Tutorial:
System-Oriented Inventory Models for Spare Parts

Geert-Jan van Houtum
Seminal paper:

Two important books:
This tutorial

Based on:

In progress:
CONTENTS

1. Introduction
2. Real-life networks
3. Single-location model with backordering
4. Single-location model with emergency shipments
5. METRIC model
6. Multi-location model with lateral and emergency shipments
7. Extensions
8. Applications in practice
9. Challenges for further research
1. Introduction
Total Cost of Ownership (TCO):
The total costs during the whole life cycle (perspective: user of the system)
TCO for an example system

- Acquisition costs
- Maintenance costs
- Downtime costs

Graph showing the percentage contributions of each cost category.
Size of maintenance industry

- Worldwide revenues after-sales services: 1500 billion US Dollars per year (AberdeenGroup [2003])
- Sales of spare parts and services in US: 8% of GNP (AberdeenGroup [2003])
- Manufacturers in US, Europe and Asia generate 26% of their revenues via services (Deloitte [2006])
- At an airline, 10% of all costs is constituted by maintenance costs (Lam [1995])
Long term trends

- Maintenance of complex systems becomes too complicated for users themselves
- Users require higher availabilities (less downtime)
- Users look at TCO

- Maintenance is outsourced to third party or OEM (pooling resources, pooling data, remote monitoring)
- More extreme: One sells function plus availability
- Feedback to design (better systems, higher sustainability)
Research topics

- **Spare parts management**
- Condition based maintenance
- Inventory models for spare parts and service tools
- Scheduling of service engineers
- Design of spare parts networks
- Forecasting of failures
- The effect of remote monitoring and diagnostics on total costs
- Service contracts and customer differentiation
- The effect of design decisions for new systems on their Total Cost of Ownership
- New business models for collaboration between users
- Game-theoretic models on the relationship between OEM-s, third parties and users
- ...
Spare parts inventory models:
• for critical components
• of advanced capital goods
• with service level constraints for system-oriented service measures such as system availability or aggregate fill rate

Application in practice: Tactical planning level!
2. Real-life networks
Network ASML

60 warehouses
5000 SKU’s
Network ASML (cont.)

1. Normal delivery: 2 hrs.
2. Lateral transshipment: 14 hrs.
Network IBM (for next day deliveries)
Network Nedtrain

- Lateral supply out of QRS < 2 hours
- Regular replenishment 2-5/week
- Urgent orders < 24 hrs
## User networks vs. OEM networks

<table>
<thead>
<tr>
<th>User networks (typical for military systems)</th>
<th>OEM networks (typical for high-tech systems)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preventive maintenance dominates</td>
<td>Corrective maintenance dominates</td>
</tr>
<tr>
<td>Two echelon levels in one region</td>
<td>Global network with two echelon levels</td>
</tr>
<tr>
<td>No emergency option</td>
<td>Emergency option at highest echelon level</td>
</tr>
<tr>
<td>Repairs at own repair shops</td>
<td>Repairs at original equipment manufacturers</td>
</tr>
<tr>
<td>Relatively loose service targets</td>
<td>Strict/high service targets</td>
</tr>
</tbody>
</table>
Examples of users maintaining their own system

<table>
<thead>
<tr>
<th>User</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>An electric utility company</td>
<td>[23]</td>
</tr>
<tr>
<td>Italian airports</td>
<td>[24]</td>
</tr>
<tr>
<td>KLM engineering &amp; maintenance</td>
<td>Our experience (e.g., [25])</td>
</tr>
<tr>
<td>NedTrain</td>
<td>Our experience (e.g., [26])</td>
</tr>
<tr>
<td>Italian paper-making industry</td>
<td>[27]</td>
</tr>
<tr>
<td>Royal Netherlands Navy</td>
<td>[28,29,7]</td>
</tr>
<tr>
<td>US Air Force</td>
<td>[16,18,19]</td>
</tr>
<tr>
<td>US Coast Guard</td>
<td>[3]</td>
</tr>
</tbody>
</table>
Examples of OEMs maintaining sold system

<table>
<thead>
<tr>
<th>OEM</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASML</td>
<td>[1,34,35]</td>
</tr>
<tr>
<td>Cisco</td>
<td>Our experience (e.g., [36])</td>
</tr>
<tr>
<td>IBM</td>
<td>[37–39]</td>
</tr>
<tr>
<td>Océ</td>
<td>Our experience (e.g., [40,41])</td>
</tr>
<tr>
<td>Teradyne</td>
<td>[42]</td>
</tr>
<tr>
<td>Vanderlande Industries</td>
<td>Our experience (e.g., [43])</td>
</tr>
<tr>
<td>Volvo Parts Corporation</td>
<td>[44]</td>
</tr>
</tbody>
</table>
3. Single-location model with backordering

(cf. Chapter 2 of Sherbrooke [1992], Single-item version: Feeney and Sherbrooke [1966])

3.1 Model description
3.2 Overview of assumptions
3.3 Evaluation
3.4 Optimization
3.5 Alternative optimization techniques and service measures
3.1. Model description
Model

- Warehouse
  - Demands (Poisson Streams)
  - Ready-for-use parts
  - Failed parts

- Repair shop
- Installed base

 TU/e Technische Universiteit Eindhoven University of Technology
Model (cont.)

- We use the terminology that is common for repairable LRU’s (Line Replaceble Units). The model is also applicable for consumable LRU’s.
- LRU’s are denoted as SKU’s (Stock-Keeping Units) in this presentation.
- We have an infinite time horizon.
- We look at the buy of the initial stock of spare parts for all SKU’s.
- No emergency shipments.
Input variables:

- $I$ : Set of SKU's, SKU's are numbered 1,..., $|I|$
- $m_i$ : Demand rate for SKU $i$ ($m_i \geq 0$)
- $M = \sum_{i \in I} m_i$ : Total demand rate ($M > 0$)
- $t_i$ : Mean repair leadtime for SKU $i$ ($t_i > 0$)
- $c_i^h$ : Price of a part of SKU $i$ ($c_i^h > 0$)
- $EBO^{obj}$ : Target level for aggregate mean number of backorders
**Decision variables:**

- $S_i : \text{Basestock level for SKU } i \ (S_i \in \{0,1,...\})$
- $S = (S_1, S_2, ..., S_{|I|}) : \text{Vector with all basestock levels denotes a solution}$
Output variables:

- \( C_i(S_i) = c_i^h S_i \): Inventory holding costs for spare parts of SKU \( i \)
- \( C(S) = \sum_{i \in I} C_i(S_i) = \sum_{i \in I} c_i^h S_i \): Total inventory holding costs
- \( EBO_i(S_i) \): Mean number of backorders of SKU \( i \)
- \( EBO(S) = \sum_{i \in I} EBO_i(S_i) \): Aggregate mean number of backorders
Problem (P):

\[ \min C(S) \]

subject to

\[ EBO(S) \leq EBO^{\text{obj}} \]

\[ S_i \in \{0,1,\ldots\} \text{ for all } i \in I \]

Relation with availability:

\[ \text{Availability}(S) \approx 1 - \left( EBO(S) / N \right) \]

where \( N \) is the number of machines
3.2. Overview of assumptions
“1. Demands for the different SKU’s occur according to independent Poisson processes”

- A failure of a component does not lead to additional failures of other components
- Assumption of Poisson demand processes is justified when:
  - lifetimes of components are exponential, or
  - lifetimes are generally distributed and number of machines is sufficiently large (a merge of many renewal processes gives a process that is close to Poisson)
“ 2. For each SKU, the demand rate is constant ”

• Justified in case fraction of machines that is down, is always sufficiently small:
  - either because downtimes are short in general
  - or downtimes occur only rarely

“ 3. Repair leadtimes for different SKU’s are independent and repair leadtimes of the same SKU are i.i.d. ”

• See repair leadtimes as planned repair leadtimes
"4. A one-for-one replenishment strategy is applied for all SKU’s"

- Justified in case:
  - there are no fixed ordering costs at all, or
  - fixed ordering costs are small relative to the prices of the SKU’s
3.3. Evaluation
Evaluation can be done per SKU

Extra notation per SKU $i$:

- $X_i(t)$: number of parts in repair at time $t$
- $I_i(t,S_i)$: number of parts on hand at time $t$
- $B_i(t,S_i)$: number of backordered demands at time $t$
- $X_i, I_i(t,S_i), B_i(t,S_i)$: corresponding steady-state variables
Petri net of repair and demand fulfilment process
Possible states

States that can occur for \((X_i(t), I_i(t, S_i), B_i(t, S_i))\):

- \((0, S_i, 0)\) : Nothing in repair, \(S_i\) good parts on stock
- \((1, S_i-1, 0)\) : 1 part in repair, \(S_i-1\) good parts on stock
- \((S_i-1, 1, 0)\) : \(S_i-1\) parts in repair, 1 good part on stock
- \((S_i, 0, 0)\) : \(S_i\) parts in repair, no parts on stock anymore
- \((S_i+1, 0, 1)\) : \(S_i+1\) in repair, no stock, 1 request in backlog
Equations

- \( I_i(t, S_i) = (S_i - X_i(t))^+ \)
- \( B_i(t, S_i) = (X_i(t) - S_i)^+ \)
- Stock balance equation:
  \[
  X_i(t) + I_i(t, S_i) - B_i(t, S_i) = S_i
  \]

And thus also:

- \( I_i(S_i) = (S_i - X_i)^+ \)
- \( B_i(S_i) = (X_i - S_i)^+ \)
- \( X_i + I_i(S_i) - B_i(S_i) = S_i \)
Palm's Theorem (cf. Palm [1938]):

If at a certain unit the arrival process of jobs is Poisson with rate $\lambda$ and if the leadtimes for the jobs are independent and identically distributed random variables corresponding to any distribution with mean $EW$, then the steady state probability distribution for the number of jobs present in that unit is a Poisson distribution with mean $\lambda EW$.

• Developed for an $M|G|\infty$ queue
• Result is easily seen for deterministic leadtimes
• Generalizes Little’s law for this particular system
Lemma 3.1. Let $i \in I$.

(i) The pipeline $X_i$ is Poisson distributed with mean $m_i t_i$, i.e.:

$$\mathbb{P} \{X_i = x\} = \frac{(m_i t_i)^x}{x!} e^{-m_i t_i}, \quad \forall x \in \mathbb{N}_0.$$ 

(ii) The distribution of the stock on hand $I_i(S_i)$ is given by:

$$\mathbb{P} \{I_i(S_i) = x\} = \begin{cases} 
\sum_{y=S_i}^{\infty} \mathbb{P} \{X_i = y\} & \text{if } x = 0; \\
\mathbb{P} \{X_i = S_i - x\} & \text{if } x \in \{1, \ldots, S_i\}.
\end{cases}$$

(iii) The distribution of the number of backordered demands $B_i(S_i)$ is given by:

$$\mathbb{P} \{B_i(S_i) = x\} = \begin{cases} 
\sum_{y=0}^{S_i} \mathbb{P} \{X_i = y\} & \text{if } x = 0; \\
\mathbb{P} \{X_i = x + S_i\} & \text{if } x \in \mathbb{N}.
\end{cases}$$
Last step

\[ EBO_i(S_i) = \mathbb{E}B_i(S_i) = \sum_{x=S_i+1} (x - S_i) \mathbb{P}\{X_i = x\} \]

\[ = m_i t_i - S_i + \sum_{x=0}^{S_i} (S_i - x) \mathbb{P}\{X_i = x\}, \quad \forall S_i \in \mathbb{N}_0. \]
3.4. Optimization
Problem (P):

\[
\begin{align*}
\text{min} & \quad C(S) \\
\text{subject to} & \quad EBO(S) \leq EBO^{\text{obj}} \\
& \quad S_i \in \{0,1,\ldots\} \text{ for all } i \in I
\end{align*}
\]

Kind of knapsack problem; hard to solve
Problem (Q):

\[ \min C(S) = \sum_{i \in I} c_i^h S_i \]
\[ \min EBO(S) = \sum_{i \in I} EBO_i(S_i) \]
subject to
\[ S \in \{(S_1, S_2, \ldots, S_{|I|}) | S_i \in \{0, 1, \ldots\} \text{ for all } i\} \]

⇒ Multi-objective programming problem

We derive so-called efficient solutions
Example 1

Input variables for Problem (Q):
- $|I| = 3$
- $m_1 = 15, m_2 = 5, m_3 = 1, M = 21$ demands/yr
- $t_1 = t_2 = t_3 = 1/6$ yrs
- $c_1^h = €1,000, c_2^h = €3,000, c_3^h = €20,000$

Extra input variable for Problem (P):
- $EBO^{obj} = 0.1$
**Example 1 (cont.)**

**Enumeration exercise:**
- Consider plausible values for the basestock levels
- Compute $C(S)$
- Compute $EBO(S)$
- Plot them in a $C(S)$ vs. $EBO(S)$ figure

<table>
<thead>
<tr>
<th>$S_{(1)}$</th>
<th>$S_{(2)}$</th>
<th>$S_{(3)}$</th>
<th>$C(S)$ (€)</th>
<th>$EBO(S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>3,500</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>2,582</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2.000</td>
<td>1,869</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3.000</td>
<td>1,413</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4.000</td>
<td>1,171</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5.000</td>
<td>1,062</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>6.000</td>
<td>1,020</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>7.000</td>
<td>1,006</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>8.000</td>
<td>1,002</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>9.000</td>
<td>1,000</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3.000</td>
<td>2,935</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4.000</td>
<td>2,017</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>5.000</td>
<td>1,304</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>6.000</td>
<td>0,848</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>7.000</td>
<td>0,605</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>8.000</td>
<td>0,497</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>9.000</td>
<td>0,455</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>10.000</td>
<td>0,440</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>10.000</td>
<td>0,440</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>10.000</td>
<td>0,440</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>10.000</td>
<td>0,440</td>
</tr>
</tbody>
</table>
Example 1 (cont.)

![Graph showing EBO(S) vs. C(S) (x € 1,000)]

- **EBO(S)**
- **C(S) (x € 1,000)**

Legend:
- **Enumeration**
- **Efficient solutions**
For Problem (Q), we obtain:

- Efficient solutions
- The whole efficient frontier
- Observation: An optimal solution for Problem (P) is efficient for Problem (Q), and vice versa

Optimal solution for Problem (P):

- \( S = (6, 2, 1) \)
- \( C(S) = € 32,000 \)
- \( EBO(S) = 0.098 \)
Problem (Q):

\[
\begin{align*}
\min & \quad C(S) = \sum_{i \in I} c_i^a S_i \\
\min & \quad EBO(S) = \sum_{i \in I} EBO_i(S_i) \\
\text{subject to} & \quad S \in \{(S_1, S_2, \ldots, S_{|I|}) \mid S_i \in \{0, 1, \ldots\} \text{ for all } i\}
\end{align*}
\]

=> Problem (Q) is separable (cf. Fox, 1966)

=> Efficient solutions via greedy algorithm
Greedy algorithm

An obvious efficient solution:

- \( S = \{0, 0, \ldots, 0\} \)
- \( C(S) = 0 \) => All other solutions: Higher costs
- \( EBO(S) = \sum_{i \in I} EBO_i(0) = \sum_{i \in I} m_i t_i \)
Greedy algorithm (cont.)

Next:
Look for the steepest decrease in $EBO(S)$ against the lowest increase in $C(S)$.
Generation of a next efficient solution:

- Current solution: $S$
- If $S_i$ would be increased with 1 unit:
  - Decrease for $EBO(S) = \Delta_i EBO(S) = \Delta EBO_i(S_i) = \ldots$ (see proof of Lemma 3.2)
  - Increase in $C(S) = \Delta_i C(S) = c_i^h$
  - $\Gamma_i := \Delta_i EBO(S) / \Delta_i C(S)$
- Pick the SKU with the largest $\Gamma_i$ (‘biggest bang for the buck’)

Greedy algorithm (cont.)
Greedy algorithm (cont.)

Algorithm 3.1 (Greedy Algorithm).

Step 1. $S_i := 0$ for all $i \in I$ (so $S := (0, \ldots, 0)$);
$\mathcal{E} := \{S\}$;
$C(S) := 0$ and $EBO(S) := \sum_{i \in I} m_i t_i$.

Step 2. $\Gamma_i := \frac{1}{c_i} \left(1 - \sum_{x=0}^{S_i} \mathbb{P}(X_i = x)\right)$ for all $i \in I$;
$k := \text{arg max}_{i \in I} \Gamma_i$;
$S := S + e_k$;
$\mathcal{E} := \mathcal{E} \cup \{S\}$.

Step 3. $C(S) := C(S) + c_k^h$;
$EBO(S) := EBO(S) - 1 + \sum_{x=0}^{S_k} \mathbb{P}(X_k = x)$;
If ‘stop criterium’, then stop, else go to step 2.
Lemma 3.3

Algorithm 3.1 generates efficient solutions for Problem (Q).

Proof: Via Fox (1966).
Alternative: Directly via the logic that always the most negative slope is chosen for the curve formed by the generated solutions.
Application of greedy algorithm:

<table>
<thead>
<tr>
<th>iteration</th>
<th>$\Gamma_1$</th>
<th>$\Gamma_2$</th>
<th>$\Gamma_3$</th>
<th>$k$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$EBO(S)$</th>
<th>$C(S)$ (Euros)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$-\cdot 10^{-4}$</td>
<td>$-\cdot 10^{-4}$</td>
<td>$-\cdot 10^{-6}$</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3.500</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$9.10 \cdot 10^{-4}$</td>
<td>$1.80 \cdot 10^{-4}$</td>
<td>$7.68 \cdot 10^{-6}$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2.582</td>
<td>1,000</td>
</tr>
<tr>
<td>2</td>
<td>$7.13 \cdot 10^{-4}$</td>
<td>$1.80 \cdot 10^{-4}$</td>
<td>$7.68 \cdot 10^{-6}$</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1.869</td>
<td>2,000</td>
</tr>
<tr>
<td>3</td>
<td>$4.56 \cdot 10^{-4}$</td>
<td>$1.80 \cdot 10^{-4}$</td>
<td>$7.68 \cdot 10^{-6}$</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1.413</td>
<td>3,000</td>
</tr>
<tr>
<td>4</td>
<td>$2.42 \cdot 10^{-4}$</td>
<td>$1.80 \cdot 10^{-4}$</td>
<td>$7.68 \cdot 10^{-6}$</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1.171</td>
<td>4,000</td>
</tr>
<tr>
<td>5</td>
<td>$1.09 \cdot 10^{-4}$</td>
<td>$1.80 \cdot 10^{-4}$</td>
<td>$7.68 \cdot 10^{-6}$</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0.605</td>
<td>7,000</td>
</tr>
<tr>
<td>6</td>
<td>$1.09 \cdot 10^{-4}$</td>
<td>$6.77 \cdot 10^{-5}$</td>
<td>$7.68 \cdot 10^{-6}$</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0.497</td>
<td>8,000</td>
</tr>
<tr>
<td>7</td>
<td>$4.20 \cdot 10^{-5}$</td>
<td>$6.77 \cdot 10^{-5}$</td>
<td>$7.68 \cdot 10^{-6}$</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0.293</td>
<td>11,000</td>
</tr>
<tr>
<td>8</td>
<td>$4.20 \cdot 10^{-5}$</td>
<td>$1.74 \cdot 10^{-5}$</td>
<td>$7.68 \cdot 10^{-6}$</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0.251</td>
<td>12,000</td>
</tr>
<tr>
<td>9</td>
<td>$1.42 \cdot 10^{-5}$</td>
<td>$1.74 \cdot 10^{-5}$</td>
<td>$7.68 \cdot 10^{-6}$</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>0.199</td>
<td>15,000</td>
</tr>
<tr>
<td>10</td>
<td>$1.42 \cdot 10^{-5}$</td>
<td>$3.47 \cdot 10^{-6}$</td>
<td>$7.68 \cdot 10^{-6}$</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>0.185</td>
<td>16,000</td>
</tr>
<tr>
<td>11</td>
<td>$4.25 \cdot 10^{-6}$</td>
<td>$3.47 \cdot 10^{-6}$</td>
<td>$7.68 \cdot 10^{-6}$</td>
<td>3</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>0.031</td>
<td>36,000</td>
</tr>
</tbody>
</table>
Example 1 (cont.)

We obtain a subset of all efficient solutions!
**Example 1 (cont.)**

*Via greedy algorithm:*

- **Heuristic** solution for Problem (P) with $EBO_{obj} = 0.1$:
  - $S = (7, 3, 1)$, $C(S) = € 36,000$, $EBO(S) = 0.031$
  - Optimality gap = $(36000-33000)/33000 = 9.1\%$

- **Optimal** solution for Problem (P) with $EBO_{obj} = 0.031$:
  - $S = (7, 3, 1)$, $C(S) = € 36,000$, $EBO(S) = 0.031$
Example 2

Real-life data, 99 SKU’s
Example 2 (cont.)

=> Smooth line, good heuristic solutions for Problem (P)
Conclusions w.r.t. greedy algorithm

- Generates efficient solutions for Problem (Q)
- Does not generate all efficient solutions, but a certain subset of solutions
- This subset has a certain robustness: small changes in the input parameters (e.g., demand rates!) lead to small changes in the subset of generated solutions
- Leads to a heuristic solution for Problem (P). **Conjecture:** This solution is robust. (Notice: This does not hold for the exact solution of Problem (P).)
Conclusions w.r.t. greedy algorithm (cont.)

- Heuristic solution is optimal for Problem (P) for some specific values of $EBO^{obj}$
- Generally, the heuristic solution for Problem (P) will be good when one has many SKU’s
- Simple and easy to understand
- Easy to implement in practice
- Requires little computational effort
3.5. Alternative optimization techniques and service measures
Alternative techniques:
• Langrange relaxation
• Dantzig-Wolfe decomposition
-> Both give the same efficient solutions as greedy algorithm

Alternative service measures:
• Aggregate mean waiting time
• Aggregate fill rate
• …
-> Works as long as you have the required convexity properties
Why is robustness of the heuristic solutions important?

Spare parts planning in practice:

- Executed every 3 months, say
- Per planning moment:
  - Generation of new forecasts for the demand rates of all SKU’s
  - Application of the greedy heuristic
- Implementation of new solution:
  - Both increasing and decreasing the stock of a SKU gives some costs
Open problems

- Show/prove that the greedy heuristic leads to robust solutions

- Similarly for Dantzig-Wolfe decomposition (may be easier when we go to dimension 3 or higher)

- How to use of the model in a rolling horizon setting such that basestock levels do not change too much
4. Single-location model with emergency shipments
Assumption for basic model:
If a demand cannot be immediately fulfilled from stock, then the demand is backordered.

In several practical situations:
- Downtimes of machines are very expensive.
- In case of a stockout, a demand will be satisfied in an alternative way, i.e., via a fast repair procedure or via an emergency shipment.
- 'lost sales' instead of 'backordering'
Changes in model assumptions

- $t_{i}^{em}$: Average time for an emergency shipment for SKU $i$
- $c_{i}^{em}$: Cost of an emergency shipment for SKU $i$, minus cost of a normal repair ($c_{i}^{em} \geq 0$)
- $c_{i}^{h}$: Inventory holding cost per time unit per part of SKU $i$ ($c_{i}^{h} > 0$)
- $W_i(S_i)$: Mean waiting time for a demand for SKU $i$
- $W(S) = \Sigma_{i \in I} (m_i/M) W_i(S_i)$: Aggregate mean waiting time for an arbitrary demand for all SKU’s together
• \( W^{\text{obj}} \): Target level for \( W(S) \)

• Problem formulation:

\[
\begin{align*}
(P'') & \quad \min \quad \hat{C}(S) \\
\text{subject to} \quad W(S) & \leq W^{\text{obj}}, \\
S & \in \mathcal{S}.
\end{align*}
\]

• Link with availability:

\[
A(S) \approx 1 - \frac{MW(S)}{Z}.
\]
Evaluation

- Total costs:

\[ \hat{C}(S) = \sum_{i \in I} \hat{C}_i(S_i). \]

\[ \hat{C}_i(S_i) = c_i h S_i + m_i (1 - \beta_i(S_i)) c_i^{\text{em}}, \]

\[ W(S) = \sum_{i \in I} \frac{m_i}{M} W_i(S_i). \]

\[ W_i(S_i) = (1 - \beta_i(S_i)) t_i^{\text{em}}, \]

- Formula for \( W(S) \):

- Expression for fill rate \( \beta_i(S_i) \) (steady-state behavior is equivalent to behavior in Erlang loss system):

\[ \beta_i(S_i) = 1 - \frac{1}{S_i} \rho_i^{S_i} \left( \sum_{j=0}^{S_i} \frac{1}{j!} \rho_i^j \right), \]

\[ \rho_i := m_i t_i. \]

Erlang loss probability
Optimization

• Switch to related multi-objective problem:

\[
(Q'') \quad \min \quad \hat{C}(S) \\
\min \quad W(S) \\
\text{subject to } S \in \mathcal{S}.
\]

• Karush (1957): Erlang loss probability is strictly convex and decreasing as a function of the number of servers
Hence:
- $\beta_i(S_i)$ is strictly concave and increasing
- $W_i(S_i)$ is strictly convex and decreasing
- $\hat{C}_i(S_i)$ is convex
- $\hat{C}_i(S_i)$ may now be decreasing for smaller values of $S_i$

Define: $S_{i,\text{min}} = \arg\min \hat{C}_i(S_i)$

Exclude solutions $S$ with $S_i \leq S_{i,\text{min}}$ for some $i$

Then the remaining problem can be solved with a greedy algorithm
Algorithm 2.3 (Greedy algorithm)

Step 1 \( S_{i,\text{min}} := \arg \min \hat{C}_i(S_i) \) for all \( i \in I \);
Set \( S_i := S_{i,\text{min}} \) for all \( i \in I \), and \( S = (S_{1,\text{min}}, \ldots, S_{|I|,\text{min}}) \);
\( \mathcal{E} := \{S\} \);
Compute \( \hat{C}(S) \) and \( W(S) \).

Step 2 \( \Gamma_i := \frac{-\Delta W_i(S_i)}{\Delta C_i(S_i)} \) for all \( i \in I \);
\( k := \arg \max \{\Gamma_i : i \in I\} \);
\( S := S + e_k \);
\( \mathcal{E} := \mathcal{E} \cup \{S\} \).

Step 3 Compute \( \hat{C}(S) \) and \( W(S) \);
If ’stop criterium’, then stop, else goto Step 2.
5. METRIC model
(cf. Sherbrooke [1968])

5.1 Model description
5.2 Evaluation
5.3 Optimization
5.1. Model description
Model

Repair of spare parts

Central Warehouse

Local Warehouse

0

1

\[ J_{loc} \]

Machines at customers

Machines at customers
Input variables:

- \( J^{\text{loc}} \): Set of Local Warehouses (LW's), LW's are numbered 1,..., \(|J^{\text{loc}}|\)
- 0: Index for Central Warehouse (CW)
- \( J \): Set of all warehouses, \( J = \{0\} \cup J^{\text{loc}} \)
- \( I \): Set of critical SKU's, SKU's are numbered 1,..., \(|I|\)
- \( m_{i,j} \): Demand rate for SKU \( i \) at LW \( j \) (\( m_{i,j} \geq 0 \))
Input variables (cont.):

- $t_{i,j}$: Order and ship time from CW to LW $j$ for SKU $i$ (these times are deterministic; $t_{i,j} > 0$)
- $t_{i,0}$: Mean repair leadtime for SKU $i$ (repair leadtimes are i.i.d.; $t_{i,0} > 0$)
- $c_i^h$: Holding cost rate per part of SKU $i$ ($c_i^h > 0$)
- $EBO_{i,j}^{obj}$: Target level for aggregate mean number of backorders at LW $j$
**Decision variables:**

- $S_{i,j}$: Basestock level for SKU $i$ at warehouse $j$ ($S_{i,j} \in \{0,1,\ldots\}$)
- $S_j = (S_{1,j}, \ldots, S_{|I|,j})$: Basestock vector for SKU $i$
- $S$: Matrix with all basestock levels
Output and other variables (cont.):  
- $EBO_{i,j}(S_{i,0}, S_{i,j})$: Mean number of backorders for SKU $i$ at LW $j$;  
- $EBO_j(S_0, S_j)$: Aggregate mean number of backorders at LW $j$  

$$EBO_j(S_0, S_j) = \sum_{i \in I} EBO_{i,j}(S_{i,0}, S_{i,j}).$$
Output and other variables (cont.):

- \( C(S) = \sum_{i \in I} \sum_{j \in J} c_i^h S_i \): Total average costs (excl. holding costs for pipeline stock)
Problem (R):

$$\min C(S)$$

subject to

$$EBO_j(S_0,S_j) \leq EBO_j^{\text{obj}}, \quad j \in J^{\text{loc}},$$

$$S_{i,j} \in \{0,1,...\} \quad \text{for all } i \in I \text{ and } j \in J$$

Remark: Sherbrooke [1968] had a constraint for sum of $EBO_j(S_0,S_j)$. Then a greedy procedure can be applied to generate efficient solutions (after convexification).
5.2. Evaluation
Exact Evaluation
(due to Graves [1985])

**Observation 1:**
Evaluation can be done per SKU $i$
(the SKU’s are only ‘connected’ via the aggregate mean waiting times)
Observation 2:
For each SKU $i \in I$:

- LW places replenishment orders at CW $j$ according to a Poisson process with rate $m_{i,j}$
- The total demand process at CW is a Poisson process with rate:

$$m_{i,0} = \sum_{j \in J_{loc}} m_{i,j}$$
Order in which LW’s place replenishment orders does not depend on the basestock levels!
Line of analysis – Variables that are determined:

- $I_{i,0}(S_{i,0})$: On-hand stock of SKU $i$ at CW (in steady state)
- $B_{i,0}(S_{i,0})$: Number of backorders for SKU $i$ at CW (in steady state)
- $B_{i,0}^{(j)}(S_{i,0})$: Number of backorders for SKU $i$ of LW $j$ at CW (in steady state)
- $I_{i,j}(S_{i,0},S_{i,j})$: On-hand stock of SKU $i$ at LW $j$ (in steady state)
- $B_{i,j}(S_{i,0},S_{i,j})$: Number of backorders for SKU $i$ at LW $j$ (in steady state)

Same variables with $E$ added $j$: mean value
Exact Evaluation (cont.)

**Given:** \( i \in I \)

**Define:**

\( Y_{i,j} \) : Poisson distributed random variable with mean \( m_{i,j} t_{i,j} \) for each warehouse \( j \)
Step 1:

- $X_{i,0}$: Number of parts in repair pipeline at CW
- $I_{i,0}(S_{i,0}) = (S_{i,0} - X_{i,0})^+$ and $B_{i,0}(S_{i,0}) = (X_{i,0} - S_{i,0})^+$

- Repair shop at CW is as a $M|G|\infty$ queue, and thus Palm's theorem may be applied
- Thus: $X_{i,0} = Y_{i,0}$
Step 2:

- Due to FCFS allocation:
  
  Each backordered demand at CW is a demand from LW \( j \) with probability \( \frac{m_{i,j}}{m_{i,0}} \)

- Hence:

\[
\mathbb{P}\left\{ B_{i,0}^{(j)}(S_{i,0}) = x \right\} = \sum_{y=x}^{\infty} \binom{y}{x} \left( \frac{m_{i,j}}{m_{i,0}} \right)^x \left( 1 - \frac{m_{i,j}}{m_{i,0}} \right)^{y-x} \mathbb{P}\left\{ B_{i,0}(S_{i,0}) = y \right\}.
\]
Step 3:
- $X_{i,j}(S_{i,0})$: Number of parts in replenishment pipeline at LW $j$
- $I_{i,j}(S_{i,0}, S_{i,j}) = (S_{i,j} - X_{i,j}(S_{i,0}))^+$ and $B_{i,j}(S_{i,0}, S_{i,j}) = (X_{i,j}(S_{i,0}) - S_{i,j})^+$
- Due to deterministic order and ship time:
  $X_{i,j}(S_{i,0})$ at time $t = B_{i,0}^{(j)}(S_{i,0})$ at time $t - t_{i,j}$
  + Demand in interval $[t - t_{i,j}, t)$
- Hence: $X_{i,j}(S_{i,0}) = B_{i,0}^{(j)}(S_{i,0}) + Y_{i,j}$
Procedure:

- Compute first two moments of $I_{i,0}(S_{i,0})$ and $B_{i,0}(S_{i,0})$
- Compute first two moments of $B_{i,0}^{(j)}(S_{i,0})$ for each LW $j$: simple formulas available
- Compute first two moments of $X_{i,j}(S_{i,0})$ for each LW $j$
- Fit a negative Binomial distribution on first two moments of $X_{i,j}(S_{i,0})$ for each LW $j$
- Compute first moments of $I_{i,j}(S_{i,0},S_{i,j})$ and $B_{i,j}(S_{i,0},S_{i,j})$

=> Accurate and efficient approximation!
METRIC approach (cf. Sherbrooke [1985])

Procedure:

- Compute first moments of $I_{i,0}(S_{i,0})$ and $B_{i,0}(S_{i,0})$
- Compute first moment of $B_{i,0}^{(i)}(S_{i,0})$ for each LW $j$:
  \[
  \text{mean of } B_{i,0}^{(i)}(S_{i,0}) = \left( \frac{m_{i,j}}{m_{i,0}} \right) \times \left( \text{mean of } B_{i,0}(S_{i,0}) \right)
  \]
- Compute first moment of $X_{i,j}(S_{i,0})$ for each LW $j$
- Fit a Poisson distribution on first moment of $X_{i,j}(S_{i,0})$ for each LW $j$
- Compute first moments of $I_{i,j}(S_{i,0},S_{i,j})$ and $B_{i,j}(S_{i,0},S_{i,j})$

=> Not always accurate!
5.3. Optimization
Greedy heuristic
(cf. Wong et al. [2007])

Main idea:
• Start with zero stock for all SKU’s at all LW’s
• Use a steepest descent method to decrease costs
• Use a greedy logic to get to a feasible solution
Greedy heuristic (cont.)

• Distance of a solution $S$ to the set of feasible solutions:

$$\sum_{j \in J^{\text{loc}}} \left( EBO_j(S) - EBO_j^{\text{obj}} \right)^+.$$
Greedy heuristic (cont.)

- Decrease in distance to feasible solutions per unit of increase in costs:

\[ \Gamma_{i,j} = \Delta_{i,j} EBO / c_i^h. \]

\[ \Delta_{i,j} EBO = \sum_{l \in J^{loc}} \left[ \left( EBO_l(S) - EBO_l^{obj} \right)^+ \right] \]

\[ - \left( EBO_l(S + e_{i,j}) - EBO_l^{obj} \right)^+ \]
Greedy heuristic (cont.)

Algorithm 4.1 (Greedy Algorithm).

Step 1. $S_{i,j} := 0$ for all $i \in I$, $j \in J$ (so $S_j := (0, \ldots, 0)$ for all $j \in J$ and $S := (S_0, S_1, \ldots, S_{|J|})$);

$$C(S) := 0 \text{ and } EBO_j(S) := \sum_{i \in I} m_{i,j}(t_{i,0} + t_{i,j}) \text{ for all } j \in J^{loc}. $$

Step 2. $\Gamma_{i,j} := \frac{\Delta_{i,j} EBO}{c_i^h}$ for all $i \in I$, $j \in J$;

$$(k, l) := \arg \max_{(i,j) \in I \times J} \Gamma_{i,j};$$

$S := S + e_{k,l}.$

Step 3. $C(S) := C(S) + c_k^h$;

Calculate $EBO_j(S)$ for all $j \in J^{loc}$;

If $EBO_j(S) \leq EBO_j^{obj}$ for all $j \in J^{loc}$, then stop, else go to step 2.

Leads to: Heuristic solution (under exact evaluations)

No theoretical results like for single-location model
Dantzig-Wolfe decomposition leads to:
- Lower bound
- Heuristic solution

Performance of greedy heuristic:
- Small optimality gap
- \# SKU’s ↑ => Optimality gap ↓
- Computation time is relatively low
6. Multi-location model with lateral and emergency shipments
General setting

Key features:

- Multiple local warehouses (LW’s)
- Each LW supports one or more groups of machines
- Aggregate mean waiting time target per group
- Central warehouse: outside scope of model, is assumed to have infinite stock
- Multiple SKU’s
- All SKU’s are critical
- In case of stockout: application of lateral transshipment or emergency shipment
- Two types of LW’s: Main and regular LW’s
Two types of LW’s: main and regular LW’s

- Only main locals are suppliers for lateral transshipment
- This distinction is made to facilitate implementation in practice
- Pooling: full, partial, none
Output variables: (basic output)

Demand stream for SKU $i$ at LW $j$:

- $\beta_{i,j}(S_i)$: Fraction satisfied by LW $j$ itself
- $\alpha_{i,j,k}(S_i)$: Fraction satisfied by main LW $k$ via a lateral transshipment
- $A_{i,j}(S_i)$: Total fraction satisfied by a lat. transshipm., $A_{i,j}(S_i) = \sum_{k \in K \setminus \{j\}} \alpha_{i,j,k}(S_i)$
- $\theta_{i,j}(S_i)$: Fraction satisfied by an emerg. shipment

It holds that: $\beta_{i,j}(S_i) + A_{i,j}(S_i) + \theta_{i,j}(S_i) = 1$
Done per given SKU $i \in I$

Via Markov process:

- States $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,J})$, where $x_{i,j}$ denotes the on-hand stock at LW $i$
- Steady-state distribution $\{\pi(x_i)\}$ is determined numerically
- The $\beta_{i,j}(S_i)$, $\alpha_{i,j,k}(S_i)$, $\theta_{i,j}(S_i)$ follow from the $\pi(x_i)$
- Number of states: $\prod_{j \in J} (S_i + 1)$
Given: $J = \{1,2,3\}$, $K = \{1,2\}$, $k_3 = 1$, $S_{i,1} = 2$, $S_{i,2} = 1$, $S_{i,3} = 0$
We obtain:

\[ \beta_{i,1}(S_i) = \pi(1,0,0) + \pi(1,1,0) + \pi(2,0,0) + \pi(2,1,0), \]
\[ \alpha_{i,1,2}(S_i) = \pi(0,1,0), \quad \theta_{i,1}(S_i) = \pi(0,0,0), \quad \ldots \]

Principle: Decompose network into single-location models

- Normal delivery by A
- If A out of stock: lateral transshipment from B
- Then B observes extra demand: overflow demand
- Approximation: Overflow demand processes are Poisson processes
- Iterative procedure
Approximate evaluation

• Efficient and accurate

• One can apply the procedure to many structures for the lateral transshipments!

• In use at ASML since 2005
  • 60 stockpoints, 5000 SKU’s, say
  • Optimization: Greedy heuristic
7. Extensions
Multi-echelon, multi-indenture systems

- Exact recursion like for METRIC model
- Efficient and accurate appr. Eval. Based on two-moment fits
Batching

Easy to incorporate for:
- Single-location model with backordering
- METRIC model with (s,Q) rule at central warehouse and given Q’s

Hard to incorporate for models with emergency shipments
Multiple demand classes

Standard way: Critical levels

Drawbacks:
- More complicated optimization problem; no greedy heuristic available
- Critical levels are too strict for very low demand rates
- Critical levels may not be accepted in practice

Challenge: Find better ways for the differentiation when demand rates are low
8. Applications in practice
Advanced software packages:
• MCA Solutions: Founded by Morris Cohen & Associates
• Xelus
• Network Neighborhood: Developed by IBM
• …

Own developments:
• US Coast Guard: Development with Deshpande et al. [2006]
• ASML: Planning algorithm developed with TU/e
• …
9. Challenges
The full problem

Central Stockpoint

Supply spare parts

Reg. repl.: 1-2 weeks

Emergency Shipm.: 1-2 days

Local Stockpoint

Customers with contracts

Reg. repl.: 1-2 weeks

Lateral Shipments:
A few hours

Local Stockpoint

Customers with contracts

Direct sales
Dynamic decisions

Motivation:
• One has visibility of actual stocks
• Condition monitoring data
• Service contracts are for 1/2/3 years

Needed:
• Approximate dynamic programming, …
• Interaction between tactical and operational planning
## Example of condition monitoring data

<table>
<thead>
<tr>
<th>MACHINE NUMBER</th>
<th>TIMESTAMP</th>
<th>MACHINE TYPE</th>
<th>CUSTOMER ID</th>
<th>P955</th>
<th>P956</th>
<th>P957</th>
<th>P958</th>
<th>P959</th>
<th>P960</th>
<th>P961</th>
</tr>
</thead>
</table>
Further challenges

• Coupling with service tools planning

• Other performance measures: Long delays vs. Short delays

• Setting of leadtimes for procurement/repair

• Choice of transport modes