Solving large structured Markov Decision Problems for perishable inventory management and traffic control

Haijema, R.

Publication date
2008

Chapter 9

Epilogue

The topic of this thesis is large structured Markov decision problems (MDP). With large we mean 'too large to solve the MDP to optimality'. With 'structured' we indicate that many MDPs exhibit some structure that can be exploited in the search for an approximate solution. Many real-life sequential decision problems can be formulated as an MDP, but too often it is believed that the MDP's state space is too large to compute an optimal strategy. The tendency for multi-dimensional MDPs to become too large to solve, is often called the *curse-of-dimensionality*. Nevertheless, the formulation of a problem as an MDP maybe very helpful in obtaining an approximate solution using existing techniques.

Two approaches that are widely used in many mathematical programs for solving large problems are aggregation and decomposition. The possibilities to aggregate or to decompose an MDP strongly depend on the specific properties of the problem that define the problem structure. Another approach to solve high-dimensional MDPs, is approximate dynamic programming [117]. Although having the same objective, we consider approximate dynamic programming as another line of research.

For two entirely different applications, we have shown that aggregation and decomposition can be applied successfully to solve MDPs that arise in real life. The two applications may be inspiring for solving similar problems. In the first part of the thesis, we have focused on an application in which aggregation is used to reduce the number of states in the MDP. In the second part decomposition of the state space is used to obtain an approximate solution through a one-step-policy improvement algorithm.

A production-inventory-problem of perishables with a short fixed shelf life

Although the platelet production-inventory problem (PPP) is a high-dimensional problem, the problem exhibits enough structure to formulate an MDP problem and solve it. Solving the MDP of the PPP at a Dutch blood bank, requires the aggregation of states such that an SA algorithm can be executed in a reasonable amount of time. Aggregation is needed at several levels. First of all, the blood groups are to be aggregated as if blood platelet pools (BPPs) are of a single universal blood group. Secondly, BPPs are aggregated in batches when the PPP plays at a large scale. Finally, to come up with a simple rule, the stock consisting of batches of different age categories are aggregated into a single number. By simulation the structure of the optimal strategy is investigated and approximated by simple rules.

Whether the state aggregation at all three levels is possible depends on the context of the problem as well as on the actual data. In cases where other standards would apply concerning the compatibility of the different blood groups, or when the supply of buffy coats is limiting, the aggregation of blood groups into a single universal blood group may not be justified. When blood groups have to be matched strictly, aggregation at this first level is not justified. Then one may decompose the problem and solve the ordering problem for each blood group in isolation of all others.

In some cases the other two aggregation steps are not required or not justified. On the one hand, when the demand for BPPs is very low, there may be no need for aggregating BPPs into batches. On the other hand, when the coefficient of variation of the demand is high, e.g. because demand levels are low, a simple order-up-to S rule may be too far from optimal. Then the SDP-Simulation approach can be used for deriving more advanced stock-level-dependent and stock-age-dependent ordering rules.

For the PPP with realistic data from a Dutch blood bank, aggregation at all three levels is possible and needed. For the given data, it appears that an order-up-to S rule roughly fits to the optimal ordering decisions. Even for smaller cases, such as that of a medium-large hospital, the order-up-to S rule performs quite well. The inclusion of fixed order costs is no problem, although it gives rise to order-up-to rules with thresholds for ordering, such as (s, S)-policies.

The SDP-Simulation approach does not only give insight in the structure of an optimal policy, but also translates it, when possible, into a simple rule and yields the corresponding (nearly) optimal parameters values. The approach seems to be fruitful for applying it to other production-inventory settings that involve perishables with a short fixed shelf life.

The software developed during this PhD project has resulted in an application named TIMO, which is currently implemented at one of the Dutch blood banks. The application shows the usefulness of Stochastic Dynamic Programming and aggregation techniques in solving problems that seem to be intractable at first sight. The application of the SDP-Simulation approach is not straight-forward and requires insight into the problem to deal with the involved modeling steps.

The dynamic control of traffic lights: a queueing problem

The dynamic control of traffic lights is a typical queueing problem, where the intersection can be seen as a server which is able to serve multiple, but not all, queues at a time. When switching, from serving one set of queues to serving another set, a fixed switch-over time applies. During the first part of the switch-over time the service of a queue might continue, e.g. as long as the light stays yellow, but the switching process cannot be interrupted. At the last part of a switching phase, no queues get served, e.g. an all red phase applies to clear the intersection.

For the dynamic control of traffic lights optimal decisions end green periods and specify the order in which queues are served, such that the long-run average waiting time is minimized. Input to the control are the number of cars waiting at each queue as well as the state of the traffic lights. Given the potential number of cars waiting at each queue, the number of states explodes rapidly when the number of queues is increased. Whereas an optimal policy for a small intersection, with say only 4 flows is tractable, one has to rely on approximate solutions for intersections with many queues.

The approximation that we discuss is a one-step policy improvement over a well-structured policy that allows decomposition of the state space. For the control of traffic lights, the control according to a fixed cycle (FC) allows for such a decomposition. Under FC the waiting time at each flow can be evaluated flow-by-flow after formulating a Markov chain. The relative values (RV) of the states under FC are input to a one-step policy improvement algorithm.

The most simple improvement rule is the policy that we called RV1: every time slot FC may be interrupted. Therefore all feasible jumps within the cycle are considered and evaluated using the relative values under FC. In the evaluation of a time jump it is assumed that the FC resumes after the jump as if it is not interrupted in the future.

RV1 appears to significantly improve FC and other control policies such as anticipative exhaustive control. There are simple infrastructures for which the difference between

anticipative exhaustive control and RV1 is small. For more complex intersections, where multiple (non-conflicting) flows receive green at the same time, RV1 gives an easy way out of the complex sequential decision problem.

A problem similar to the control of traffic lights may be observed in a production-inventory setting. A production facility, or production department, produces various products. Suppose it is able to work at only a few product series at a time, and switching between series of different products requires a switch-over time, e.g. for cleaning and setting-up machines and instructing operators and other staff. One has to decide (dynamically) on the length of a product run, and which product run to set-up next. On the one hand, the decisions should balance the probabilities of getting out of stock for one or more products: short product runs may therefore be favored. On the other hand, short production runs for each product may be very inefficient, as switching implies switching costs and some set-up time. Optimal decisions on the order and the length of the production runs depend, amongst other, on the actual stock-levels of all products and on the demand distribution.

An approximation of this production-inventory problem, is by imposing a special policy that imposes additional structure to the problem such that the state space can be decomposed. Such a policy may be a cyclic production schedule, in which the order as well the length of production runs are fixed. A one-step policy improvement approach for this problem is considered by Bruin and Van der Wal ([22] and [23]).

Aggregation or Decomposition?

It is hard to formulate general statements on when aggregation or decomposition is possible, as it depends strongly on both the problem and the data. Without claiming any generality, we give some thoughts at a high level.

In general decomposition of a problem implies the distinction of subproblems that can be solved one-by-one. When solving an MDP, the state space can sometimes be decomposed when one restricts to a special class of well-structured policies. Depending on the problem, the state space can also be decomposed by imposing additional modeling assumptions. Decomposition of the state space into subspaces requires that each subspace contains all relevant information of the system's state to evaluate a related subproblem.

Whereas decomposition may help the evaluation of a policy, a one-step policy improvement algorithm is used to find a better policy. Usually, the improved policy is not well-structured, and does not allow the decomposition of the state space. Consequently, the improved policy can be evaluated by simulation only. A second improvement step is

not possible, as the relative values of the states under the improved policy cannot be determined.

In an MDP where the different dimensions of the state space are directly correlated, decomposition of the state space is not possible. For example, in the PPP the element x_1 at some day, strongly depends on the value of x_2 at the previous day: BPPs with a residual shelf life of 2 days that are not issued on a day, have the next day a residual shelf life of one day. For the PPP the dimensions r and $r+1$ of successive state vectors, $\mathbf{x}$, are too much correlated to allow the decomposition of the stock-state space into subspaces.

The vector $\mathbf{x}$ in the PPP can take a large number of values in each dimension. As the elements refer to a number of BPPs, aggregation is possible by counting them in batches of say 4. For an arbitrary problem, this way of aggregating states may not be possible, as it strongly depends on the interpretation of the elements in the vector. Another level of aggregation is the aggregation of multiple dimensions into one dimension, e.g. BPPs of 8 different blood groups are modeled as if their are from a single universal blood group. This way of aggregation leaves information out of the state space. This is only possible when this information is (expected to be) not relevant for deriving optimal decisions.

Conclusions – The formulation of a sequential decision problem as an MDP maybe very helpful in finding an approximately optimal solution, even when the problem seems to be too large to solve. Whether aggregation or decomposition can be applied to obtain a good approximation of an optimal MDP policy, depends strongly on the structure of the problem as well as on the data. Looking for problem structure or imposing additional structure helps in approximating a high-dimensional MDP. Whether the approximation is justified is to be checked by simulation. Simulation is often the only technique that is capable of evaluating the resulting policy at a detailed level.