Aggregation and disaggregation in Markov decision models for inventory control
Veugen, L.M.M.; van der Wal, J.; Wessels, J.

Published: 01/01/1982

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.

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 14. Oct. 2018
Abstract.

In this paper the possibility is investigated of using aggregation in the action space for some Markov decision processes of inventory control type. For the standard \((s,S)\) inventory control model the policy improvement procedure can be executed in a very efficient way, therefore aggregation in the state space is not of much use. However, in situations where the decisions have some aftereffect and, hence, the old decision has to be incorporated in the state, it might be rewarding to aggregate actions. Some variants for aggregation and disaggregation are formulated and analyzed. Numerical evidence is presented.
1. Introduction.

For large scale Markov decision problems the possibilities for numerical analysis have increased considerably in the last 15 years, particularly via successive approximation methods. However, as for linear programming, it is generally acknowledged that for really large problems the only way is to exploit the specific structure of the problem at hand. The most apparent ways of exploiting the structure in successive approximation methods are:

(i) using the structure for a more efficient computation in each iteration step;

(ii) using the structure for obtaining a better convergence rate, thus diminishing the number of iteration steps.

(For a more elaborate discussion of these two ways, their effects and interference, cf. Hendrikx, van Nunen and Wessels [2]).

As has been shown in [2], it is quite well possible to exploit the specific structure in many problems of inventory or replacement type. Also periodicity can be treated very well, cf. [7], [8]. A general feeling says that — as in linear programming — very useful ways of using the problem structure could be provided by decomposition and by aggregation.

When applying successive approximation, the advantage of aggregation will primarily be of the first type, i.e. more efficient execution of the iteration step, possibly at the cost of an increase of the number of iteration steps.

In this paper the use of aggregation and subsequent disaggregation will be explored for inventory control models. In the literature there has been a lot of attention for the approximate value of aggregated Markov
decision models, cf. Whitt's papers [11] and also Mendelssohn [4]. That line of research has led to a general theory for the estimation of the difference between the exact and the approximative results. Specific results in this direction for inventory control models have been reported by Waldmann [9]. Recently, a new line of research has evolved in the form of iterative aggregation-disaggregation processes, cf. Mendelssohn [5] for Markov decision models and Schweitzer, Puterman and Kindle [6] for Markov reward models. This line of research heavily bears on (mainly Russian) developments for iterative aggregation-disaggregation in linear programming. It makes use of the linear programming formulation of the Markov decision model.

Here we will consider aggregation-disaggregation in the context of successive approximation procedures for Markov decision models, particularly inventory models. We will confine ourselves essentially to aggregation with respect to actions. What the term 'essentially' stands for will become clear in the sequel. The reason for the restriction to aggregation in the action space in this first exercise is that aggregation with respect to actions is very direct and does not require any remodelling: just leave out most of the actions. Aggregation in the state space on the other hand requires a new definition of the transition mechanism.

The simplicity of aggregation in the action space also facilitates the subsequent disaggregation and the evaluation of the results (possibly leading to the decision of a less crude aggregation).

Although aggregation in the action space is a very natural operation for inventory control models, it is also unclear whether it will help very much computationally, since, as [2] and [8] demonstrated, big action
spaces need not be a nuisance in analyzing inventory control models. In [2] it has been shown that the use of action elimination procedures may even be counterproductive if an efficient form of the value iteration procedure is applied. Moreover, if the inventory control model is such that an \((s, S)\) policy is optimal, then this property can be used to devise an extremely efficient policy iteration procedure, cf. Johnson [3]. So, in that case there is also not much need for aggregation.

Nevertheless, there are inventory control models in which a big action space is inconvenient. In particular one may expect that to be the case if there is a timelag between the moment the decisions are taken and the moment that they are executed. Then it might be necessary to incorporate the decisions of one or more previous periods in the state description. If this is so and if, moreover, one cannot restrict to \((s, S)\) policies, then one may hope that aggregation in the action space can be rather helpful.

In section 3 the aggregation-disaggregation approach, consisting of some variants, will be formulated. In section 4 the method is tested numerically on three examples. The first example is a standard inventory problem where \((s, S)\) policies are known to be optimal. Here, as expected, the aggregation-disaggregation method is not rewarding. The other two examples are models for the control of hard cash in a bank where there is a timelag between (part of) the control decision and its execution. The aggregation-disaggregation approach performs very well in the first model. At first sight it is somewhat surprising that the method does not work in the second model. Taking a closer look, however, reveals the cause of this relative failure and tells us how to make the method effective
also in the second model.

We think that having in mind the cash-flow regulation problem the aggregation-disaggregation approach is easier formulated and probably better understood than without that real-life background. For that reason we start in section 2 with a description of the cash-flow models.

2. Control of hard cash in a bank.

As examples we consider two variants of a problem concerning the control of cash in the local branch of a bank. In the branch office of the bank one is faced with the problem that on the one hand having a large amount of hard cash increases the loss of interest, whereas on the other hand having only a small amount of hard cash increases the risk of running out of stock. Moreover, frequent ordering or sending away of hard cash is also expensive. For a more elaborate model, see [10].

The total demand (positive or negative) for hard cash by customers is supposed to be random for each morning and each afternoon. The demand distributions are supposed to show only a daily pattern. So the demand pattern is cyclical with period 2. In each of the two model variants which will be considered, it is possible either to obtain extra hard cash from the regional office of the national reserve system or to deposit hard cash at this regional office, but only at the end of each morning. This is realized by ordering an armored car to come by. The differences between the two variants are in the way of ordering.

In the first model the armored car has to be ordered at the end of the preceding afternoon, if one wants it to come by at the end of the mor-
ning. The security regulations of the national reserve system require that also the amount of money to be brought by the car is specified the preceding afternoon. However, when the car arrives the cashier of the branch office may decide to take less of even to remove hard cash. So, in the afternoons it has to be decided whether the car should come next morning and if so with how much hard cash, whereas in the morning the exact amount of intake or go-out may be decided upon within the possibilities. When modelling this as a Markov decision process it is clear that for the afternoons the inventory level constitutes a sensible state concept. However, in the mornings one should also know what decision has been taken the preceding afternoon. So the state space for the mornings becomes 2-dimensional.

In the second model all decisions are taken at the end of each morning. The car has to be ordered a full day ahead and also the amount of money to be brought by the car has to be decided upon one day ahead. So every morning there are two decisions to be made. First, if a car has been sent for, it has to be decided how much cash to take in or to remove, and next one has to decide upon ordering a car for the next day. In the afternoons there are no decisions, so the afternoons could be eliminated by calculating full day transition probabilities from the half day ones and the day probabilities for running out of stock. In general this is not very efficient and we did not expect it to be efficient here. So we have not done this. In section 4 we will see that the structure of the problem forces us to reconsider our expectations. So in this second model the state will contain the predecision at all times, mornings and afternoons.
Some details about the problem:

The inventory level is measured on a discrete scale with 81 levels, 0,1,...,80. The number of demand levels for half a day is 40. The number of action levels when calling a car is on the average about 40, when the car arrives on the average about 60. Costs involved are inventory costs (interest losses), fixed costs for ordering and penalty costs for running out of stock.

These two models have large state spaces, even if we do not take into account the cyclical effect as is justified by [7,8]. Aggregation in the action space also diminishes the state space in these variants, since the second state dimension is an old decision. When ever, one may hope that in these cases aggregation in the action space will help in analyzing Markov decision problems more efficiently.

3. Aggregation-disaggregation in the action space.

Aggregation in the action space is very simple: just leave out part of the actions. Or more systematically, partition the action space into some subsets, choose a representative element from each set, leave out all other actions, and consider the problem with this thinned action space.

In inventory type problems a natural partitioning of the action set is obtained by dividing the action set into intervals. Mid- or endpoints of these intervals may serve as natural representatives. For a suitable choice of intervals and representatives the aggregated problem will be an inventory problem with ordering in batches.
For an \((s,S)\) inventory model with action set \(\{0,1,2,\ldots\}\) (actions being the order sizes) the aggregated action set will be \(\{0,Q,2Q,\ldots\}\) with \(Q\) the batch size. For the cash-flow models of section 2 the action set when ordering a car is \(\{-1,0,1,\ldots\}\), where \(-1\) stands for "no car" and \(a = 0,1,\ldots\) stands for "let a car come by and bring \(a\)". A suitable aggregated action set then is \(\{-1,0,Q,2Q,\ldots\}\).

The main remaining question is about the batch size \(Q\), or, more generally, about the size of the intervals. It is possible to chose the aggregated problem in such a way that its solution is quite near the solution of the original problem. However, it would be nicer to use a fairly rough aggregation to begin with and let it follow by some disaggregation step.

When designing an appropriate disaggregation step which can follow the solution of a problem with an aggregated action space, one has to find a compromise between exactness and efficiency. Below we will formulate 4 variants for the disaggregation step of which only 3 guarantee the optimal answer. If one wants to be sure that the optimal answer is obtained, then either one should work with iterated aggregation-disaggregation as in the linear programming approach (cf. Mendelssohn [5]) or one should use the aggregation step mainly to get a good estimate for the value function and ultimately use this estimate for the solution of the original problem. In the 3 variants leading to the exact solution we will use the latter approach. A disadvantage apparently is that at some stage one should work in this approach with the full sized problem.

Our fourth variant for the disaggregation step only requires the solution
of a small problem, which is formulated using the solution of the aggregated problem. It is possible to construct examples in which this approach does not lead to the optimal solution, however the solution is always at least as good as the solution of the aggregated problem, and practically it is always optimal.

Below we give a short description of the 4 variants. In the description we assume that successive approximation methods will be used for the aggregated as well as for the disaggregated problem.

Disaggregation variants (suppose that the aggregated problem with batch size Q has produced the estimate \( v \) for the value function and action \( \pi(i) \) for state \( i \):

1. Solve the original Markov decision problem, starting with guess \( v \) for the value function.

2. Solve a sequence of Markov decision problems, where each of the problems has as initial guess for the value function the result of the preceding problem, and where the level of aggregation, the batch size, is half the one of the preceding problem. The sequence stops if the original problem is solved.

3. Solve the Markov decision problem which is obtained from the original one by only allowing in state \( i \) the \( Q \) actions which lie around \( \pi(i) \), using \( v \) as starting guess for the value function. After this step one successive approximation (policy improvement) step is made in the original problem to check the overall optimality of the result. If some actions outside the restricted regions give nonnegligible improvements, then the whole procedure is repeated with these actions as midpoints of the restricted action spaces and so on.
4. First one successive approximation step is made in the Markov decision problem which is obtained from the original one by allowing in state \( i \) only the \( Q \) actions lying around \( \ell(i) \). In each subsequent iteration the restricted action sets are shifted in such a way that always the optimal actions from the preceding step are midpoints of the allowed regions. In this variant we never iterate with the full action sets.

The choice of the particular successive approximation method depends on the structure of the problem at hand. For instance, for periodic problems, as the cash-flow models of the previous section, a Gauss-Seidel method as given in [7,8] is most efficient. This Gauss-Seidel method makes it also possible to exploit the typical inventory control structure for a very efficient handling of the iteration step. In fact, this iteration step is so efficient that one cannot expect to win much by aggregating actions, except in the case that the action space is also involved in the state space. (We will come back to this point in the next section.)


In this section we will exhibit some of our experience with the aggregation-disaggregation methods described in the preceding section. In each of the three examples (an \((s,S)\) inventory model and the two cash-flow models) we have worked with a discountfactor of 0.999. The discount-factor only guarantees theoretical convergence for successive approximation methods, in practice the convergence is fully determined by the stochastics involved.
The computations have been made on the Burrough’s 7700 of the Eindhoven University of Technology. In order to make comparison possible we mainly give processing times.

A. A standard \((s,S)\) inventory problem.

In the one-point inventory problem with backordering, linear holding and penalty costs, and fixed order costs an \((s,S)\) policy is known to be optimal. It is also known that for a cleverly chosen variant of the policy iteration method, moving from one \((s,S)\) policy to another, the usual bottleneck of solving a set of linear equations can be avoided (cf. Johnson [3]). Using this method a processing time of 4 seconds has been obtained for a problem with 424 stock levels and 40 demand levels. The best successive approximation method (Gauss-Seidel using bisection [1,2] if possible) yielded a processing time of 23.5 seconds. Beforehand one may prophesy that this result will only be improved slightly by an aggregation-disaggregation approach, since the large number of decisions does not give much extra work if the inventory structure is exploited well. Indeed the best result obtained was a processing time of 20 seconds (batch size \(Q = 10\)). So for this problem by far the best result is obtained for a clever policy iteration method and the aggregation-disaggregation method is worthless.

B. The hard cash inventory problem with one-period timelag.

For the hard cash inventory problem with one-period timelag one may hope that a substantial gain can be made. For an efficiently programmed successive approximation method which exploits the specific cost and transition structure a processing time of 32.9 seconds has
been obtained (relative error .001 both for the aggregated and for the disaggregated problem). Table 1 shows how the processing time is reduced by the simplest disaggregation variant 1. As we see the best result is obtained for Q = 11: a reduction of over 60 percent.

<table>
<thead>
<tr>
<th>Q</th>
<th>aggregation</th>
<th>disaggregation</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12.4</td>
<td>7.4</td>
<td>19.8</td>
</tr>
<tr>
<td>5</td>
<td>8.4</td>
<td>7.5</td>
<td>15.9</td>
</tr>
<tr>
<td>7</td>
<td>6.2</td>
<td>7.5</td>
<td>13.7</td>
</tr>
<tr>
<td>9</td>
<td>5.2</td>
<td>10.1</td>
<td>15.3</td>
</tr>
<tr>
<td>11</td>
<td>4.4</td>
<td>7.5</td>
<td>11.9</td>
</tr>
<tr>
<td>13</td>
<td>4.0</td>
<td>12.3</td>
<td>16.3</td>
</tr>
<tr>
<td>15</td>
<td>3.7</td>
<td>12.4</td>
<td>16.1</td>
</tr>
<tr>
<td>17</td>
<td>3.3</td>
<td>15.0</td>
<td>18.3</td>
</tr>
<tr>
<td>19</td>
<td>3.5</td>
<td>17.5</td>
<td>21.0</td>
</tr>
</tbody>
</table>

Table 1: processing times for variant 1

An aspect of interest is the choice of the relative errors. It is clear that in the aggregation phase there is no need to use the same small relative error as in the disaggregation phase. In table 2 processing times are given for relative error Q * .001 in the aggregation phase. Particularly for the larger batch sizes Q this gives quite an improvement.
<table>
<thead>
<tr>
<th>Q</th>
<th>aggregation</th>
<th>disaggregation</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>10.6</td>
<td>7.6</td>
<td>18.2</td>
</tr>
<tr>
<td>5</td>
<td>6.5</td>
<td>7.5</td>
<td>14.0</td>
</tr>
<tr>
<td>7</td>
<td>4.8</td>
<td>7.5</td>
<td>12.3</td>
</tr>
<tr>
<td>9</td>
<td>3.6</td>
<td>7.6</td>
<td>11.2</td>
</tr>
<tr>
<td>11</td>
<td>3.1</td>
<td>10.0</td>
<td>13.1</td>
</tr>
<tr>
<td>13</td>
<td>2.5</td>
<td>10.1</td>
<td>12.6</td>
</tr>
<tr>
<td>15</td>
<td>2.3</td>
<td>10.0</td>
<td>12.3</td>
</tr>
<tr>
<td>17</td>
<td>2.2</td>
<td>12.9</td>
<td>15.1</td>
</tr>
<tr>
<td>19</td>
<td>2.0</td>
<td>12.5</td>
<td>14.5</td>
</tr>
</tbody>
</table>

Table 2: processing times in seconds for variant 1 with relative error $Q * .001$ in the aggregation phase and relative error $.001$ in the disaggregation phase.

Note that if we compare with table 1 for larger values of $Q$ also the time in the disaggregation phase decreases and also note the robustness with respect to the batch size of the total processing time. Variant 2 with its decreasing sequence of batch sizes gives no improvement of the results of table 2, but it is again quite robust with respect to the starting value of $Q$, see table 3.

<table>
<thead>
<tr>
<th>Q = 4</th>
<th>Q = 8</th>
<th>Q = 12</th>
<th>Q = 16</th>
<th>Q = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.0</td>
<td>14.9</td>
<td>15.2</td>
<td>14.1</td>
<td>14.3</td>
</tr>
</tbody>
</table>

Table 3: processing times in seconds for variant 2 with relative error $Q * .001$ for the aggregated problem.
Finally variants 3 and 4 are run, again with relative error $Q^* .001$. Variant 3 gives some further improvement, however, as could be expected the real winner is variant 4 (cf. table 4).

<table>
<thead>
<tr>
<th>Q</th>
<th>aggregation</th>
<th>disaggregation</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>variant 3</td>
<td>variant 4</td>
</tr>
<tr>
<td>3</td>
<td>10.6</td>
<td>3.7</td>
<td>1.7</td>
</tr>
<tr>
<td>5</td>
<td>6.5</td>
<td>4.2</td>
<td>1.7</td>
</tr>
<tr>
<td>7</td>
<td>4.8</td>
<td>4.2</td>
<td>1.8</td>
</tr>
<tr>
<td>9</td>
<td>3.6</td>
<td>4.6</td>
<td>2.2</td>
</tr>
<tr>
<td>11</td>
<td>3.1</td>
<td>4.5</td>
<td>2.7</td>
</tr>
<tr>
<td>13</td>
<td>2.5</td>
<td>6.2</td>
<td>3.7</td>
</tr>
<tr>
<td>15</td>
<td>2.3</td>
<td>9.0</td>
<td>3.7</td>
</tr>
<tr>
<td>17</td>
<td>2.2</td>
<td>7.5</td>
<td>4.7</td>
</tr>
<tr>
<td>19</td>
<td>2.0</td>
<td>10.8</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Table 4: processing times for variants 3 and 4 with relative error $Q^* .001$.

As we see variant 4 reduces for all batch sizes from 7 to 15 the processing time to about 20 percent of the original processing time of 32.9 seconds. So for this problem the aggregation-disaggregation approach is quite successful.
C. The hard cash inventory problem with two-period timelag.

Surprisingly the aggregation-disaggregation approach does not help very much in the hard cash inventory problem with two-period timelag. Only variant 4 gives a reduction in processing time from 89.1 seconds to 62.2 seconds for Q = 11. Bisection does not help either. In order to discover the cause of this somewhat unexpected failure (why would the situation of a two-period timelag be so different from the case with a one-period timelag) we need a detailed investigation of what happens in an iteration step. Let us consider a one day cycle starting at the end of the morning. Subsequently four things happen:

(i) when the car arrives we have to decide what to do.
(ii) we have to decide about ordering a car for the next day.
(iii) there is a random change of the cash level in the afternoon.
(iv) there is a random change of the cash level in the morning.

Writing i, i', i'' and i''' for cash levels and \( \ell \) and \( \ell' \) for states of the car we can make the following schematic picture of a day cyclus

\[
\begin{align*}
(i, \ell) \xrightarrow{\text{decision}} & i' \xrightarrow{\text{decision}} (i', \ell') \xrightarrow{\text{random}} (i'', \ell') \xrightarrow{\text{random}} (i''', \ell').
\end{align*}
\]

Now let us consider the steps (i) - (iv).

(i) Due to the specific structure of the inventory problem the amount of work needed to find for each pair \((i, \ell)\) the optimal decision is small and hardly influenced by the number of available actions.

(ii) The number of states \(i'\) is small, so this step requires relatively little time.
(iii) Here we have to calculate for any of the appearing pairs 

\((i',t')\) an expression of the form \(E_d p_d v((i' - d, t'))\), where 

\(p_d\) is the probability of a demand \(d\) and \(v((i' - d, t'))\) is the 

present guess for the total expected discounted cost when 

starting in \((i' - d, t')\). The number of pairs \((i', t')\) that will 

appear in the disaggregation phase of variant 4 is substantially 

smaller than the number that will appear in the original, 

\(Q = 1\), problem.

(iv) This step explains the relative failure of variant 4. The sit-

tuation is comparable with step (iii) except that now also in 

the disaggregation phase practically all pairs \((i'', t'')\) may 

appear, due to the random charge of the cash level the prece-

ding afternoon. So this step, which together with step (iii) 

is also the bottleneck for the original problem, is about 

equally time consuming for the original problem as for the 

disaggregation variant 4.

Now that we have discovered the cause of the failure, the next question 

must be: is there a way to circumvent this difficulty? In this case 

there is one. We can combine the half day transitions of the after-

noon and the morning to a full day transition. This requires the a 

priori computation of these one day transition probabilities, which 

in this case, due to the inventory structure, is not very expensive. 

We only need the convolution of the afternoon and morning demand 

distributions. (In general combining two transitions into one requi-

res the multiplication of the transition matrices and will be expen-

sive if the matrices are large and full.)
Combining (iii) and (iv) yields the following picture for the state changes during a day.

\[
\begin{array}{ccc}
\text{decision} & \text{decision} & \text{random} \\
(i, \ell) & \rightarrow & i' \rightarrow (i', \ell') \rightarrow (i'', \ell').
\end{array}
\]

Let us compare this with the picture for the state changes in a day for the one-period timelag model

\[
\begin{array}{ccc}
\text{decision} & \text{random} & \text{decision} & \text{random} \\
(i, \ell) & \rightarrow & i' \rightarrow i'' \rightarrow (i'', \ell') \rightarrow (i''', \ell').
\end{array}
\]

We see that the picture for the two-period timelag is even simpler. So, once we have calculated the full day transition probabilities (eliminating the afternoons) the aggregation-disaggregation approach will yield at least as good results as in the one-period timelag model.

5. Conclusions.

Apparently the aggregation-disaggregation approach for action spaces does not help for standard \((s, S)\) inventory problems. However, for problems in which actions also appear in the state space, the approach may be quite useful. The effectiveness of the method is also easily ruined as case C shows. A big advantage of the approach is that the modelling as well as the programming is very simple and straightforward.
References.


