Chapter 13
Dynamic Control of Traffic Lights

Rene Haijema, Eligius M.T. Hendrix, and Jan van der Wal

Abstract Traffic lights are put in place to dynamically change priority between traffic participants. Commonly, the duration of green intervals and the grouping, and ordering in which traffic flows are served are pre-fixed. In this chapter, the problem of minimizing vehicle delay at isolated intersections is formulated as a Markov Decision Process (MDP). Solving the MDP is hampered by a large multi-dimensional state space that contains information on the traffic lights and on the queue lengths. For a single intersection, an approximate solution is provided that is based on policy iteration (PI) and decomposition of the state space. The approach starts with a Markov chain analysis of a pre-timed control policy, called Fixed Cycle (FC). The computation of relative states values for FC can be done fast, since, under FC, the multi-dimensional state space can be decomposed into sub-spaces per traffic flow. The policy obtained by executing a single iteration of Policy Iteration (PI) using relative values is called RV1. RV1 is compared for two intersections by simulation with FC, a few dynamic (vehicle actuated) policies, and an optimal MDP policy (if tractable). RV1, approximately solves the MDP, and compared to FC, it shows less delay of vehicles, shorter queues, and is robust to changes in traffic volumes. The approach shows very short computation times, which allows the application to networks of intersections, and the inclusion of estimated arrival times of vehicles approaching the intersection.

R. Haijema
Operations Research and Logistics group, Wageningen University, Wageningen, The Netherlands
e-mail: Rene.Haijema@wur.nl

E.M.T. Hendrix
Universidad de Málaga, Computer Architecture, Málaga, Spain
e-mail: eligius@uma.es

J. van der Wal
Faculty of Economics and Business, University of Amsterdam, Amsterdam, The Netherlands

R.J. Boucherie, N.M. van Dijk (eds.), *Markov Decision Processes in Practice*,
International Series in Operations Research & Management Science 248,
DOI 10.1007/978-3-319-47766-4_13

13.1 Problem

Traffic lights are introduced to resolve conflicts between road users by dynamically changing the priority to cars approaching an intersection from different directions. In practice, the underlying optimization problem is not always clearly defined by policy makers. Several objectives are possible: minimize average delay, minimize pollution by cars (e.g., CO2 emission) 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. In practice, traffic engineers aim to set a good (hopefully nearly optimal) control scheme using simulation software, queueing delay formulas, and experience. For an overview of existing methods to control road traffic, see [9].

Many road users experience traffic lights as a source of delay. This chapter presents a model for the underlying Markov Decision Process (MDP) of minimizing the expected delay or waiting time of cars. The principle of the MDP model is to dynamically adjust the traffic lights depending on the number of cars queued in each queue, and the current state of the traffic lights. Accurate information on the number of queued cars is available to the controller from magnetic loop detectors cut into the surface of the road, or other sensors or cameras positioned along the road [1, 7, 12].

A number of dynamic control policies, reported in the literature, like SCOOT and SCAT [8], dynamically switch between off-line calculated fixed cycles (FC). The approach that we present in this chapter has more freedom to choose: it does not have to choose between pre-calculated time plans; instead it requires only one FC to be calculated off-line. Another class of dynamic control is vehicle actuated (VA) control policies, that are characterized by a minimum and a maximum green period for each traffic flow, and a gap-out time to dynamically decide to end a green period. The optimization of VA policies is complicated and usually requires heuristics, because of the number of parameters to be set jointly for all traffic flow (see [14]).

Also solving an MDP for infrastructures with many traffic flows, requires heuristics solution procedures. Because of the curse of dimensionality, the number of states grows exponentially in the number of queues. Parallelization of algorithms to solve MDPs, as in [6], provides (at best) a linear reduction of the solution time, which is not enough to solve many complex infrastructures in practice. Two general techniques to solve large scale MDPs are aggregation and decomposition to reduce the state space. Decomposition has been applied successfully to other problem settings, see [13] and [2]. Another method is named Approximate Dynamic Programming, which is based on approximating the value function [10]. In this chapter, we present an approach based on decomposition.

Section 13.2 describes the MDP model. Section 13.3 describes a way to approximately solve the MDP. The resulting policy is evaluated by simulation in Sect. 13.4. A short discussion and some conclusions follow in Sect. 13.5. In Appendix of this chapter follows a summary of notations.

13.2 Markov Decision Process (MDP)

The model includes the flows of cars at an intersection, excluding other participants in traffic, such as pedestrians and public transport. The applied definition of waiting time is the time a car spends in the queue regardless of the color of the light. The corresponding optimization is modeled in terms of an MDP. Approximate solutions of the model are derived via policy iteration in Sect. 13.3. The resulting traffic control tables are then simulated in Sect. 13.4.

13.2.1 *Examples: Terminology and Notations*

To get familiar with the notation, consider the infrastructures in Fig. 13.1. Figure 13.1a depicts a simple intersection with only $F = 4$ traffic flows leading to four queues. The flows and queues are numbered clockwise: 1–4. Flows 1 and 3 are grouped into combination 1 (C_1), as they receive green simultaneously. Combination 2 (C_2) consists of flows 2 and 4. At most one combination at a time has right of way (when its lights are green or yellow). When switching from green to one combination to green to the other combination, the green lights are first changed into yellow (for two time slots) followed by a clearance period (of one time slot) during which all lights are red and after which lights of some combination are changed into green. The clearance time is included to safely clear the intersection. The recurring questions are when to end a green period, and which combination

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

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

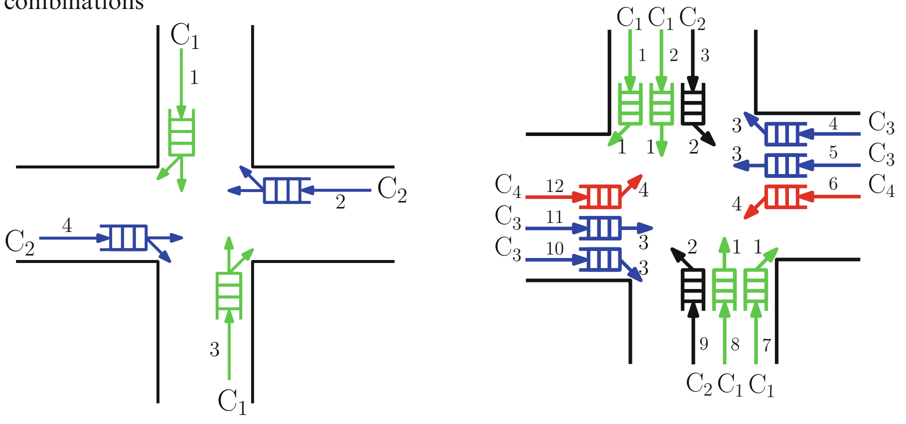


Fig. 13.1: Two typical infrastructures. (**a**) “F4C2” serve four flows in two symmetric combinations. (**b**) “F12C2” serve 12 flows in four asymmetric combinations

to serve next, given the actual color of the traffic lights, the actual number of cars present at each queue, and the probabilistic arrival process of new cars.

Figure 13.1b shows a more complex intersection where $F = 12$ flows are grouped into four combinations C_1–C_4. C_1 and C_3 consist of twice as many flows than C_2 and C_4. 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: $\mathbf{q} = (q_1, q_2, \ldots, q_{12})$. An optimal policy prescribes for each possible state of the queues, i.e. each value of the vector $\mathbf{q}$, the action to take given the current state of the lights l. Such an optimal policy can be obtained from the solution of the underlying MDP.

13.2.2 MDP Model

To model the decision problem, time is modeled in time slots of a fixed length equal to a safe traveling distance between two cars. Hence we set the length of a time slot to 2 s, which correspond roughly with the time between two departures from a queue.

13.2.2.1 State

The state (at the start of a slot) is $\mathbf{s} = (l, \mathbf{q})$, where $l \in L$ is the state of the traffic lights. S denotes the state space, which is composed of the state space of the traffic lights L and the state space related to the queue lengths. L is a finite set of $|L|$ elements. The number of cars queued is in theory unbounded. However, for computational reasons, we limit the length of each queue to $Q-1$ cars. The number of queue states is Q^F. The total number of states is $|S| = |L|Q^F$, which grows exponentially in the number of traffic flows F.

13.2.2.2 Action

The action $a \in A(\mathbf{s}) \subseteq L$, taken at the start of a slot, changes the traffic light state instantaneously, i.e. immediately after observing the state s. As switching between green for some (combination of) flow(s) to green to another, takes time to clear the intersection the choices for adjusting the lights is limited to $A(\mathbf{s}) \subseteq L$.

13.2.2.3 State Transition Probabilities

The state changes by the action choice on the traffic light, and by having cars departing or arriving at the queues. From each queue with one or more cars waiting, and that has priority according to light state l, exactly one car will leave within a time slot. Within a time slot, new cars arrive at the queues by F independent Bernoulli processes: i.e. with probability λ_f, a new car arrives at flow f (and with probability $1-\lambda_f$ no car arrives). The car either forms or joins a queue, or, if having priority and no car is queued on his lane, it crosses the intersection in the same slot without delay.

The state transition probability from state $\mathbf{s}$ to state $\mathbf{s}' = T(\mathbf{s},a) = (a,\mathbf{q}')$ taking action a is

$$P(\mathbf{q}'|\mathbf{s},a) = \prod_{f=1}^{F} p_f(q'_f|(l,q_f),a) \tag{13.1}$$

where $p_f(q'_f|(l,q_f),a)$ depends on l and whether a car arrives (w.p. λ_f) or not, as follows:

- when l implies priority (green or yellow) to flow f:

$$p_f(q'_f|(l,q_f),a) = \begin{cases} \lambda_f & \text{if } q'_f = q_f, \\ 1-\lambda_f & \text{else if } q'_f = \max\{0,q_f-1\} \\ 0 & \text{otherwise.} \end{cases} \tag{13.2}$$

- when l implies no priority (=red) to flow f:

$$p_f(q'_f|(l,q_f),a) = \begin{cases} \lambda_f & \text{if } q'_f = q_f+1 \\ 1-\lambda_f & \text{else if } q'_f = q_f \\ 0 & \text{otherwise.} \end{cases} \tag{13.3}$$

13.2.2.4 Contribution: Waiting Costs

The objective is to minimize the expected number of cars queued at the start of a time slot, such that by Little's law the expected waiting time is minimized. The contribution in a time slot to this objective function is the so-called one-period or direct cost function:

$$c(\mathbf{q}) = \sum_{f=1}^{F} q_f. \tag{13.4}$$

13.2.2.5 Bellman Equation

A stationary dynamic control policy π^* that minimizes the expected number of queued cars fulfils

$$\pi^*(\mathbf{s}) = \arg\min_{a\in A(\mathbf{s})} \sum_{\mathbf{q}'} P(\mathbf{q}'|\mathbf{s},a))v^*(\mathbf{s}'). \tag{13.5}$$

where there exists a constant g^* and a value function $\mathbf{v}^*$ being a solution of the Bellman equation:

$$\forall \mathbf{s} \in S: \quad v^*(\mathbf{s}) + g^* = c(\mathbf{q}) + \min_{a \in A(\mathbf{s})} \sum_{\mathbf{q}'} P(\mathbf{q}'|\mathbf{s}, a)) v^*(\mathbf{s}') \tag{13.6}$$

The constant g^* represents the expected number of cars waiting at the start of a time slot when an optimal policy π^* is followed.

At this point, we have to make a technical remark: to get a finite state space S, we have limited each queue length to at most $Q-1$ cars. However, in the Bellman equation we have included transitions to states $\mathbf{s}'$ that are beyond the state space. (e.g. when a car is arriving at a queue that contains already $Q-1$ cars). To determine $v(\mathbf{s}')$ for these states we apply quadratic extrapolation. For details see [3].

13.2.2.6 Computational Complexity

Solving the Bellman equation involves solving a set of $|S|$ equations in $|S|+1$ unknowns; therefore one has one degree of freedom to fix one of the elements of $\mathbf{v}^*$. Hence $v^*(\mathbf{s})$ is a relative value of state $\mathbf{s}$ when an optimal policy π^* is applied. The Bellman equation can be solved using fixed point algorithms, such as value iteration, or by exact algebraic methods. A detailed discussion of the theory and methods to solve MDPs is found in [11].

However, solving the MDP is only possible for infrastructures with a 'small' number of traffic flows, and under non-saturated conditions such that a reasonable bound to the queue lengths can be set. For example, for infrastructure F12C4, when all 12 queues may have up to 9 cars, then each element q_f can take 10 possible values (0–9). Consequently, the number of possible vectors $\mathbf{q}$ is 10^{12}. Hence the state space of the MDP easily exceeds the computationally acceptable limit of say 10 million states. The computational limit can be extended a bit by parallelization [6]. However, that only partly solves the computational issue, as the number of states grows exponentially with the number of traffic flows F. Large MDPs that cannot be solved to optimality require an approximate solution method, like the one that we discuss in the next Section.

13.3 Approximation by Policy Iteration

13.3.1 Policy Iteration (PI)

Instead of applying exact algebraic methods to solve Eq. (13.6), we apply a policy iteration (PI) algorithm to approximate an optimal policy. PI successively repeats the following two steps.

Step 1 *Policy evaluation step:* for an initial policy π, determine for all $\mathbf{s} \in S$ the associated relative state values $v^{\pi}(\mathbf{s})$ that satisfy:

$$v^{\pi}(\mathbf{s}) + g^{\pi} = c(\mathbf{q}) + \sum_{\mathbf{q}'} P(\mathbf{q}'|\mathbf{s}, \pi(\mathbf{s})))v^{\pi}(\mathbf{s}'). \tag{13.7}$$

Step 2 *Policy improvement step:* Next, policy π is improved by executing a policy improvement step:

$$\pi'(\mathbf{s}) = \arg\min_{a \in A(\mathbf{s})} \sum_{\mathbf{q}'} P(\mathbf{q}'|\mathbf{s}, a))v^{\pi}(\mathbf{s}'). \tag{13.8}$$

if $\pi' = \pi$, then stop (as $\pi = \pi^*$), otherwise set $\pi := \pi'$ and return to Step 1.

Just as any other methods to solve the Bellman equations, PI suffers from the computational burden due to the large state space. The advantage of PI is that doing only one iteration may already give a good approximation when the initial policy is reasonably good.

13.3.2 Initial Policy: Fixed Cycle (FC)

A well studied policy is a static policy for which one pre-fixes the time intervals at which flows get green and the cyclic order in which combinations of flows are served. The cycle has a fixed length of D time units, which is the sum of the green periods and the time period needed to switch between combinations. Such a policy we call FC. The slots within a cycle are numbered $t = 1, 2, \ldots, D$. The slot number provides all relevant information about the state of the traffic lights: i.e. from t one can derive for each flow f the color of the light, the time it takes till getting green, yellow, or red. Thus the state of the traffic light l is for $\pi = \text{FC}$ given by t.

FC could act as an initial policy for PI. Therefore one first needs to configure FC: i.e. set D, (nearly) optimal lengths of the green periods, and the cyclic order in which combinations get priority. An optimization algorithm for setting an initial FC is presented in [3] and [4].

13.3.3 Policy Evaluation Step of FC

The relative state values of FC, $v^{FC}(\mathbf{s})$, can be decomposed in relative state values $v_f^{FC}(t, q_f)$ per traffic flow f:

$$v^{FC}(\mathbf{s}) = \sum_{f=1}^{F} v_f^{FC}(t, q_f). \tag{13.9}$$

Relative state values $v_f^{FC}(t,q_f)$ are determined by value iteration:

Step 1a. Define $V_0^f(t,q) = 0$ for all $t \in \{1,\ldots,D\}$ and $q \in \{0,\ldots,Q\}$ and let ε take a very small value (compared to the expected number of cars waiting at the start of any time slot), e.g. 10^{-5}.

Step 1b. For all $t \in \{1,\ldots,D\}$ and $q \in \{0,\ldots,Q\}$, recursively compute $V_{n+1}^f(t,q)$ as follows (with $V_n^f(D+1,\cdot)$ read as $V_n^f(1,\cdot)$):
Start with $n=0$ and $\mathbf{V}_0 = \mathbf{0}$.
Repeat

- if flow f is having priority (green or yellow) during time slot t:

$$V_{n+1}^f(t,q) := q + \lambda_f \cdot V_n^f(t+1,q) + (1-\lambda_f)\cdot V_n^f(t+1,(q-1)^+), \tag{13.10}$$

where $x^+ = \max\{0,x\}$,
- if flow f is not having priority during time slot t (its light is red):

$$V_{n+1}^f(t,q) := q + \lambda_f \cdot V_n^f(t+1,q+1) + (1-\lambda_f)\cdot V_n^f(t+1,q)\,. \tag{13.11}$$

- $n := n+1$;

until $\max(\mathbf{V}_n^f - \mathbf{V}_{n-D}^f) - \min(\mathbf{V}_n^f - \mathbf{V}_{n-D}^f) < \varepsilon$.
Set $N := n$.
For all $t \in \{1,\ldots,D\}$ and $q \in \{0,\ldots,Q\}$, the difference $(V_N^f(t,q) - V_{N-D}^f(t,q))$ is at most ε off from the long-run average cost per cycle ($g^{(f)}$), as the D-step Markov chains are all irreducible.

Step 1c. We set the relative state values relative to a fixed reference state, $(D,0)$. Thus $\mathbf{v}_f^{FC}$ is:

$$\mathbf{v}_f^{FC} \equiv \frac{\mathbf{V}_{N-D+1}^f + \cdots + \mathbf{V}_N^f}{D} - \frac{V_{N-D+1}^f(D,0) + \cdots + V_N^f(D,0)}{D} \cdot \mathbf{1}\,, \tag{13.12}$$

where $\mathbf{1}$ is the all-ones vector.
A discussion of the relative value definition in (13.12) is found in [3]. The differences $V_N^f(q,t) - V_N^f(D,0)$ cannot be used for this purpose as FC is a periodic policy and consequently these differences change periodically with N.
An alternative criterion is to compare $\sum_{d=0}^{D-1} \mathbf{V}_{N-d}^f / D$ against the average cost over N slots:

$$\mathbf{v}_f^{FC} \equiv \frac{\mathbf{V}_{N-D+1}^f + \cdots + \mathbf{V}_N^f}{D} - N \cdot g\,, \tag{13.13}$$

where g may be approximated by any element of $\frac{\mathbf{V}_N^f - \mathbf{V}_{N-D+1}^f}{D}$, as N is sufficiently large.

For example, Fig. 13.2 shows for a particular queue state $\mathbf{q} = (4,2,2,1)$ at infrastructure F4C2, the valuation of the traffic light state t. Figure 13.2a shows for each of the four flows, the individual preference of time slot t. Figure 13.2b shows the sum of these preferences. On top of the x-axis (which shows the time slots of FC), labels are added that indicate which of the two combination gets green (G1/G2), or yellow (Y1/Y2), or whether all lights are red (R).In this case with $\mathbf{q} = (4,2,2,1)$, the best time slot is slot 1 (i.e. first slot of green to combination 1), as it gives the lowest relative (cost) value. FC loops over all time slots: after slot 12 (R=all red) it continues at slot 1 (G1=green to combination 1).

13.3.4 Single Policy Improvement Step: RV1 Policy

The relative value $v_f^{FC}(\tau, q_f)$ quantifies the preference of flow f for time slot τ, when q_f cars are waiting at queue f. Assuming all flows are equally important, the overall relative appreciation of time slot τ is the sum $\sum_{f=1}^{F} v_f^{FC}(\tau, q_f)$, as depicted in Fig. 13.2b.

By applying a single policy improvement step, one finds a new policy π' that may interrupt or breaks FC by dynamically deciding to decrease or increase a green period. That is, in state $(t, \mathbf{q})$, policy π' switches the state of the traffic light to the best time slot reachable from the current slot t:

$$\pi'(t, \mathbf{q}) = \arg \min_{\tau \in A(t,\mathbf{q})} \sum_{f=1}^{F} v_f^{FC}(\tau, q_f) \;, \tag{13.14}$$

where $A(t, \mathbf{q})$ is the set of time slots to which one may switch safely from the current traffic light state t.

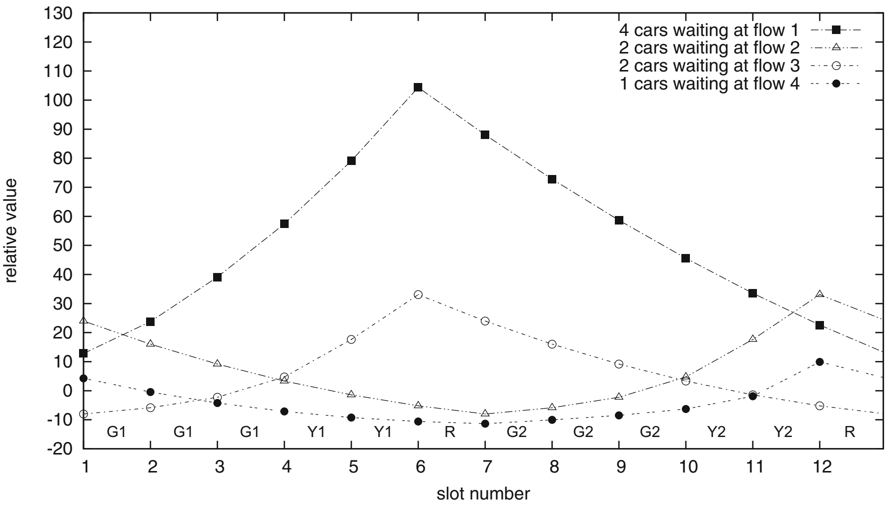
This process is illustrated by Fig. 13.2b: one selects the time slot that can be reached from the current time slot t, and that yields the lowest sum of relative values. That is, under cyclic control, if $t = 1$, 2, or 3, then $\pi'(t, \mathbf{q}) = 1$. If $t = 9$ or 10, then $\pi'(t, \mathbf{q}) = 10$. For $t = 8$ holds $\pi'(t, \mathbf{q}) = 9$, as lights cannot switch from red to yellow without granting first green to one combination. For all other values of t holds $\pi'(t, \mathbf{q}) = t$, i.e., one may not interrupt FC during switching.

We call this policy in the rest of this chapter the RV1 policy as it follows from a 1-step policy improvement using the relative values of FC.

13.3.5 Computational Complexity of RV1

The computational complexity of determining RV1 is very low as the state space under FC is effectively decomposed into the state per traffic flow. The computation of the relative values for each flow can be done quickly off-line. It allows a high

(a) Relative value curves for flows 1 to 4 when $\mathbf{q} = (4,2,2,1)$ cars are waiting.



(b) Sum of the relative value curves when $\mathbf{q} = (4,2,2,1)$ cars are waiting at flow 1 to 4.

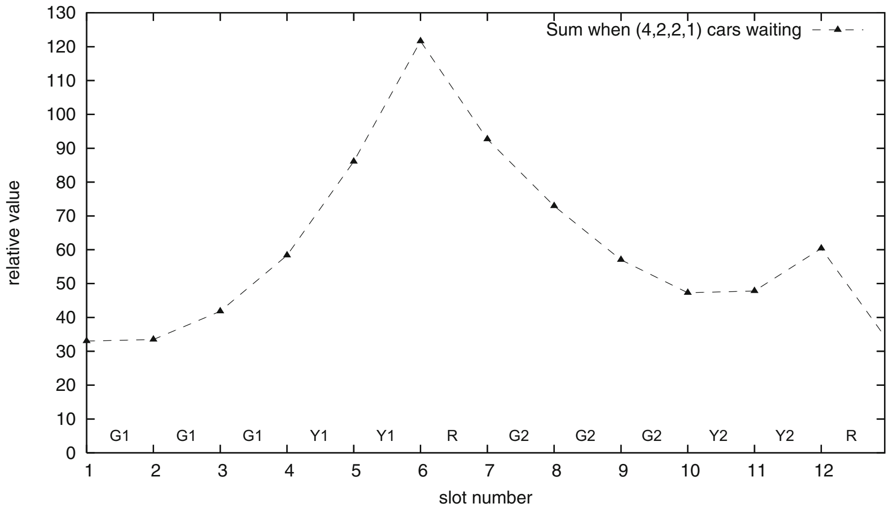


Fig. 13.2: Relative state values of FC. (**a**) Relative value curves for flows 1–4 when $\mathbf{q} = (4,2,2,1)$ cars are waiting. (**b**) Sum of the relative value curves when $\mathbf{q} = (4,2,2,1)$ cars are waiting at flow 1–4

bound on the queue length that is practically not restrictive. The computation of the sum of relative value curves is to be done quickly online: based on the actual number of cars queued at each queue and the actual state of the traffic lights, the best time slot to jump to can be computed in real time using Eq. (13.14). This is an important characteristic that allows applying RV1 to large and complex infrastructures.

13.3.6 Additional Iterations of PI

For problems with a large state space S, additional iterations of PI are not considered, since this is not possible to determine relative state values for policy RV1: i.e. $v^{RV1}(\mathbf{s})$ cannot be computed in reasonable time, as its state space cannot be decomposed. For problems with a small state spaces that allows additional iterations of PI (without decomposition of the state space), one may continue PI by following the procedure sketched in Sect. 13.3.1 until an optimal MDP policy is found.

13.4 Results

To assess the quality of RV1, its performance is evaluated by simulation and compared against FC and some exhaustive control policies. Exhaustive control (XC) grants green to a combination as long as cars are queued. Policy XC-1 switches to yellow as soon as at each ‘green’ queue at most one car is waiting anticipating its departure during the next (=first) yellow slot. XC-2 switches to yellow as soon as at each ‘green’ queue (at most) two cars are waiting, and thus XC-2 anticipates their departure during the next two yellow slots.

For both infrastructures F4C2 and F12C4 we consider symmetric cases: i.e. the arrival rates are identical for all flows. For F4C2 an optimal MDP policy is evaluated. For F12C4, the optimal MDP policy could not be determined due to the large state space. Acyclic control, non-identical arrival rates, and other infrastructures are reported in [3] and [5].

13.4.1 Simulation

The simulation model relies on the same assumptions as the MDP model. The accuracy of the results presented in the next subsections is primarily set by the number of simulation runs and the length of each run. All reported simulation results are based on 100 runs of 72,000 slots per run, corresponding to 4000 h in the real system. At the start of each run, a warming-up period of 450 slots (= 15 min) is applied. The reported mean waiting times are accurate up to (at least) 2–3 digits. To be concise, we do not report confidence intervals.

13.4.2 Intersection F4C2

Table 13.1 shows the results for the fully-symmetric F4C2 intersection at varying workloads ρ. In the cases of a workload of $\rho = 0.4, 0.6, 0.8$ all flows have identical arrival probabilities of respectively $\lambda_f = 0.2, 0.3, 0.4$. The workload ρ should be well below 1, to accommodate time for switching between combinations.

The Optimize-fixed-cycle algorithm (see [3, 4]) suggests cycle lengths of 8, 12 and 22 slots respectively, as reported in the next-to-last row (in seconds). The effective green time of a combination is the length of the related green and yellow period under FC. RV1 is based on FC with these cycle lengths and effective green times.

Table 13.1: Overall mean waiting time (in s) for the fully-symmetric F4C2

Rule	$\rho = 0.4$		$\rho = 0.6$		$\rho = 0.8$	
RV1	5.06		7.01		14.2	
FC	5.43	+7%	8.27	+18%	17.0	+20%
XC	5.76	+14%	8.82	+26%	19.9	+40%
XC-1	5.03	−1%	7.21	+3%	15.5	+9%
XC-2	5.09	+1%	7.31	+4%	14.2	+0%
MDP cyclic	4.89	−3%	6.95	−1%	13.5	−5%
FC cycle length (in s) and	16		24		44	
effective green times per combination	(6, 6)		(10, 10)		(20, 20)	

The performance of the cyclic RV1 strategy obtained by the one-step policy improvement algorithm, is close to that of the optimal cyclic MDP strategy. Next to the average waiting times, we report the relative difference compared to RV1. The cyclic MDP policy performs only slightly better. FC and XC yield on average 15 and 27% larger waiting times; when the load is high ($\rho = 0.8$) the biggest differences can be observed. On average, RV1 is just slightly better than the anticipating exhaustive variants (XC-1 and XC-2). However, the differences are small for this simple fully-symmetric case.

13.4.3 Complex Intersection F12C4

For infrastructure F12C4, the optimal MDP strategy cannot be computed, because the number of states is prohibitively large. Even if the MDP model would truncate queues at the unrealistic level of two cars, the total number of states is still quite large: $8.5 \cdot 10^6$ states (= $(1+2)^{12}$ queue states times 16 traffic light states). F12C4 is not only computationally more complex because of the number of states, but also because the combinations are asymmetric: C_1 and C_3 consist of four flows, whereas C_2 and C_4 consists of two flows only. It seems to be more profitable to under-serve combinations C_2 and C_4, since serving the other two combination results in more departures per time slot as long as cars are present at the respective queues.

The asymmetry in the number of flows per combination, makes it more difficult to define simple rules that perform well. We keep the same definition of exhaustive control: all queues must be empty before the green signals are turned into yellow. Under XC-2, green lights are turned into yellow as soon as at each of the flows that have right of way less than three cars are queued.

In Table 13.2 the results for varying workloads at a F12C4 intersection are presented for the case where all flows have identical arrival intensities $\lambda_f = 0.1$, 0.15, and 0.2, providing a workload of $\rho = 0.4$, 0.6 and 0.8 respectively. RV1 outperforms all other strategies. When the workload of the intersection is low (0.4) XC-2 performs equally well. At a high load XC-2 is too simplistic. At $\rho = 0.8$, FC performs even better than XC-2: XC-2 yields an average waiting time that is 28% higher than under RV1.

Table 13.2: Overall mean waiting time (in s) at different loads for F12C4 ($\lambda_f = \rho/4$)

Rule	$\rho = 0.4$		$\rho = 0.6$		$\rho = 0.8$	
RV1	13.5		19.3		41.8	
FC	15.0	+11%	23.7	+23%	50.5	+21%
XC	19.2	+42%	33.4	+73%	89.8	+115%
XC-1	14.9	+10%	25.1	+30%	70.1	+68%
XC-2	13.5	+0%	19.6	+2%	53.3	+28%
FC cycle length (in s)	32		40		88	
FC effective green times (in sec.)	(6, 6, 6, 6)		(8, 8, 8, 8)		(20, 20, 20, 20)	

Equal Allocation of Waiting Time

In Table 13.3, we study the waiting time at the different flows under a heavy workload of $\rho = 0.8$. Although the arrival rates λ_f are identical for all flows, the mean waiting time differs per combination. Combinations C_1 and C_3 are 'thicker' than combinations C_2 and C_4, since the latter two combinations have only two flows each, whereas C_1 and C_3 consists of four flows each. Therefore C_1 and C_3 experience a lower waiting time than combinations C_2 and C_4 under all policies except FC. The average waiting time per car under FC is identical for each flow as each flow experiences an effective green time of 20 s (10 slots) per cycle.

Table 13.3: Mean waiting times (in s) for symmetric F12C4 at $\rho = 0.8$

Rule	*EW* overall		*EW* C_1, C_3	*EW* C_2, C_4
RV1	41.8		37.4	50.6
FC dep. times 20, 20, 20, 20 s	50.5	+21%	50.5	50.4
XC	89.8	+115%	88.5	92.4
XC-1	70.1	+68%	68.9	72.4
XC-2	53.3	+28%	52.1	55.8

Although FC seems to be most fair in the sense that the flows experience similar average waiting times, RV1 performs much better: the average waiting time to flows of C_1 and C_3 is 13 s (or 26%) lower than under FC, while the average waiting times to C_2 and C_4 are virtually the same as under FC. Notice further that both XC and anticipative-exhaustive control (XC-1 and XC-2) perform worse than FC, when the workload is high.

13.5 Discussion and Conclusions

The dynamic control of traffic lights can be formulated as an MDP. In practice, for many infrastructures the state space may be too large to determine an optimal MDP policy. Nevertheless, this chapter shows that the principles and theory of MDP can still be applied to obtain good approximate solutions by executing a single iteration of a policy improvement (PI) algorithm. Key to this approach is to start PI with a well structured policy for which relative values of the state can be determined. For the problem of controlling traffic lights, such a well-structured policy is fixed cycle control (FC). Under FC the computation of relative values can be decomposed in computing relative values of the traffic light state.

For a single intersection, an approximate solution is provided that is based on policy iteration (PI) and decomposition of the state space. The approach starts with a Markov chain analysis of a pre-timed control policy, called Fixed Cycle (FC). The computation of relative states values for FC is fast as under FC the multi-dimensional state space can be decomposed into sub-spaces per traffic flow. The policy obtained by executing a single iteration of PI using relative values of FC, is called RV1.

Numerical results obtained by simulation, shows RV1 greatly reduces the average waiting time compared to FC (and other policies). As the relative values of states under policy RV1 cannot be computed using decomposition, additional PI steps can be executed only for infrastructures for which the state space is not too large.

RV1 seems to be a promising policy for practical application as:

- RV1 is robust to changes in traffic volumes: RV1 performs well even when the underlying FC is sub-optimal,
- RV1 is thus easy to maintain: if traffic conditions change the performance of RV1 deteriorates less rapidly than FC,
- RV1 is fast: evaluating a change of the traffic lights is done online in a micro second; also the off-line calculation of the relative values for each flow can be done quickly,
- RV1 can be extended to include information on the predicted arrival time of a car approaching the intersection,
- RV1 can be scaled up to complex intersection and networks of intersections.

Future research may be devoted to the above aspect. For bringing RV1 to practice, it is relevant to include other vehicles and traffic flows, such as public transport, bicycles and pedestrians. These additional traffic flows do not hamper the use of RV1 policies. In addition, multiple criteria, like vehicle speed and CO2 emissions, can be included. This chapter has shown that insights and approximations obtained using an MDP framework are of practical importance to improve the control of traffic lights.

Acknowledgements This work has been funded by grants from the Spanish state (TIN2015-66680-C2-2-R) and, Junta de Andalucía (P11-TIC-7176), in part financed by the European Regional Development Fund (ERDF).

Appendix: Notation

This section summarizes the notation used for defining the MDP: i.e. for respectively, the state, action, transition probability, and value function.

$\mathbf{s}$	$= (l, \mathbf{q} = (l, q_1, q_2, \dots, q_F)$, with $l \in L$ = state of traffic light, and $q_f \in \{0, 1, \dots, Q-1\}$ is # cars queued at lane f
F	= Number of traffic flows (=lanes)
$a \in A(\mathbf{s}) \subseteq L$	= Change of traffic lights in state $\mathbf{s}$
$P(\mathbf{q}'\|\mathbf{s}, a)$	$= \prod_{f=1}^{F} p_f(q'_f\|(l, q_f), a)$ = Probability that next period's state is
$\mathbf{s}'$	$= (a, \mathbf{q}')$, when current state is $\mathbf{s}$ and action a is taken
$p_f(q'_f\|(l, q_f), a)$	= Single-period transition probability related to lane f.
λ	= probability a car arrives at lane f
$c(\mathbf{q})$	= Costs per period
$v^*(\mathbf{s})$	= Relative valuation of state s of an optimal policy
g^*	= Gain of Markov chain for an optimal policy
	= Long-run average costs per period of an optimal policy
$\pi^*(\mathbf{s})$	= Optimal change of traffic lights in state $\mathbf{s}$

References

1. L. Baskar, B. De Schutter, J. Hellendoorn, Z. Papp, Traffic control and intelligent vehicle highway systems: a survey. IET Intell. Transp. Syst. **5**(1), 38–52 (2011)
2. S. Bhulai, Dynamic routing policies for multiskill call centers. Probab. Eng. Inf. Sci. **23**(01), 101–119 (2009)
3. R. Haijema, Solving large structured Markov decision problems for perishable inventory management and traffic control. Ph.D. thesis, Univeristy of Amsterdam, Tinbergen Institute, Amsterdam School of Economics, 2008
4. R. Haijema, E.M. Hendrix, Traffic responsive control of intersections with predicted arrival times: a Markovian approach. Comput. Aided Civ. Infrastruct. Eng. **29**(2), 123–139 (2014)
5. R. Haijema, J. van der Wal, An MDP decomposition approach for traffic control at isolated signalized intersections. Probab. Eng. Inf. Sci. **22**(4), 587–602 (2008)
6. J.F. Herrera, E.M. Hendrix, L.G. Casado, R. Haijema, Data parallelism in traffic control tables with arrival information, in *Euro-Par 2014: Parallel Processing Workshops* (Springer, Berlin, 2014), pp. 60–70
7. H.X. Liu, A. Danczyk, Optimal sensor locations for freeway bottleneck identification. Comput. Aided Civ. Infrastruct. Eng. **24**(8), 535–550 (2009)
8. J.Y.K. Luk, Two traffic-responsive area traffic control methods: SCAT and SCOOT. Traffic Eng. Control **25**, 14–22 (1984)
9. M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, Y. Wang, Review of road traffic control strategies, in *Proceedings of the IEEE*, vol. 91 (IEEE, New York, 2003), pp. 2043–2067
10. W.B. Powell, Approximate Dynamic Programming: Solving the Curses of Dimensionality. Wiley Series in Probability and Statistics (Wiley, New York, 2007)
11. M.L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (Wiley, New York, 2014)
12. A. Stathopoulos, L. Dimitriou, T. Tsekeris, Fuzzy modeling approach for combined forecasting of urban traffic flow. Comput. Aided Civ. Infrastruct. Eng. **23**(7), 521–535 (2008)
13. J. Wijngaard, Decomposition for dynamic programming in production and inventory control. Eng. Process. Econ. **4**, 385–388 (1979)
14. D. Zhao, Y. Dai, Z. Zhang, Computational intelligence in urban traffic signal control: a survey. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **42**(4), 485–494 (2012)