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

Haijema, R.

Publication date
2008

Chapter 5

Introduction and outline of Part II

In 1868, about 30 years before the motor car was invented, the first traffic lights were installed. At that time, traffic lights were introduced in front of the British House of Commons to control the flow of horse buggies and pedestrians. The invention came from the British railroad signal engineer J. P. Knight. The lights consisted of a gas lantern with red and green signals. Unfortunately, these gas lanterns appeared to explode easily.

In the beginning of the twentieth century more traffic signals were developed as motor traffic grew rapidly these days. From 1910 a traffic control system consisted of signals showing 'Stop' and 'Proceed' (or 'Move'), which had to be operated manually. In 1917 automated signals were introduced using two electric lights: red and green. In 1920 the yellow (or amber) light was added by a Detroit policeman. In these days automatic traffic lights were introduced to regulate the priority of the different traffic participants with no intervention of a policeman.[1]

From the early fifties the control of traffic lights is studied from the perspective of car drivers who wish to minimize their delay due to waiting at intersections. Since then several delay models and control policies are developed.

In this part of the thesis we study the optimal dynamic control of traffic lights. In the first section (Section 5.1) we define the scope of our research and we introduce the problems that are studied in the next chapters. In Section 5.2 we report on the most relevant studies on the control of traffic lights. Before we present our approach in the next chapter we report in Section 5.3 on similar approaches to solve other high dimensional problems.

Those who are less interested in the literature should at least read Section 5.1, before they proceed to Chapters 6, 7 and 8.

[1] This introduction into the history of traffic lights is mostly based on the information from [161].

5.1 Three traffic light problems of interest

5.1.1 Single intersection in isolation

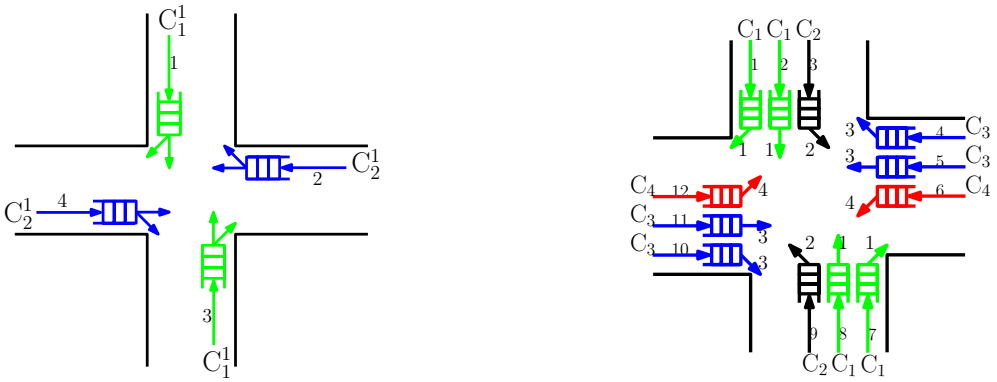
Traffic lights are introduced in the early start of the previous century to make road traffic safer, at places where traffic from different directions cross the same road segment, called the intersection or crossing. By giving right of way to traffic in some direction(s), cars approaching from other directions need to wait before they get priority. By sophisticated control of the traffic lights the overall delay or waiting time of the cars can be kept to a minimum. In practice, traffic engineers aim to set a good (hopefully nearly optimal) control scheme using simulation software, delay formulas and experience. The underlying optimization problem is not always clearly defined by policy makers. Several objectives are possible: minimize the delay, minimize pollution by cars or a combination. In addition, (political) constraints may apply such as public transport traveling at a separate lane has the highest priority over all other traffic flows.

We formulate throughout the next three chapters the control problem as a clean mathematical problem of minimizing the long-run average waiting time, dealing with cars only. We acknowledge that changing priority takes a switching time during which green lights turn into yellow and next an all red phase applies to clear the intersection. Cars continue passing the stopping line as long as the light shows green or yellow. Our definition of waiting time is the time a car spends in the queue, no matter the color of the light: waiting time is thus the time between joining a queue and passing the (downstream) stopping line.

By induction loops, that are cut into the surface of the roads, or by using cameras, one may count (or estimate) the number of cars waiting in each queue. The thus obtained information on the queue lengths can be used in reducing the waiting time at intersections. In the next chapter we formulate the optimization problem for a single intersection in isolation as a multi-dimensional Markov decision problem (MDP). The number of cars waiting in each queue is included in the state description.

Example – Some notations and the computational complexity

For example, consider the infrastructures in Figure 5.1. Figure 6.1(a) depicts a simple intersection with only $F = 4$ traffic flows (or streams) leading to 4 queues. The flows and queues (f) are numbered clockwise: 1–4. Stream 1 and 3 receive green simultaneously and are grouped in combination 1 (C_1). Combination 2, C_2, consists of the streams 2



(a) "F4C2" serve 4 flows in 2 symmetric combinations

(b) "F12C2" serve 12 flows in 4 asymmetric combinations

Figure 5.1: Two typical infrastructures.

and 4. At most one combination at a time has right of way (when its lights are green or yellow). Suppose many more cars are waiting at C_2 than at C_1, but C_1 is currently getting served, a typical question is: 'When should we turn the lights of C_1 into yellow?' Should we wait until both queues of C_1 are empty or should we switch as soon as one of the queues is empty or does an optimal action depend on the precise numbers of cars waiting at each queue? A similar question arise when only one car is waiting at C_2 and the queues of C_1 are almost exhausted: should we keep the lights green to C_1 until more cars are waiting at C_2?

Figure 6.1(b) shows a more complex intersection where $F = 12$ flows are grouped into 4 combinations C_1 to C_4. C_1 and C_3 consist of twice as many flows than C_2 and C_4. The grouping of the streams is also indicated by the numbers in the front of each queue. The asymmetry in the number of queues in the four combinations, complicates the decision making. As long as some cars are waiting at all queues, giving priority to C_1 or C_3 results in more departures per unit time than giving priority to C_2 and C_4.

The decisions *'when-to-end-a-green-period'* as well as *'which-combination-to-serve-next'*, should depend on the number of cars waiting at each queue: $\mathbf{q} = (q_1, q_2, ..., q_{12})$. An optimal policy prescribes for each possible state, i.e. each value of the vector $\mathbf{q}$, the action to take given the current state of the lights. Such an optimal policy can be obtained from the solution of the underlying Markov decision problem. When each element q_f can take only 10 possible values (0 to 9), then there are already 10^{12} possible vectors $\mathbf{q}$ to consider. Thus the state space of the MDP easily exceeds the computationally acceptable limit of say 10 million states.

Solving the MDP is only possible for simple infrastructures with a 'small' number of traffic streams and under non-saturated conditions such that the queue lengths can be bounded. Large MDPs cannot be solved due to the large number of states. Therefore we develop an approximate solution, based on a one-step policy improvement algorithm. Such an approach has been successful to other types of problems, which are discussed in Section 5.3. In Chapter 6 the new approach is presented for the control of a single intersection in isolation. As we will see, these new policies greatly improve existing policies that we have evaluated in a simulation model.

5.1.2 Isolated intersection with arrival information

With todays technology one can observe the number of cars waiting in each queue, as well as the individual cars driving on each lane ([93], [65]): hence the arrival times to the queue of nearby cars can be predicted. This information can be used in a control policies that anticipates near-future arrivals. In Chapter 7, we extend the new policies that are introduced in Chapter 6, by including arrival information. In a detailed (microscopic) simulation model we show that the average waiting time can be reduced by taking only a limited amount of arrival information into account.

5.1.3 Networks of intersections

Finally, in Chapter 8, we consider several networks of intersections. We suggest to control the traffic lights in the network for each intersection in isolation of all others. By using some arrival information, we hope to realize a desirable degree of synchronization of the intersections. The resulting policies are evaluated by an extended simulation model, in which we may simulate an arbitrary number of intersections at different workloads and where car speeds may differ. Main criterion remains minimizing the long-run average waiting time that a car experiences at an arbitrary intersection.

5.1.4 Outline

Before we present the MDP approach for the above three problems in the next three chapters, we successively discuss in this chapter relevant studies on:

- the control of traffic lights, and
- the application of a one-step policy improvement approach to high-dimensional MDPs.

5.2 Studies on control of traffic lights

The optimal control of traffic lights is studied by researchers from different disciplines using different techniques:

- Operations Research-ers study the problem using mathematical techniques to evaluate and optimize the control of the lights,
- Computer scientist use techniques from Artificial Intelligence (AI) to develop heuristics, local search procedures and learning algorithms, and
- Simulation experts develop genuine microscopic simulation models to carefully evaluate different control schemes.

There is a rich amount of literature making it impossible to give an extensive overview. Instead we address some relevant articles. The studies on the control of traffic lights can be divided into two categories:

1. studies on the static control of the lights, in which the control is fixed and does not depend on the actual situation at the intersection,
2. studies of the dynamic control of the lights that use information on the actual situation at the intersection.

In the next sections we discuss a number of control schemes that are well studied.

- FC or pre-timed control,
- Exhaustive control,
- Vehicle actuated control, and
- Adaptive control.

5.2.1 Fixed cycle

The fast majority of the Operations Research literature on traffic control is on fixed cycle control (FC), since under FC the control has a simple structure that allows mathematical analysis. FC is a static control policy: it does not take into account any information that changes dynamically over time. Under FC the order in which queues get green as well as the duration of each green period is fixed. Hence the decisions about when to switch are predetermined and do not depend on the actual number of cars waiting at the queues. Therefore FC is also called fixed-time or pre-timed control.

A traffic responsive version of FC allows to choose between several predetermined FCs: at times at which it is very busy one applies an FC with a long cycle length. Hence during rush hours a longer cycle length applies than during the rest of the day. During the night the lights might be even turned off.

One of the first and most cited studies on FC is that of Webster in 1958 at the Road Research Laboratory in London [162]. By applying analytical and numerical techniques (simulation and curve fitting) he derived his famous delay formula based on the M/D/1 queueing system:

$$\mathbb{E}W_f \approx \frac{(D-d_f)^2}{2D(1-\lambda_f/\mu_f)} + \frac{\lambda_f/\mu_f D^2}{2d_f(\mu_f d_f - \lambda_f D)} - 0.65\left(\frac{D}{d_f^2}\right)^{1/3}\left(\frac{\lambda_f D}{\mu_f d_f}\right)^{2+5d_f/D} \tag{5.1}$$

which estimates the expected delay $\mathbb{E}W_f$ of a car in flow f, as a function of the duration of a cycle D, the effective green time d_f during which departures may happen from queue f, and the arrival and departure rates λ_f and μ_f.

Shortly after the introduction of Webster's delay formula, other approximations were developed; see the work of Miller [94], Newell [103] and McNeill [90]. McNeill's delay formula is exact when one would have an exact expression for the average queue length at the start of a red period. In 1964 such an expression was found by Darroch.

In [33] Darroch presents a formal solution for the stationary distribution of the queue length under FC at discrete points in the cycle. Herewith Darroch provides an expression for the mean number of cars queued upon the start of a red period. However, the expression requires extensive numerical computations, for this and other reasons his results was not much used. In 1978 Darroch's approximations were reformulated by Ohno [106]; he made Darroch's analysis more accessible and compared the different approximations available at that time. Nowadays the computational problems to evaluate Darroch's exact

expression are resolved, since the numerical schemes are clarified and computing power has increased. Nevertheless, approximations are still very useful in providing insight into the performance of FC and to heuristically optimize FC [145].

Recently, in 2006, Van den Broek, Van Leeuwaarden, Adan and Boxma provided new bounds and approximations, see [146]. They provide efficient numerical procedures for some of their results, and they have derived also some closed-form expression that do not require numerical procedures.

Van Leeuwaarden extends the work of Darroch on the queue length distribution. He derives in [153] the probability generating function of the queue length distribution and the delay distribution for more general discrete arrival distributions. (Darroch focussed on (compound) Poisson arrivals in a discrete time model.) Van Leeuwaarden argues that the numerical efforts for finding the distribution are not a big issue any more, since numerical search procedures can be structured as FC is well-structured. Once the distributions are computed, one easily computes the performance measures such as the mean, the variance and the percentiles of the waiting time.

So far, all articles focus on a single intersection in isolation. The control of the traffic lights of successive intersections of an arterial is considerably more difficult, since one has to specify the start of the green periods, next to the length of each green period. Setting green waves in 1 or 2 directions might contribute to the minimization of the waiting time. In the master thesis of Van den Broek [145], the developed approximations are used to optimize FC at some realistic intersections. The approximate delay functions are transformed into spline functions, such that a Mixed Integer linear Programming model (MIP) can be formulated and solved to determine (nearly) optimal lengths and starting times of green periods. The approach is also used to control an arterial of two intersections.

Another technique for finding a good synchronized FC is by using a genetic algorithm. In TRANSYT-7F hill climbing techniques and genetic algorithms are implemented to heuristically optimize FC for a network of intersections ([122], [84]). Park et al. [109] discuss a genetic algorithm that optimizes over the start and length of the green periods as well as over the sequencing of the phases of FC: a phase is a period of time during which a specific combination receives green. We do not discuss all software that is available for this purpose. For an overview we refer to surveys like the one of Papageorgiou et al. [108] and the survey published on the web site of the ITS group of the University of Leeds (see [160]).

5.2.2 Exhaustive control

Under exhaustive control a (effective) green period of a combination is ended as soon as all queues related to that combination become empty. Based on the visiting order, one distinguishes cyclic and acyclic exhaustive control policies. Under cyclic control the order in which the combinations are served is fixed, but a combination may be skipped in a cycle when no cars are waiting upon the intended start of the green period. Under acyclic control we need to specify which queue to serve after an all-red phase. For example, this could be the combination with the highest number of waiting cars or the combination that contains the longest queue. We distinguish two classes of exhaustive control policies based on the definition of 'exhaustion': *pure-exhaustive control* and *anticipative-exhaustive control*.

Pure-exhaustive control

Under *pure-exhaustive control* signals stay green until all queues that have right of way are empty, say 'exhausted'. We abbreviate the cyclic pure-exhaustive control policy by XC; XA is the acyclic variant. Note that XC does not require detailed information concerning the number of cars queued: next to the state of the traffic lights, XC needs for each queue only information on whether cars are present or not.

Anticipative-exhaustive control

Cars leave a queue as long as its lights show green or yellow. A policy may anticipate the departures during a yellow slot by switching from green to yellow before the queues that have right of way are exhausted. Therefore we introduce, what-we-call, *anticipative-exhaustive control* policies. Under anticipative-exhaustive control, departures during the yellow phase are anticipated by switching already when at each of the queues that have right of way one or two cars are present. These polices are denoted by XC-1 (or XA-1) respectively XC-2 (or XA-2). Compared to pure-exhaustive control, anticipative-exhaustive control needs information about the actual number of cars waiting at each of the queues but ignores any information on near-future arrivals.

Remark – Polling models: For intersections with only a single queue per combination, the analysis of exhaustive control can be done by modeling the service of the queues by a polling model. Polling models have been extensively studied in Queueing theory: in

a polling model a single server visits one queue at a time to provide service to jobs or customers in that queue only. For a few problems, it has been proven that exhaustive control is optimal under specific conditions. When switching costs or switching times apply, switching thresholds may apply: one switches only when the queues that get service become empty and 'enough' jobs, say cars, are waiting at the other queues. Most of the articles deal with production or service networks rather than signalized intersections.

A rich amount of articles have been published on polling systems. We do not discuss polling models here, since at most infrastructures a combination consists of multiple queues. For an overview of such queueing models we refer to surveys of Takagi ([136], [137],[138],[139]), and Adan, Boxma and Resing [1]. Specific polling models that acknowledge a positive switching time are discussed by Hofri and Ross in [67], by Boxma, Levy and Weststrate in [20], by Reiman and Wein in [121], by Van der Mei in [91], and by Winands in [167].

5.2.3 Vehicle actuated control

In practice, one often finds a mixture of FC and exhaustive control: lights stay green until the queues are exhausted or a maximum green time has elapsed. An induction loop cut in the pavement upstream at a fixed distance of the stopping line detects when a car is approaching the stopping line. When no car has passed this induction loop for β seconds, the green period of that stream is ended. The green phase is also ended when some predetermined maximum green time has elapsed. Next to a maximum green time often a minimum green time applies. Such a policy is called *vehicle actuated control*, since the actions are triggered by the presence of vehicles.

Two of the first articles on vehicle actuated control appeared in 1964 by Darroch, Newell and Morris [34], and by Dunne and Potts [39] respectively. Dunne and Pott relate their algorithm to a kind of vehicle actuated control: the control functions they propose allow dynamic control based on the actual number of cars queued and thresholds apply in determining when to switch. Darroch et al. study the impact of the choice of β on the average delay. In their paper they provide formulas for the mean and variance of the (variable) cycle length, for the delay per vehicle and for the optimal value of β_f for each flow f.

Some later studies on vehicle actuated control are by Cowan [32], and Akcelik [3]. In 2002 Taale [135] compares different methods to optimize vehicle actuated control.

5.2.4 Adaptive control

Under adaptive control the lights are adjusted based on the actual conditions of the traffic flows. According to this definition vehicle actuated control is adaptive. Pre-timed control can be adaptive when from a set of off-line calculated FCs a best one is selected, e.g. based on the workload over the last say 5-15 minutes. Existing control systems, like SCOOT and SCAT ([87], [70], [86]), are based on this idea. Under adaptive control, we thus assume new information to be fed to the controller on a regular basis, say every T seconds. An adaptive control policy that gets frequent updates of the state of the system has, what we call, a high resolution.

In this subsection, we restrict adaptive control to strategies that take into account at least the information on the actual number of cars waiting. According to the definition by Van Katwijk, De Schutter and Hellendoorn [151], adaptive control strategies do optimize online over the state of the traffic lights. Therefore an adaptive control considers the actual numbers of queued cars and, if available, estimated arrival times of cars on their way to the intersection(s).

Based on the system's state, a number of actions is evaluated over some planning horizon and a best one is selected for implementation. Commonly a rolling planning horizon approach is used, since new information becomes available, say every T seconds. The optimization process to find the best (set of) decision(s) to implement for the next T seconds should be done in real time. All necessary calculations have to be done in a short time-window, at least before new information becomes available. The computation times can be kept low, when the evaluation of a decision is based on a short planning horizon. However, an accurate evaluation needs an horizon of sufficient length.

For an accurate evaluation of a decision one commonly takes future arrivals and future decisions into account. From the decision tree in Figure 5.2, it becomes clear that the number of sequences of decisions until the end of the planning horizon is huge. At each square in the decision tree a decision concerning the lights has to be taken: in the illustration it is assumed that there are two possible decisions every decision epoch. Each circle represents the event of arrivals and departures, which is modeled by a stochastic experiment when the arrival times of cars are unknown. The potential number of states to consider in the last layer, can be very large, since the number of events is exponential in the number of flows F. In Figure 5.2 it is assumed that the arrival times of cars are known for the first two time intervals, this reduces the number of nodes significantly.

Commonly the potential number of paths to evaluate is thus very large. Fortunately

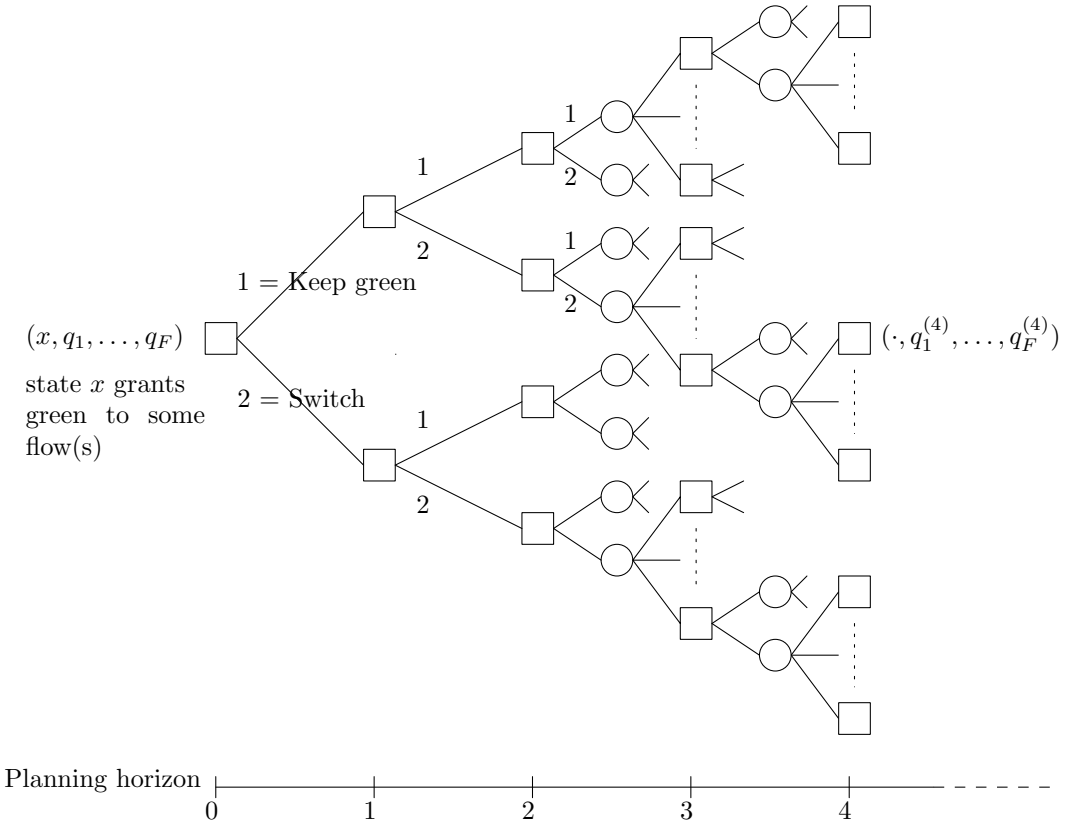


Figure 5.2: Decision tree under adaptive control.

standard OR techniques will skip the evaluation of many sub-optimal paths. By making use of Bellman's principle of optimality the computations can be done efficiently by forward Dynamic Programming (DP)[14]. DP is applied in existing control systems OPAC, PRODYN and RHODOS, see [48], [63], and [127] respectively. Alternatively, one uses a Branch and Bound (B&B) technique to find an optimal sequence of decisions. B&B techniques are used in ALLONS and in UTOPIA, and in RHODOS in combination with DP (see [116], [88], [127]). Even when applying DP or B&B algorithms, the computation time exceeds the available time when the depth of search tree is too large.

In the traffic engineering literature common ways to limit the computations are:

1. leaving out unknown future arrivals and evaluating waiting time until all present cars have left; this way the number of branches in the tree is reduced significantly,

2. decreasing the resolution by which decisions are taken, such that the number of layers in the tree reduces, and

3. shortening the planning horizon.

The first option is clearly inaccurate, since the impact of the decision on unknown future car arrivals is ignored. Hence, when the departure process is deterministic, only deterministic components of the waiting time are considered. This approach is implemented in OPAC, PRODYN and UTOPIA, see [48], [63] and [88] respectively. The resolution at which decisions are updated is 3 seconds for UTOPIA and 5 seconds for PRODYN and OPAC.

The latter two options do consider future arrivals, but ignore any delay experienced by cars that stay behind at the end of the planning horizon. Therefore one often applies terminal costs to value the impact of a decision on cars that depart (or arrive and depart) after the planning horizon, see Newell [104] and Shelby [128]. Bell and Brookes [13] state that the rolling planning horizon should be 2 minutes to compute the short term delay, and the effects beyond the horizon are estimated by state dependent terminal costs. From Figure 5.2 it becomes clear that the evaluation of the short term delay over 2 minutes may be too time consuming: when the decisions are taken every 5 seconds, then there are 60 layers in the search tree.

In ALLONS the fixed time intervals can be up to 15 seconds: every 5-15 seconds an optimal (sequence of) decision(s) is updated. In the online evaluation of a decision only deterministic arrivals are assumed, based on the predicted arrival times of all cars driving up stream. Van Katwijk et al. [152] suggest to split the planning horizon in time periods of unequal length. The estimation of the delay over a time interval requires a delay model. Therefore simulation can be used, but Van Katwijk et al. argue that a detailed microscopic simulation program may not fit when the update-frequency is high, due to the high computation times of such an program. Nevertheless more and more simulation models are used that take into account the physical length of queued cars. The phenomenon that a queue may grow upstream is called in the traffic engineering literature *queue spillback*.

Stochastic car arrivals are considered by Bell [11] using Markov chain techniques that require the storage of the expected delays for all possible state vectors. In PRODYN and RHODOS states are aggregated according to some approximate state equivalence relation. Yu and Recker formulate in [168] an MDP and transform the queue state description by imposing a threshold for each queue: when the number of cars queued is above the threshold the state is called saturated otherwise the state is non-saturated. This way they reduce the number of states in the MDP significantly. The transformation of the transition probabilities is however not clearly presented, which makes it hard to asses the approach.

When arrival information is not available, the number of sample paths can be very large, as illustrated in the right part of the decision tree in Figure 5.2. By simulation a decision can be evaluated over a number of arrival patterns (or sample paths). Since simulation requires less computations, a longer planning horizon can be considered. When only a small fraction of all sample paths is evaluated, the accuracy of the estimated delay may be poor.

Adaptive control policies are popular subjects of study for computer scientists in the field of Artificial Intelligence (AI). Commonly used techniques are learning algorithms and neural networks. Neural Networks are used to predict the delay that may be input to a dynamic program to find an optimal sequence of decisions. See for example the work of Theodorović et al. ([142], [143]).

An interesting application of a learning algorithm is that of Wiering et al. [164]: Q-learning is applied to heuristically minimize the overall average delay per car by dynamically adjusting the lights and routes traveled by cars having specific destinations in a network. The prediction of the delay is made in a microscopic simulation environment through reinforcement learning, so no prior known distributions of car arrivals are used. By dynamically computing routes to the known destinations of the cars traveling in the network, the algorithm aims at spreading the traffic over the different intersection such that the overall waiting time is reduced.

Tan et al. [140] and Lee et al. [83] use fuzzy logic. There are many more interesting articles published in the domain of AI. We do not go into the details here. For a review of the advances in traffic signal control we refer to [12], [37], and [108].

5.2.5 Simulation

Simulation plays an important role in testing the different control strategies. Several simulation programs are developed: most of them are microscopic simulation models, in which individual cars are generated such that the processes can be modeled at a realistic level. At the ITS group of the University of Leeds (UK) a thorough review [160] is executed of more than 30 (microscopic) simulation programs, like VISSIM, MIXIC, MICROSIM, and CORSIM. Most of them are developed by the scientific community to test new control policies against existing ones: nine of them are commercially available.

5.3 Studies on Decomposition and One-step policy improvement to solve high dimensional systems

Despite the fact that dynamic programming principles are known to and used by computer scientists, there appears to be no article in which a one-step policy improvement algorithm is used in combination with decomposition of the state space. Nevertheless, such an approach is proven to be beneficial to solve high-dimensional MDPs as we will learn in this section. In this section an overview is given of some applications for which a high-dimensional MDPs is approximately solved by a one-step policy improvement algorithm after decomposition of the state space. We skip the technical details and simple report the modeling assumptions that allow the decomposition of the state space, such that a one-step policy improvement can be executed. The idea to approximately solve a high-dimensional MDP by a one-step policy improvement algorithm based on decomposition of the state space dates back to Norman [105]. As will become clear, the way to decompose the state space is problem specific.

5.3.1 Ordering in multi-item production-inventory systems

In 1979 Wijngaard [165] discusses a production-inventory problem in which production capacity is shared for the production of multiple products. Sequential decisions concern the production volumes for each product. Since production capacity is to be shared, the decisions are dependent and should be taken integrally. Since the state description is the number of items in stock of each product, the total number of states can be very large. However, when the shared capacity is sufficiently large, the underlying MDP, in which capacity is finite and to be shared, can be approximated by an MDP model in which capacity is infinite.

The MDP with infinite capacity can be decomposed into subproblems for each product. Hence an order decision for one product can be evaluated separate of the other products. To come to an approximate solution one makes an integral ordering decision based on a finite capacity for the current decision epoch, but future decisions are evaluated as if the production capacity is not shared. The approach is thus a one-step policy improvement of the optimal policy for the case of infinite capacity.

Note that Wijngaard assumes that the products are no substitutes to each other and do not share order costs: the sole correlation between the products is through the shared capacity. A similar approach is suggested in [165] to solve the problem of ordering products in a two-echelon inventory system.

Federgruen, Groenevelt and Tijms [45] study a more complex multi-item inventory system: a fixed order cost K applies when an order is placed for any of n items (next to fixed order costs k_i for each item i). They study the common (s_i, c_i, S_i) ordering strategy (with $s_i \leq c_i < S_i$): orders are placed as soon as the inventory position of one or more items drops below their critical value s_i. For each item i one orders $S_i - x_i$ products whenever the inventory position x_i is at or below c_i. In their paper they make use of a decomposition scheme introduced by Silver [129]. By a policy improvement algorithm they search for the best values of (s_i, c_i, S_i) for each item i, independently of the other items. Through an iterative scheme the values of (s_i, c_i, S_i), for all items i, are successively adjusted to obtain a coordinated replenishment rule of the (s_i, c_i, S_i) type. Although the thus-obtained coordinated ordering policy is sub-optimal, numerical results indicate a great cost saving compared to uncoordinated ordering.

5.3.2 Routing telephone calls through a network

Another application of a one-step policy improvement algorithm is that of routing telephone calls through a network of switches as studied by Ott and Krishnan in [107]. They consider a network of n nodes or switches through which a call request between any node pair $k = (a, b)$ can be routed, with $a, b \in \{1, 2, \ldots, n\}$. The total number of node pairs is $K = n(n-1)/2$. Between node pair k maximal S_k (direct) channels are available, when all channels are occupied no call can be directed over channel k. When all pairs are connected directly, the number of possible routes from node a to b is exponential in n.

The criterion for selecting a route is to minimize the long-run average number of blocked calls for the entire system. For call requests between a node pair k, one thus needs to specify the best route to select, given the number of occupied channels for all K node pairs: $(x_1, x_2, \ldots, x_K)$. Although the problem can be formulated as an MDP, its state and action space are far too large to compute an optimal routing policy. To keep the number of occupied channels low, one wishes to make use of a low number of channels. When a direct connection is available, that connection can be used. Alternatively, the call is directed via an intermediate node. In the latter case the authors speak of a 2-hop call, since two serial channels are used.

The 1-hop routing policy that allows only direct routing has two possible actions: to block a call or to accept the call when at least one direct channel is available. The decision does not affect the acceptance or rejection of calls between other pairs, and thus the blocking rate for node pair k can be evaluated in isolation of all other node pairs. The blocking rate of the entire system is then simply the sum of the blocking rates over all node pairs. The long-run average cost function $v(x_1, x_2, \ldots, x_K)$ under the 1-hop routing policy, when starting in a given state $(x_1, x_2, \ldots, x_K)$, is separable in the cost functions $v_k(x_k)$ for each node pair k: $v(x_1, x_2, \ldots, x_K) = \sum_k v_k(x_k)$.

The separable routing policy that Ott and Krishnan formulate comes from a single policy improvement step over the 1-hop routing policy. For any call a number of possible routes (including indirect routes) is available. The evaluation of a route is done by assuming future calls either to be routed through a direct channel or to be blocked. The separable state-dependent routing policy shows a number of practical advantages over classical routing schemes, and has lead to a number of patents.

5.3.3 Assigning customers to queues

A different problem arising at telecommunication networks is addressed by Bhulai [18]. He considers a multi-skill call center, at which incoming calls have to be directed to a service group. The destination of a call is thus not fixed. Each call requires a specific skill. The agents in the call-center are grouped according to the different skill sets they possess: agents in a single-skill agent group possess a single specific skill only, more flexible agents possess multiple skills. A call can thus be handled by several agent groups, when the agents in the group possess the right skill(s).

Bhulai treats two scenarios. In the first scenario the system has no buffers, hence if a call cannot be directed to any available agent that possesses the right skill(s), the call is blocked and lost. The routing has to be such that the blocking rate is minimized. In a second scenario each agent, or group of agents with the same skill set, has an finite buffer in which calls are held when all agents are occupied. Incoming calls have to be routed directly and cannot be rerouted. The objective in this case is to route incoming calls to a queue of an appropriate agent group, such that the long-run average number of calls in the system is minimized. We deliberate a bit more on this last scenario.

For the optimal routing, one considers the state of each agent group: the number of calls either in the buffer or in service. The system's state space has as many dimensions as there are agent groups, consequently the number of possible states can easily become

very large. To come to an approximate solution, Bhulai suggests to execute a policy improvement step over a policy that allows the decomposition of the system into its agent groups. When an incoming call cannot be directed to a single-skill agent group (because its buffer is full), the call is directed to a second layer of more flexible agent groups. In the computation of the relative value function he assumes that the overflow can be approximated by a Poisson process and that the routing at the second layer happens according to fixed routing probabilities for all groups that possesses the required skill. Hence under the initial policy, the routing of a call that requires skill k depends only on the number of calls buffered at or being served by the single-skill agent group k.

Under this initial routing policy the state of an agent group is statistically independent of the state of the other agent groups. Therefore the average number of calls present at a group can be analyzed in isolation of all other groups. Relative value functions are derived based on this decomposition and under the assumption of Poisson arrivals and exponentially distributed service times. Next, these relative value functions are used in a single policy improvement algorithm: each incoming call is assigned to an agent group based on the actual number of calls in each agent group, but all future calls are assumed to be routed randomly, according to the initial policy. Numerical results demonstrate that the resulting dynamic routing policy performs quite well.

The latter problem is related to the problem of routing customers to parallel queues as studied before by Sassen, Tijms and Nobel (See Sassen et al. [124]). In their paper they extend the results of Krishnan and Ott [80] and Krishnan [79], by allowing the service times to be generally distributed. Sassen et al. assign an incoming customer to a queue according to the Bernoulli-splitting rule. With a fixed probability, p_i, a customer is assigned to queue i. First, optimal values of p_i, for $i = 1$ to n, are determined such that they minimize the long-run average sojourn time per customer. Next, the relative value of states under this splitting rule are derived in an analytical way. Finally, the separable heuristic algorithm is formulated. Numerical experiments show that the separable rule combines good performance with minor computing time.

5.4 Conclusion

The dynamic control of traffic lights at road intersections is a high-dimensional decision problems, when the number of cars waiting at each queue is taken into account. Therefore one relies on heuristics and approximate solutions to minimize the average waiting time per car at an intersection. The static control of traffic lights by a so-called fixed cycle control policy (FC) is well studied. For the dynamic control vehicle actuated control schemes are considered but hard to optimize. A number of adaptive control schemes rely on dynamic programming over a finite horizon, but the dimensionality of the problem hampers an exact approach.

Solving or circumventing the curse of dimensionality in high dimensional control systems requires a problem specific approach. For a number of applications decomposition of the state space is possible, such that a one-step policy improvement algorithm can be executed. In the examples given in Section 5.3, decomposition of the state space is possible either by modifying the problem assumptions or by imposing a special well-structured policy. Whereas the decomposition of the call routing and customer assignment problems rely on a randomized routing strategy, such a strategy may not be applicable in the control of traffic lights.

There is no universal way of finding a special strategy that allows the decomposition of the state space. Instead, it is a problem-specific approach that requires intuition and insight in the problem under consideration. If a policy does exists that allows the decomposition of the state space, a one-step policy improvement does not necessarily yields a great improvement. Nevertheless, it seems to be worth to investigate whether a good approximate policy may be obtained by a one-step policy improvement approach.