Decomposition for dynamic programming in production and inventory control
Wijngaard, J.

Published in:
Engineering and process economics

Published: 01/01/1979

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):
DECOMPOSITION FOR DYNAMIC PROGRAMMING IN PRODUCTION AND INVENTORY CONTROL

JACOB WIJNGAARD

Eindhoven University of Technology
Department of Industrial Engineering
P. O. Box 513
Eindhoven, The Netherlands

INTRODUCTION

Straightforward application of dynamic programming on production planning is impossible in general by the dimensionality of the problem. A calculation of the state values is only possible if the dimensionality can be reduced, for instance by decomposition. A lot of production planning problems are almost decomposable. A system of n inventory points, for instance, with one supplying production unit gives an (at least) n-dimensional dynamic programming problem. But if the production capacity is infinite the problem reduces to n one-dimensional problems. A system of n inventory points in series, with constrained production capacities in between, also gives an n-dimensional problem. But if the production costs are linear and the production capacities infinite then the problem reduces to n one-dimensional problems (see Clark and Scarf [2]).

In this paper we will show with aid of two examples how this almost decomposability can facilitate the application of dynamic programming. The first example is a system with two parallel inventory points. The methods used in this case are closely related to the methods used by van Beek [1] in a multi-dimensional production smoothing problem. The second example is a two-echelon problem.

TWO PRODUCTS WITH A JOINT PRODUCTION CAPACITY

The first system considered is the following

There are two inventory points, both supplied by the same production unit P. \( \phi_1 \) and \( \phi_2 \) are the demand density functions for 1 and 2. Production orders are placed at the end of each period and are assumed to arrive during the next period. If \( x_1 \) is the inventory on hand at point 1 at the beginning of a period and if an order of \( z_1 \) units is placed then the expected inventory and stockout cost during that period is \( L_1(x_1 + z_1) \). The same for point 2.

The ordering costs are given by the functions \( c_1(\cdot) \) and \( c_2(\cdot) \). The production capacity of P is \( C \) units per period, that means \( z_1 + z_2 \leq C \). Unsatisfied demand is backlogged.

The discounted dynamic programming model for this problem is

\[
V^*_k(x_1, x_2) = \min_{z_1 + z_2 \leq C, z_1 \geq 0, z_2 \geq 0} \{ c_1(z_1) + c_2(z_2) \\
+ L_1(x_1 + z_1) + L_2(x_2 + z_2) \\
+ \sum_{d_1, d_2} v^{k-1}(x_1 + z_1 - d_1, x_2 + z_2 - d_2) \phi_1(d_1) \phi_2(d_2) \}
\]

The common production capacity causes the two-dimensionality of the problem. If \( C \) is large enough, the problem reduces to two one-dimensional problems. We will sketch a method to use this almost decomposability in the construction of optimal and good strategies.

Step 1. Solve the one-dimensional problems
The dynamic programming models for these problems are

$$
\begin{align*}
\phi_a^{1k}(x_1) &= \min_{0 \leq z_1 \leq C} \{c_1(z_1) + L_1(x_1+z_1) \\
&\quad + a \sum_{d_1} v_{a}^{1k-1}(x_1+z_1-d_1) \phi_{a}(d_1) \} \\
\phi_a^{2k}(x_1) &= \min_{0 \leq z_2 \leq C} \{c_2(z_2) + L_2(x_2+z_2) \\
&\quad + a \sum_{d_2} v_{a}^{2k-1}(x_2+z_2-d_2) \phi_{a}(d_2) \}
\end{align*}
$$

Step 2. Define the two-dimensional function $\phi_a^{0}(\cdot, \cdot)$ by

$$
\phi_a^{0}(x_1, x_2) = \min_{z_1, z_2} \{c_1(z_1) + c_2(z_2) \\
+ L_1(x_1+z_1) + L_2(x_2+z_2) \\
+ a \sum_{d_1} v_{a}^{0}(x_1+z_1-d_1) \phi_a^{1}(d_1) \\
+ a \sum_{d_2} v_{a}^{0}(x_2+z_2-d_2) \phi_a^{2}(d_2) \}
$$

This yields a feasible strategy $(f_0)$ for the two-dimensional problem. (Van Beek [1] used this way to construct a strategy for a multi-dimensional production smoothing problem.)

Step 3. Use $\phi_a^{1}(x_1, x_2)$ as starting vector in the two-dimensional dynamic iteration procedure.

Now there are two interesting questions

a. The quality of the strategy $f_0$

b. The number of iterations needed starting with $\phi_a^{1}(\cdot, \cdot)$ compared with the number of iterations needed starting with the null vector.

With respect to practical problems the first question is the most important one of course. In large production networks it is impossible in general to execute more than one iteration step and even that first step can be difficult.

The second question has to be seen in the framework of numerical methods of dynamic programming. The point is: how can one use the structure of the problem to reduce the computation time?

We have tried to answer these questions with aid of some numerical examples.

<table>
<thead>
<tr>
<th>$d_1$</th>
<th>$d_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.55</td>
</tr>
<tr>
<td>0.1</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Inventory cost (linear) 1

Stockout cost (linear) 3

Set-up cost: varying from 1 ... 10

The inventory levels are restricted to the points -3 ... +6.

We considered the ratios $r_1$ and $r_2$, where $r_1$ is the ratio of the costs under strategy $f_0$ and the optimal costs $r_1 = \frac{\text{costs}(f_0)}{\text{opt. costs}}$, and $r_2$ is the ratio of the number of iterations needed starting with $\phi_a^{1}(\cdot, \cdot)$ and the number of iterations starting with the null vector. For all values of the set-up costs we found values for $r_1$ of about 1.02 and for $r_2$ of about 1/2.

It has to be mentioned here that we did not use standard value iteration but the proce-
TWO ECHELON PROBLEMS

The results for the parallel case are rather hopeful. But if one wants to use this method for whole networks it is necessary to make it suitable for multi-echelon problems. Therefore, we consider here the most simple multi-echelon problem.

There are two echelons, echelon 2 supplying echelon 1 by a production unit P. The density function of the demand per period is \( \delta(\cdot) \). Unsatisfied demand is backlogged. Orders are placed at the beginning of each period and arrive at the beginning of the next period. The expected inventory and stockout cost in a period is \( L_1(x_1) + L_2(x_2) \) where \( x_1 \) is the inventory on hand at echelon 1 at the beginning of the period and \( x_2 \) is the total inventory in the system. The costs for orders from outside are given by the function \( c_1(\cdot) \), the costs of production by the function \( c_2(\cdot) \). The production capacity of P is \( C \) units per period.

The discounted dynamic programming model for this problem is

\[
\begin{align*}
V^k_a(x_1, x_2) &= L_1(x_1) + L_2(x_2) \\
&+ \min_{z_1 \geq 0, z_2 \geq 0} \left( c_1(z_1) + c_2(z_2) \right) \\
&+ \min_{z_1 < \min(C, x_2 - x_1)} \left( c_1(z_1) + c_2(z_2) \right) \\
&+ a \sum_{d} v^k_a(x_1 + z_1 - d, x_2 + z_2 - d) \delta(d)
\end{align*}
\]

(1)

For the case where \( C = \infty \), \( c_1(z) = c_1 \cdot z \) and \( L_1(\cdot) \) is convex. Clark and Scarf [2] have shown that \( v^k_a(x_1, x_2) = v^1_a(x_1) + v^2_k(x_2) \)

where \( v^1_a(\cdot) \) and \( v^2_a(\cdot) \) can be found in the following way

\[
\begin{align*}
v^1_a(x_1) &= \min_{z_1 \geq 0} \left( c_1 \cdot z_1 + L_1(x_1) \right) \\
&+ a \sum_{d} v^k_{a-1}(x_1 + z_1 - d) \delta(d)
\end{align*}
\]

(2)

\[
\begin{align*}
v^2_a(x_2) &= \min_{z_2 \geq 0} \left( c_2(z_2) + L_2(x_2) + p^k_a(x_2) \right) \\
&+ a \sum_{d} v^k_{a-1}(x_2 + z_2 - d) \delta(d)
\end{align*}
\]

(3)

and \( p^k_a(\cdot) \) is given by

\[
p^k_a(x_2) = 0 \quad \text{for} \quad x_2 > S_k
\]

\[
p^k_a(x_2) = \left( c_1 \cdot x_2 + \sum_{d} v^1_{a-1}(x_2 - d) \delta(d) \right) \\
&- \left( c_1 \cdot S_k + \sum_{d} v^1_{a-1}(S_k - d) \delta(d) \right)
\]

for \( x_2 \leq S_k \) and \( S_k \) is the point where the function \( c_1(\cdot) \) and \( L_1(\cdot) \) convex suggests that it is possible to use the same procedure as in the first problem for cases where the decomposition is not exact.

So the first step is the computation of \( v^{1\infty}_a(\cdot) \) and \( v^{2\infty}_a(\cdot) \) from the relations (2) and (3) adding of course the restriction \( z_1 \leq C \).

The second step is the construction of \( v^1_a(\cdot, \cdot) \) by \( v^0_a(x_1, x_2) = v^{1\infty}_a(x_1) + v^{2\infty}_a(x_2) \) and the construction of \( f^0_a \) by substituting \( v^0_a(\cdot) \) in the right hand side of relation (1).

Then we can make the same comparisons as in the first problem. We have worked out the following numerical examples.
Inventory cost linear, 1 for echelon 1, 1 for echelon 2.
Stockout cost linear, 3
Set-up cost varying (K)
Capacity is varying (C)
The inventory levels are restricted to the points -9 ... +9.

If the set-up costs are high we have to change the penalty function \( p \) somewhat.

In case the set-up costs are 0, the last echelon, echelon 1, wants to reorder each period up to a level \( S \). If \( x_2 < S \) this is not possible and the echelon 2 has to pay a penalty. But in case the set-up costs are positive the optimal strategy for echelon 1 is of the \( s-S \) type, with \( S-s > 0 \). If the difference between \( S \) and \( s \) is large echelon 1 does not order each period and it is not correct to penalize echelon 2 for having an inventory less than \( S \) if echelon 1 does not order.

To take into account this effect we replaced \( S \) in the definition of \( p \) by \( s + u \). We can try to choose \( u \) such that echelon 2 is only penalized if echelon 1 wants to order.

It is also possible to improve the results by excluding in the construction of the strategy \( f_0 \) possibilities which are certainly not optimal. For instance, if \( x_2 - x_1 - z_1 \geq C \) then it can never be optimal that \( z_2 > 0 \).

We have the following results.

<table>
<thead>
<tr>
<th>( \mu )</th>
<th>( C )</th>
<th>( K )</th>
<th>( r_1 )</th>
<th>( r_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 2 0</td>
<td>1.04</td>
<td>0.83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 3 10</td>
<td>1.11</td>
<td>0.33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 3 10</td>
<td>1.23</td>
<td>0.33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 3 20</td>
<td>1.10</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 4 10</td>
<td>1.16</td>
<td>0.33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 4 20</td>
<td>1.11</td>
<td>0.43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 ( \infty ) 10</td>
<td>1.18</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 ( \infty ) 20</td>
<td>1.26</td>
<td>0.67</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Some of the results are reasonable, some rather bad. The difference between \( \mu = 2 \) and \( \mu = 0 \) for the case \( C = 3, K = 10 \) suggests that it may be possible to get better results by choosing the penalty function in another way.

REFERENCES

