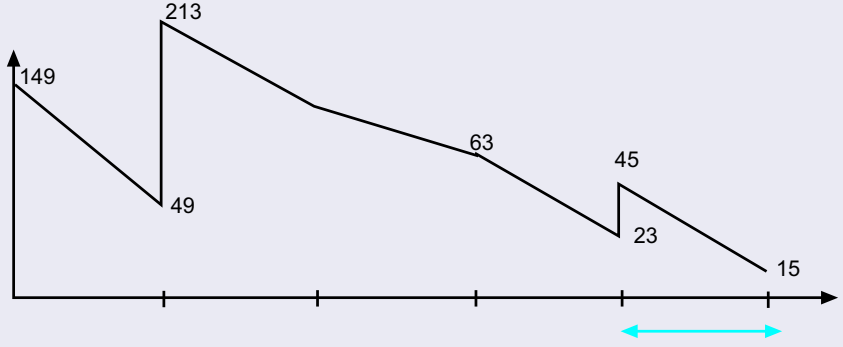


Augment

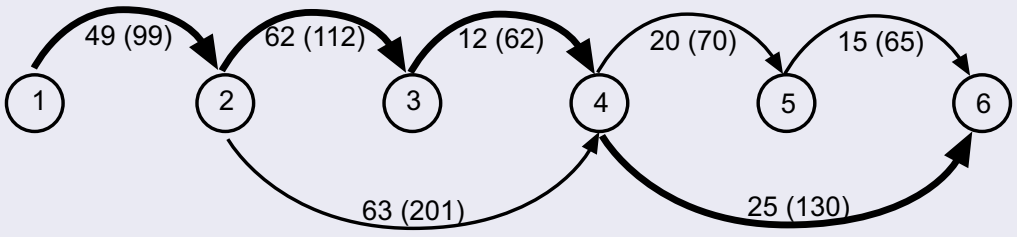


According to the procedure described we augment the filtered graph...

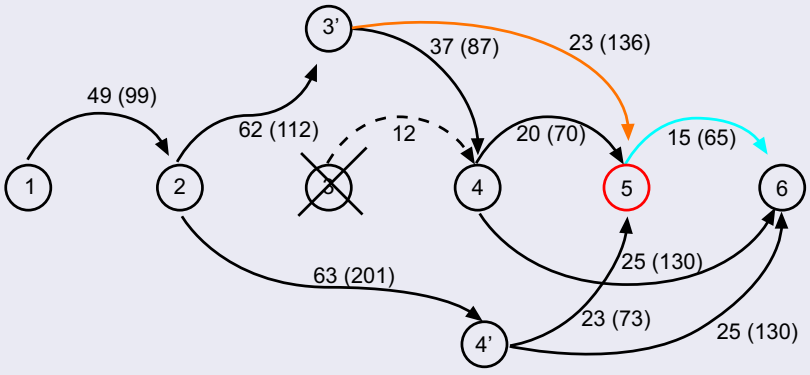


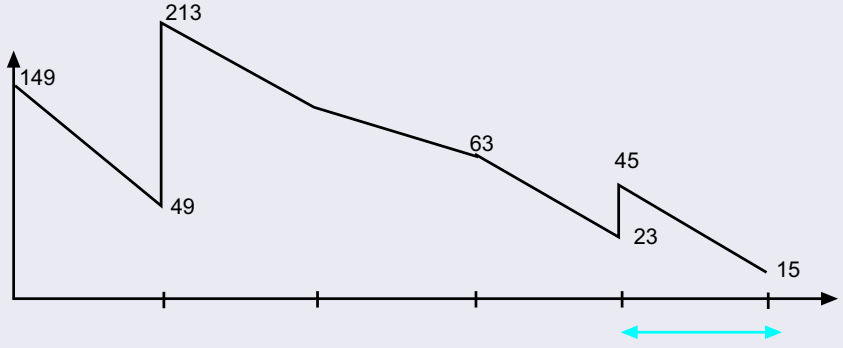


Augment

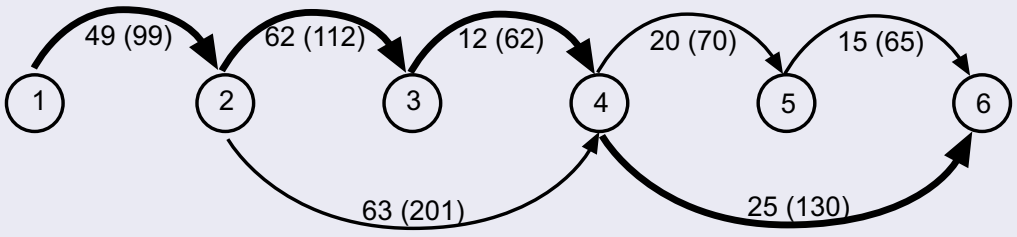


According to the procedure described we augment the filtered graph...

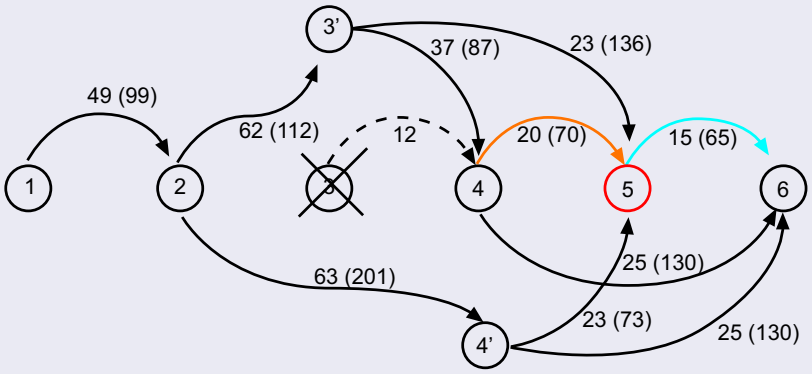


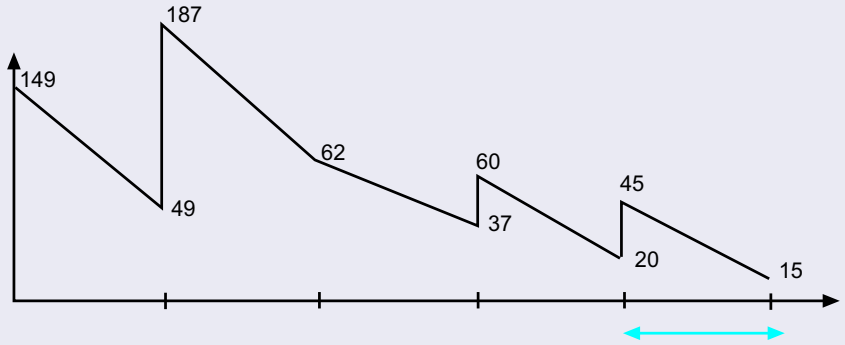


Augment

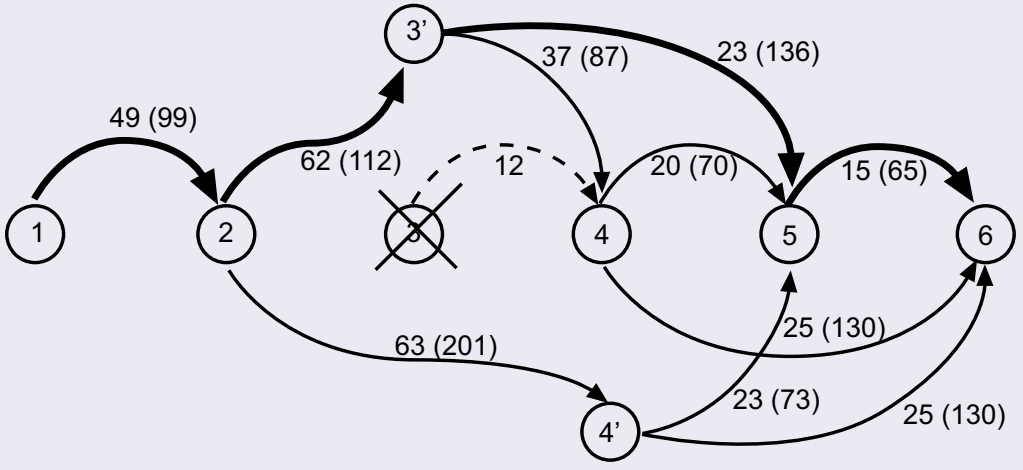


According to the procedure described we augment the filtered graph...





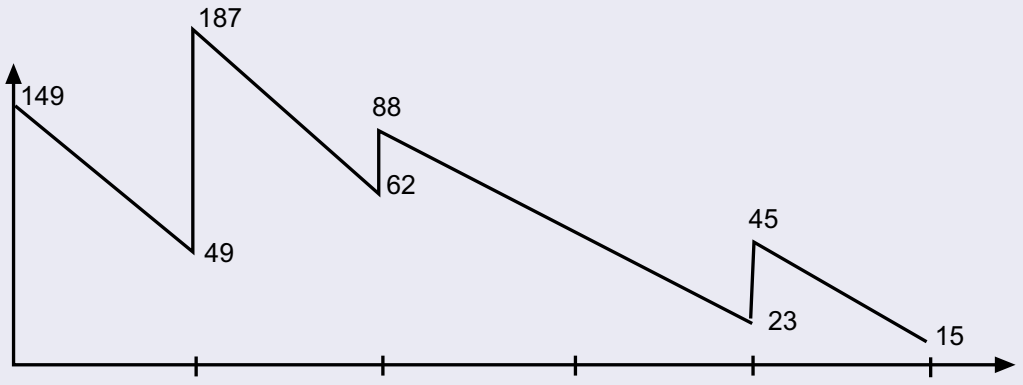
Augment



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

- The shortest path in this augmented network has a cost of 412 and represents the optimal solution cost of our original problem
- The order periods and the order quantities are therefore respectively $[1, 2, 3, 5]$ and $[149, 138, 26, 22]$

Augment



The optimal replenishment plan corresponding to the shortest path computed

- The shortest path in this augmented network has a cost of 412 and represents the optimal solution cost of our original problem
- The order periods and the order quantities are therefore respectively $[1, 2, 3, 5]$ and $[149, 138, 26, 22]$

Input data - 1

- We compare results obtained with our approach with those produced by the state-of-the-art approach presented in (Tarim et al. [14])

Input data - 1

- We compare results obtained with our approach with those produced by the state-of-the-art approach presented in (Tarim et al. [14])
- All experiments presented here were performed on an Intel(R) Centrino(TM) CPU 1.50GHz with 500Mb RAM. Note that this is the same hardware used for the experiments in [14]

Input data - 1

- We compare results obtained with our approach with those produced by the state-of-the-art approach presented in (Tarim et al. [14])
- All experiments presented here were performed on an Intel(R) Centrino(TM) CPU 1.50GHz with 500Mb RAM. Note that this is the same hardware used for the experiments in [14]
- We consider expected period demands, $\tilde{d}_t$, generated as uniformly distributed random numbers in $[0, 100]$
- fixed ordering cost $a \in \{25, 50, 100, 200\}$
- holding cost $h = 1$ per unit per period
- coefficient of variation $\sigma_t / \tilde{d}_t \in \{1/3, 1/6\}$
- The planning horizon length takes the following values $\{50, 75\}$
- service levels $\alpha = 0.95$ ($z_{\alpha=0.95} = 1.645$) and $\alpha = 0.99$ ($z_{\alpha=0.99} = 2.326$)

Experiments - 1

		$\sigma_t/\tilde{d}_t = 1/3$							
		$\alpha = 0.95$				$\alpha = 0.99$			
		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	5412	$\langle 102; 125\rangle$	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: 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)

Input data - 2

- We want to show that our approach is effective even when the planning horizon is significantly larger (up to 250 periods), and that the computation is not affected by the magnitude of the demands considered...

Input data - 2

- We want to show that our approach is effective even when the planning horizon is significantly larger (up to 250 periods), and that the computation is not affected by the magnitude of the demands considered...
- The expected period demands $\tilde{d}_t$ are generated as uniformly distributed random numbers in $[0, 10000]$
- fixed ordering cost $a \in \{2500, 5000, 10000, 20000\}$
- holding cost $h = 1$ per unit per period
- coefficient of variation $\sigma_t / \tilde{d}_t \in \{1/3, 1/6\}$
- The planning horizon length now ranges up to 250 periods
- service levels $\alpha = 0.95$ ($z_{\alpha=0.95} = 1.645$) and $\alpha = 0.99$ ($z_{\alpha=0.99} = 2.326$)

Experiments - 2

		$\sigma_t/\tilde{d}_t = 1/3$				$\sigma_t/\tilde{d}_t = 1/6$			
		$\alpha = 0.95$		$\alpha = 0.99$		$\alpha = 0.95$		$\alpha = 0.99$	
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

Literature Review

- A detailed discussion on the characteristics of (R,S) can be found in (de Kok [5])

Literature Review

- A detailed discussion on the characteristics of (R,S) can be found in (de Kok [5])
- (R^n,S^n) provides an effective means of reducing planning instability (deviations in planned orders, also known as *nervousness* [6, 8]) and coping with demand uncertainty

Literature Review

- A detailed discussion on the characteristics of (R,S) can be found in (de Kok [5])
- (R^n,S^n) provides an effective means of reducing planning instability (deviations in planned orders, also known as *nervousness* [6, 8]) and coping with demand uncertainty
- 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 [12]

Literature Review

- A detailed discussion on the characteristics of (R,S) can be found in (de Kok [5])
- (R^n,S^n) provides an effective means of reducing planning instability (deviations in planned orders, also known as *nervousness* [6, 8]) and coping with demand uncertainty
- 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 [12]
- As Silver points out computing replenishment cycle policy parameters under non-stationary stochastic demand is a computationally hard task [11]

Literature Review

- A detailed discussion on the characteristics of (R,S) can be found in (de Kok [5])
- (R^n,S^n) provides an effective means of reducing planning instability (deviations in planned orders, also known as *nervousness* [6, 8]) and coping with demand uncertainty
- 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 [12]
- As Silver points out computing replenishment cycle policy parameters under non-stationary stochastic demand is a computationally hard task [11]
- Early works in the area were heuristic (Silver [11], Bookbinder and Tan [3])

Literature Review

- A detailed discussion on the characteristics of (R,S) can be found in (de Kok [5])
- (R^n,S^n) provides an effective means of reducing planning instability (deviations in planned orders, also known as *nervousness* [6, 8]) and coping with demand uncertainty
- 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 [12]
- As Silver points out computing replenishment cycle policy parameters under non-stationary stochastic demand is a computationally hard task [11]
- Early works in the area were heuristic (Silver [11], Bookbinder and Tan [3])
- The first complete approach for computing (R,S) policy parameters was proposed by Tarim and Kingsman [15] in the form of a MIP model

Literature Review

- A detailed discussion on the characteristics of (R,S) can be found in (de Kok [5])
- (R^n,S^n) provides an effective means of reducing planning instability (deviations in planned orders, also known as *nervousness* [6, 8]) and coping with demand uncertainty
- 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 [12]
- As Silver points out computing replenishment cycle policy parameters under non-stationary stochastic demand is a computationally hard task [11]
- Early works in the area were heuristic (Silver [11], Bookbinder and Tan [3])
- The first complete approach for computing (R,S) policy parameters was proposed by Tarim and Kingsman [15] in the form of a MIP model
- Tempelmeier extended this MIP formulation in order to consider different service level measures [17]

Literature Review

- A detailed discussion on the characteristics of (R,S) can be found in (de Kok [5])
- (R^n,S^n) provides an effective means of reducing planning instability (deviations in planned orders, also known as *nervousness* [6, 8]) and coping with demand uncertainty
- 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 [12]
- As Silver points out computing replenishment cycle policy parameters under non-stationary stochastic demand is a computationally hard task [11]
- Early works in the area were heuristic (Silver [11], Bookbinder and Tan [3])
- The first complete approach for computing (R,S) policy parameters was proposed by Tarim and Kingsman [15] in the form of a MIP model
- Tempelmeier extended this MIP formulation in order to consider different service level measures [17]
- Tarim et al. proposed an efficient CP approach for solving large instances [14]

Literature Review

- A detailed discussion on the characteristics of (R,S) can be found in (de Kok [5])
- (R^n,S^n) provides an effective means of reducing planning instability (deviations in planned orders, also known as *nervousness* [6, 8]) and coping with demand uncertainty
- 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 [12]
- As Silver points out computing replenishment cycle policy parameters under non-stationary stochastic demand is a computationally hard task [11]
- Early works in the area were heuristic (Silver [11], Bookbinder and Tan [3])
- The first complete approach for computing (R,S) policy parameters was proposed by Tarim and Kingsman [15] in the form of a MIP model
- Tempelmeier extended this MIP formulation in order to consider different service level measures [17]
- Tarim et al. proposed an efficient CP approach for solving large instances [14]
- Pujawan and Silver recently proposed a novel and effective heuristic approach [9]

Conclusions

- Our model is based on the deterministic equivalent formulation originally proposed by Tarim and Kingsman [15]

Conclusions

- Our model is based on the deterministic equivalent formulation originally proposed by Tarim and Kingsman [15]
- Our DP reformulation performs significantly better than the original MIP approach proposed by Tarim and Kingsman

Conclusions

- Our model is based on the deterministic equivalent formulation originally proposed by Tarim and Kingsman [15]
- Our DP reformulation performs significantly better than the original MIP approach proposed by Tarim and Kingsman
- It also beats the state-of-the-art reformulations proposed by Tarim and Smith [16] and by Tarim et al. [14]

Conclusions

- Our model is based on the deterministic equivalent formulation originally proposed by Tarim and Kingsman [15]
- Our DP reformulation performs significantly better than the original MIP approach proposed by Tarim and Kingsman
- It also beats the state-of-the-art reformulations proposed by Tarim and Smith [16] and by Tarim et al. [14]
- The algorithm we propose is based on the known concept of SSR, originally introduced by Christofides et al. [4]

Conclusions

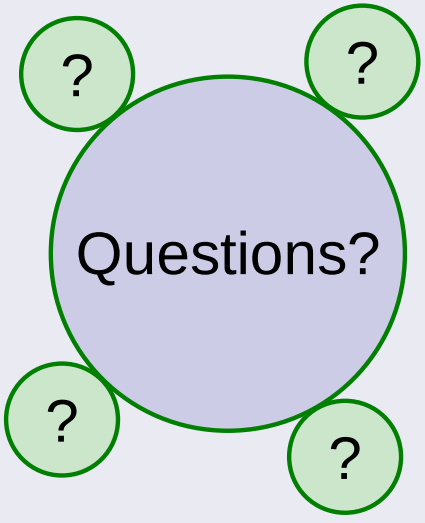
- Our model is based on the deterministic equivalent formulation originally proposed by Tarim and Kingsman [15]
- Our DP reformulation performs significantly better than the original MIP approach proposed by Tarim and Kingsman
- It also beats the state-of-the-art reformulations proposed by Tarim and Smith [16] and by Tarim et al. [14]
- The algorithm we propose is based on the known concept of SSR, originally introduced by Christofides et al. [4]
- The concept of *state space augmentation* [2] is also known in the Operations Research literature. A dual approach to state space augmentation is *decremental SSR* (Righini & Salani [10])

Conclusions

- Our model is based on the deterministic equivalent formulation originally proposed by Tarim and Kingsman [15]
- Our DP reformulation performs significantly better than the original MIP approach proposed by Tarim and Kingsman
- It also beats the state-of-the-art reformulations proposed by Tarim and Smith [16] and by Tarim et al. [14]
- The algorithm we propose is based on the known concept of SSR, originally introduced by Christofides et al. [4]
- The concept of *state space augmentation* [2] is also known in the Operations Research literature. A dual approach to state space augmentation is *decremental SSR* (Righini & Salani [10])
- Nevertheless, the idea of filtering a relaxed state space graph is, to the best of our knowledge, a novel contribution.

The End

The End



Acknowledgments: this work was supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain Research (CTVR) and Grant No. 05/IN/I886. S. Armagan Tarim and Brahim Hnich are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027

R. E. Bellman.
Dynamic Programming.
Princeton University Press, Princeton, NJ, 1957.

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.

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

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

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.

T. de Kok and K. Inderfurth.
Nervousness in inventory management: Comparison of basic control rules.
European Journal of Operational Research, 103:55–82, 1997.

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.

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

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.

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

E. A. Silver.

Inventory control under a probabilistic time-varying demand pattern.
AIIE Transactions, 10:371–379, 1978.

C. S. Tang.
Perpectives in supply chain risk management.
International Journal of Production Economics, 103:451–488, 2006.

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

S. A. Tarim, B. Hnich, R. Rossi, and S. Prestwich.
Cost-based filtering techniques for stochastic inventory control under service level constraints.
Constraints, x(x):xxx–xxx, 2009.

S. A. Tarim and B. G. Kingsman.

The stochastic dynamic production/inventory lot-sizing problem with service-level constraints.
International Journal of Production Economics, 88:105–119, 2004.

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

H. Tempelmeier.
On the stochastic uncapacitated dynamic single-item lotsizing problem with service level constraints.
European Journal of Operational Research, 127(1):184–194, August 2007.