UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

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

Haijema, R.

Publication date
2008

Link to publication

Citation for published version (APA):
Haijema, R. (2008). *Solving large structured Markov Decision Problems for perishable inventory management and traffic control*. Thela Thesis.

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:18 Jan 2022

Chapter 6

Single intersection in isolation

In this chapter we focus on the dynamic control of a signalized intersection in isolation. We introduce some new control policies based on an MDP formulation. By simulation the new policies are tested and compared to other control policies for a number of isolated intersections. Some signalized intersections are part of a network, but often the control of the lights can be done for each intersection in isolation. The approach is extended in the next chapters. In Chapter 8 we consider networks of intersections.

6.1 Modeling the problem at a single intersection

In Section 5.1.1 we have introduced the problem of minimizing the overall average waiting time at a single intersection in isolation. Our definition of waiting time is the time spent in the queue, hence the time between joining a queue and passing the stopping line. Throughout this (first) chapter we assume cars to have zero length to ease the analysis. This assumption corresponds to vertical queueing. Our definition of waiting time corresponds then to the definition of waiting time that is common in queueing theory. In the Chapters 7 and 8, we do acknowledge queued cars to take space. As we will see this slightly affects the waiting times reported by the simulation model since then cars sooner hit the tail of a queue, when the queue length increases.

In this section we formulate the problem in more detail, introduce the notations that we use in this and the next chapters and we discuss some modeling assumptions.

6.1.1 Infrastructure

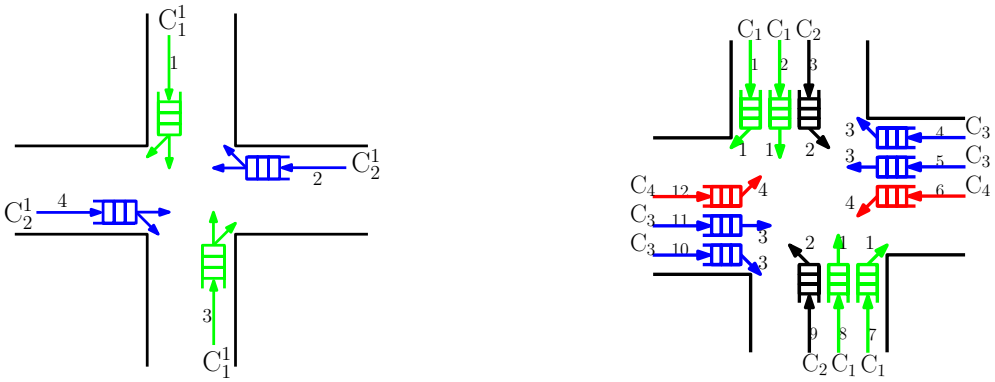
Figure 8.4 shows two typical cases that we are interested in. We refer to these cases by code names: F4C2 for an intersection of $F = 4$ traffic flows grouped in $C = 2$ combinations and F12C4 an intersection with $F = 12$ flows grouped into $C = 4$ combinations. The set of F flows is thus divided in C disjoint subsets $\mathcal{C}(1), \mathcal{C}(2), \ldots, \mathcal{C}(C)$, called *combinations*. In We abbreviate specific queues and combinations: Q_f refers to the queue of cars in traffic flow f and C_s abbreviates *combination* s consisting of flows $\mathcal{C}(s)$. The flows, queues, and combinations are numbered clockwise (rather than using the more detailed notation that is common in traffic engineering). Both F4C2 and F12C4 consist of four *approaches* or directions from which traffic comes. We refer to these directions by North, East, South, West, as if the top of each figure faces North. North and South are *opposite approaches*, and so are East and West.

At F4C2 each approach consists of a single lane. Cars are not allowed to turn left and thus flows 1 and 3 are not in conflict and constitute C_1, flows 2 and 4 constitute C_2. Ongoing or through-traffic is referred to as *'thru' traffic*. At F12C4 each of the four approaches consists of three separate lanes for right-turning, thru, and left-turning traffic. The left-turning traffic of two opposite approaches are served at the same time. The right-turning and thru traffic of two opposite approaches are served together in a combination of four flows. The 12 flows are thus grouped in 4 combinations (C_1 to C_4). C_1 and C_3 are composed of four flows each, whereas C_2 and C_4 have only two flows:

$$\mathcal{C}(1) = \{1, 2, 7, 8\}, \quad \mathcal{C}(2) = \{3, 9\}, \quad \mathcal{C}(3) = \{4, 5, 10, 11\}, \quad \mathcal{C}(4) = \{6, 12\}.$$

We assume that these combinations are given and fixed. The order in which the combinations are served (=get right of way) maybe cyclic or acyclic. Under cyclic control the combinations are visited in the order: 1, 2, 3, 4, 1, etcetera.

Summary of notations – An intersection consists of F flows of car arrivals; each flow f has a single lane and a single queue Q_f. The vector $\mathbf{q} = (q_1, \ldots, q_F)$ stores the numbers of cars waiting in each queue. The set of all flows $\mathcal{F} = \{1, \ldots, F\}$ is partitioned into C disjoint subsets $\mathcal{C}(1) - \mathcal{C}(C)$ of combinations of non-conflicting flows that will receive green, yellow and red together. We call a subset of flows a combination (of flows). The set of the C combinations, $\mathcal{S} = \{1, \ldots, C\}$, is given and hence not part of the optimization problem. Flows in the same combination will always receive green, yellow and red together. If one combination has green or yellow, all other lights show red. We assume that *conflicting traffic flows* are not part of the same combination.



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

(b) "F12C4" serve 12 flows in 4 asymmetric combinations

Figure 6.1: Two base-case infrastructures.

6.1.2 Discrete time modeling assumptions

The (automated) controller of the traffic lights periodically takes decisions about how to adjust the lights. When switching from green for one set of flows to green for another set, a fixed switching time is to be acknowledged. This suggests to model the problem in *discrete time* which calls for some simplifications of the processes.

Discrete time

The time unit or slot is taken to be the (average) time a car needs to pass the intersection when the light is green. We fix the slot length to two seconds, which corresponds also to a safe driving distance between cars at different speeds. When car drivers keep all the time a safe driving distance of exactly 2 seconds, then either 0 or 1 cars passes the stopping line within a slot.

Signal process

Changing from green for one combination to green for another combination of flows takes ($\gamma =$) 3 slots (6 seconds): ($\beta =$) 2 slots of yellow and ($\gamma - \beta =$) 1 slot in which all lights show red to clear the intersection. This *switching time* is assumed to be independent of the two sets of flows involved. In addition a minimum green time, of say 1 slot, applies such when a combination is served it signals shows green for at least 1 slot. (In our model all slots are of identical length. Although notations become more involved, the approach can be extended to different slot lengths and to different, flow-dependent switch-over times.)

Arrival process

In the basic model of this chapter cars are not observed until they arrive at the stopping line at which they are queued vertically. The position of the tail of a queue corresponds thus with the stopping line, as if cars take zero length. It is assumed that the arrival of cars at the queues occur uniformly over time rather than in platoons that may be formed at any upstream signalized intersection. Arrivals at different queues and in different slots are independent. Per flow the number of car arrivals in a slot is either 0 or 1. The probability of an arrival in flow f is λ_f: the arrival rate at queue f. (Instead of a Bernoulli arrival process one could assume Poisson arrivals but having more than 1 car arrival within a slot at a single queue does not happen when cars keep a safe traveling distance of one-slot.)

Departure process

When a signal shows green or yellow and as long as cars are present, every slot (exactly) one car passes the stopping line in a deterministic fashion. Remember car drivers acknowledge a safe driving distance of 1 slot. In countries where yellow officially means 'stop-when-safe-to-do-so' it might be more realistic to say that during the first yellow slot the probability that a car passes is still (close to) 1, but during the second yellow slot it is less than one [1]. The model allows for this, as it allows for any Markovian stochastic departure process, but notations would become more involved. Assuming deterministic departure processes with an inter-departure time of 1 slot (2 seconds) is quite common in the practice of traffic engineering ([119]).

In the cases that are studied, conflicting traffic flows do not get simultaneously right of way. In practice conflicting flows might have green at the same time, in which case (regular) basic traffic rules apply. As will be discussed at the end of this chapter, the models we present can be extended at this point.

Queueing process

A car arriving at an empty queue that has right of way passes the stopping line without delay. Therefore we assume new arrivals to take place at the beginning of the slot (just after observing the state of the queues and the lights). When one or more cars are queued a new arrival joins the end of the queue. Departures from the queues take place at the end of the slot prior to the observation of the new state. In Figure 6.2 we depict the modeling of the course of events and actions.

[1] In most countries, the sequence is red (stop), green (go), amber (prepare to stop). In the UK and Canada, amber officially means 'stop' (unless it would cause an accident to do so), but in practice amber is treated as 'prepare to stop'. [Source: http://en.wikipedia.org/wiki/Traffic_light.]

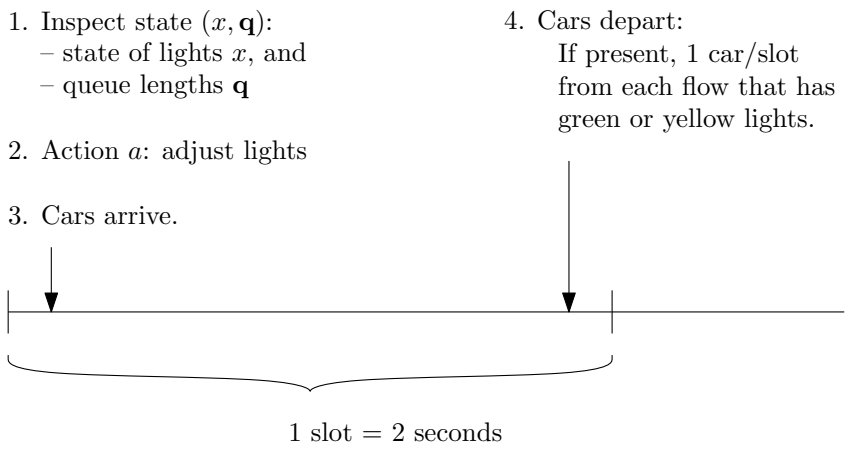


Figure 6.2: The queueing process: modeling the course of events in the MDP model

Workload

The workload that a flow brings to an intersection is the (long-run) fraction of time the light of flow f to serve all cars. Since a departure takes exactly 1 slot. The workload of flow f equals λ_f. The workload ρ_s that combination s brings to the intersection is the maximum of λ_f over all flows part of that combination:

$$\rho_s = \max_{f \in \mathcal{C}(s)} \lambda_f. \tag{6.1}$$

The workload of the intersection is the sum of the workload over all combinations:

$$\rho = \sum_{s=1}^{C} \rho_s = \sum_{s=1}^{C} \max_{f \in \mathcal{C}(s)} \lambda_f \,. \tag{6.2}$$

The workload ρ does not include the all-red time needed to clear an intersection.

We restrict our attention to under-saturated cases where queues are stable: $\rho < 1$.

Remark Note that the effective workload is in general higher than ρ, since the effective workload includes the fraction of time all lights are red for switching from green to one combination to green to another combination. For example, consider the F4C2 case, with $\lambda_f = 0.3$ for all flows. From (6.2) it follows that the workload is $\rho = 0.3 + 0.3 = 0.6$. When FC grants green for 3 slots to C_1 and for another 3 slots to C_2, then the cycle length equals 12 slots. During a single cycle on average $0.3 \cdot 12 = 3.6$ cars arrive at each queue. On average 7.2 slots in a complete cycle are actually used for and two slots are

used for switching between the two combinations to clear the intersection. The effective workload is thus $9.2/12 \approx 0.77$. The residual $12 - 9.2 = 2.8$ slots are used to deal with the uncertainty in car arrivals; since cars arrive at different flows independently some queues of a combination may be empty while cars depart from other queues of that combination.

Remark – Thickness of a combination – The workload ρ_f of a combination can be used to find an indication of how much green time a combination requires. In addition, we define the *'thickness' of combination* s: $\sum_{f \in \mathcal{C}(s)} \lambda_f$. The thickness of a combination is the mean number of cars that arrives at the intersection by flows in combination s. (The thickness of flow f is simply λ_f.

6.1.3 Outline

In the next section the problem is formulated as an MDP. For large intersections the number of possible queue states becomes tremendously large, prohibiting straightforward optimization of the MDP. Starting with a (nearly) optimal fixed cycle policy, a one-step policy improvement approach is proposed in Section 6.3. The approach is demonstrated and tested in Section 6.4 assuming cyclic control. Acyclic control and more advanced rules are introduced in Section 6.5 and tested in Section 6.6. Finally we report some conclusions in Section 6.7.

6.2 Formulation as a Markov Decision Problem

As argued the problem is modeled in discrete time. We refer to Figure 6.2 for the order in which we model the inspection of the state, the change of the lights, and the arrivals and departures of cars. The states are inspected at the beginning of a slot, next an action is selected and accordingly the lights are adjusted. Cars arrive at the beginning of a slot and may leave the very same slot, when the lights show green or yellow and no cars are waiting in front of it. At the end of a slot a number of cars have left their respective queues. Waiting costs are incurred over the number of cars present at the start of a slot; a car that arrives and passes the stopping line in the very same slot does not contribute to the cost over that slot since it has not experienced waiting time.

To formulate the MDP model, next the states, the decisions, the transitions and the costs are specified in detail.

6.2.1 States

The arrival and departure processes are assumed to be Markovian (i.e. memoryless), so the state of the traffic flows and related queues is described by the vector $\mathbf{q} = (q_1, \ldots, q_F)$, with q_f the number of cars in flow f present at the beginning of a slot. For the moment we do accept an infinite state space, in Section 6.2.6 we discuss the truncation of the queues in the MDP model.

The state space is completed by the state of the lights x. To allow both cyclic and acyclic control, we define x as a tuple (s, i), which indicate that the lights of combination s:

- are green when $i = 0$,
- are at the first yellow slot when $i = 1$,
- are at the second yellow slot when $i = 2$,
- are red as all other lights when $i = 3$.

Note that the 'all red' state $(s, i = 3)$ also marks that combination s was most-recently served. The total number of signal states x is thus $4C$. For acyclic control this number can be reduced to $3C + 1$, since then a single 'all red' state suffices.

The states at the start of a slot are thus denoted by the vector $(x, \mathbf{q}) = ((s, i), \mathbf{q})$.

6.2.2 Decisions

Decisions are taken at the beginning of a slot and executed instantaneously. Thus, if a decision grants green to a combination, cars of that combination can leave in the very same slot. The set of feasible decisions one may consider in state $((s, i), \mathbf{q})$ depends at least on the traffic light state (s, i) and maybe also on the number of queued cars $\mathbf{q}$.

- If the lights are green for combination s, i.e. if $i = 0$, there are two feasible decisions:
  - keep the lights as they are, or
  - change from green to the first yellow slot to combination s.
- When the lights show yellow there is no choice, since the signaling process may not be interrupted:
  - At the end of a first yellow slot continue to the second yellow slot,
  - After the second yellow the only decision is to change into red for all flows.
- If all lights are red the feasible decisions are
  - to keep all lights red (if all queues are empty this might be optimal because switching takes 3 slots), or
  - to give green to one of the combinations. If the cyclic order is to be kept, we only allow to skip a combination if all queues for that combination are empty. When skipping a combination the cycle is resumed with the next non-empty combination: as a result of skipping combination 2 two successive green periods of combination 2 are far apart: 1, **2**, 3, 1, 3, 1, **2**, 3, etc.

The decision space in state $(x, \mathbf{q}) = ((s, i), \mathbf{q})$ is thus:

$$\mathcal{A}((s,i),\mathbf{q}) = \begin{cases} \{(s,0),(s,1)\} & \text{if } i = 0 \quad \text{(lights of } s \text{ show green)}, \\ (s, i+1) & \text{if } i = 1,2 \quad \text{(lights of } s \text{ show yellow)}, \\ \{(s,3),(s',0)\} & \text{if } i = 3 \quad \begin{array}{l}\text{(all lights show red,} \\ s' \in \mathcal{S}\text{, but under cyclic control} \\ s' = \text{next non-empty combination)}.\end{array} \end{cases}$$

Remark – According to the given definition of $\mathcal{A}((s,i),\mathbf{q})$, the action space does not depend on $\mathbf{q}$. In the cases on which we report in this thesis, the action space does depend on the queue lengths as follows. To allow a fair comparison of the MDP policy against exhaustive control, we copy a special action that applies when no cars are present at the queues. Under exhaustive control, we suggest to freeze the signals when no cars are queued. The state of the lights (if not yellow) are frozen when the system is empty: green lights stay green and red lights stay red during the coming slot.

In some cases this may be optimal, but it is suboptimal in general. We do not bother about the optimality of this action, since, in the cases that we are interested in, it happens rarely that all queues are empty at the same time.

6.2.3 Transition probabilities

Action a implies an instantaneous change of the lights from state x into state a. If action a implies red to flow f, then the transition probability with respect to the queue length are

$$p_f(q_f, q_f|a) = 1 - \lambda_f, \quad \text{and} \quad p_f(q_f, q_f + 1|a) = \lambda_f. \tag{6.3}$$

If, instead, flow f receives, as a result of action a, green or yellow during the coming slot, the transition probabilities for the length of queue f are given by (with $y^+ = \max\{y, 0\}$):

$$p_f(q_f, (q_f - 1)^+|a) = 1 - \lambda_f, \quad \text{and} \quad p_f(q_f, q_f|a) = \lambda_f. \tag{6.4}$$

Note that in Equations (6.3) and (6.4) we have assumed that within a slot 0 or 1 car arrives at each queue, and, if present, 1 car departs from each queue whenever the lights shows green or yellow.

The transition probability from state $(x, \mathbf{q})$ to state $(a, \mathbf{q}')$, denoted by $p(x, \mathbf{q}; a, \mathbf{q}')$, is simply the product of the $p_f(q_f, q'_f|a)$'s:

$$p(x, \mathbf{q}; a, \mathbf{q}') = \prod_{f=1}^{F} p_f(q_f, q'_f|a) \ . \tag{6.5}$$

6.2.4 Criterion and costs

The criterion is the minimization of the long-run average waiting time per car. According to Little's law this is equivalent to minimizing the average number of cars waiting at all queues. Therefore we apply a linear cost structure with one unit of costs for every car present at the beginning of a slot. Let $c(\mathbf{q})$ denote the one slot costs or direct costs in state $(x, \mathbf{q})$, then

$$c(\mathbf{q}) = \sum_{f=1}^{F} q_f \ . \tag{6.6}$$

Note that waiting costs are incurred no matter the light shows green or yellow. The cost structure thus relates to the time spent in the queue assuming queued cars to have length 0. When cars have non-zero length and are queued horizontally, as in the next chapters, then waiting time (in slots) is measured also for queued cars that move forward one position per slot when the lights show green or yellow.

6.2.5 Successive approximations

In order to obtain an optimal policy, one may apply an SA algorithm. Therefore let $V_n(x, \mathbf{q})$ denote the expected minimal costs over an horizon of n slots when starting in state $(x, \mathbf{q})$. In principle the following DP relation holds between $V_{n+1}(x, \mathbf{q})$ and $V_n(\cdot, \cdot)$ for all states $(x, \mathbf{q})$:

$$V_{n+1}(x, \mathbf{q}) = c(\mathbf{q}) + \min_{a \in \mathcal{A}(x, \mathbf{q})} \sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}')\, V_n(a, \mathbf{q}') \ . \tag{6.7}$$

Since the state space is infinite, (6.7) cannot be used directly in an SA algorithm. Therefore the state space is truncated to a finite one. To compensate for any errors so-called externality costs are defined.

6.2.6 Reduction to a finite state space and Externality costs

In the MDP model we truncate each queue to a maximum of Q cars, such that all computations can be done in finite time. The bound on the queue length Q is preferably high enough such that overflow happens rarely. In the SA algorithm one thus only considers state vectors $\mathbf{q}$ with all elements q_f in $\{0, 1, \ldots, Q\}$.

In the evaluation of the transition from $\mathbf{q}$ to $\mathbf{q}'$, arrivals at queues that are full are rejected in the MDP model. In the real system cars cannot be rejected: q'_f may exceed Q for one or more queues f. When a car is not admitted at queue f in the MDP model, one needs to compensate for the waiting time and the additional delay the car would give to all other cars: already present or still to arrive in the real system. Therefore so-called *externality costs*, $E(a, \mathbf{q}')$, are included in the (expected) direct cost function.

Let $\mathbb{E}C(a, \mathbf{q})$ denote the expected cost to incur in state $(a, \mathbf{q})$. $\mathbb{E}C(a, \mathbf{q})$ is the sum of the direct waiting costs $c(\mathbf{q})$ and the expectation over the externality costs $E(a, \mathbf{q}')$:

$$\mathbb{E}C(a, \mathbf{q}) \equiv c(\mathbf{q}) + \sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}')\, E(a, \mathbf{q}'). \tag{6.8}$$

When the externality costs $E(a, \mathbf{q}')$ are known or approximated, one may apply successive approximations using Equation (6.9) for all states $(x, \mathbf{q})$ with $\mathbf{q} \leq Q \cdot \mathbf{1}$:

$$V_{n+1}(x, \mathbf{q}) = \min_{a \in \mathcal{A}(x, \mathbf{q})} \left(\mathbb{E}C(a, \mathbf{q}) + \sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}')\, V_n(a, \mathbf{q}^{\mathrm{T}}) \right), \tag{6.9}$$

where $q_f^{\mathrm{T}} \equiv \min\{q'_f, Q\}$.

A simple expression or approximation for the externality cost $E(a, \mathbf{q})$ is only available for some queueing models [61], but is not present for the complex queueing system that we are dealing with. A wrong approximation of the externality costs may affect the optimal strategy. On the one hand, when these costs are estimated too high, the optimal policy tries to stay away from the border Q and thus the systems tends to switch too early from green to yellow. On the other hand, when one underestimates the costs, it may be optimal to serve a queue that is 'full', only when all other queues are empty. We thus need a numerical scheme to (accurately) approximate the externality costs. A relatively easy way, which we explain below, is to approximate the costs by extrapolation of the value function in an SA algorithm.

Computation of externality costs by SA algorithm

Under an optimal policy for the stationary infinite horizon MDP, the externality costs are by definition of the value vector $\mathbf{V}_n$:

$$E(a, \mathbf{q}') = \lim_{n\to\infty} \left[V_n(a, \mathbf{q}') - V_n(a, \mathbf{q}^{\mathrm{T}}) \right], \tag{6.10}$$

where $\mathbf{q}^{\mathrm{T}}$ is the truncation of the vector $\mathbf{q}'$: $\mathbf{q}^{\mathrm{T}} \equiv (\min\{q'_1, Q\}, \ldots, \min\{q'_F, Q\})$. In addition we define the overflow, $\mathbf{q}^{\mathrm{O}} \equiv \mathbf{q}' - \mathbf{q}^{\mathrm{T}}$.

This definition suggest to approximate $E(a, \mathbf{q}')$ in iteration n of an SA algorithm by

$$E_n(a, \mathbf{q}') \equiv V_n(a, \mathbf{q}') - V_n(a, \mathbf{q}^{\mathrm{T}}) \tag{6.11}$$

Clearly, $E_n(a, \mathbf{q}')$ is 0, whenever all $q'_f \leq Q$. Since $V_n(a, \mathbf{q}')$ is unknown when q'_f exceeds Q for one or more flows f, $V_n(a, \mathbf{q}')$ is estimated by quadratic extrapolation based on the three values $V_n(a, \mathbf{q}^{\mathrm{T}} - 2\mathbf{q}^{\mathrm{O}})$, $V_n(a, \mathbf{q}^{\mathrm{T}} - \mathbf{q}^{\mathrm{O}})$, and $V_n(a, \mathbf{q}^{\mathrm{T}})$. In Appendix E one reads how to extrapolate multi-dimensional tabulated functions. Here we simply state the resulting formula for quadratic extrapolation of $\mathbf{V}$:

$$V_n(a, \mathbf{q}') \approx 3 \cdot V_n(a, \mathbf{q}^{\mathrm{T}}) - 3 \cdot V_n(a, \mathbf{q}^{\mathrm{T}} - \mathbf{q}^{\mathrm{O}}) + V_n(a, \mathbf{q}^{\mathrm{T}} - 2\mathbf{q}^{\mathrm{O}}), \tag{6.12}$$

Note that this extrapolation is feasible when $\mathbf{q}^{\mathrm{T}} - 2\mathbf{q}^{\mathrm{O}}$ contains no negative elements, i.e. when $\forall f : q_f^{\mathrm{O}} \leq \frac{1}{2}Q$). In the SA algorithm the overflow is at most 1 per flow, since at most 1 car arrives per flow. Furthermore, quadratic extrapolation works well, since the direct cost structure is linear in $\mathbf{q}$.

In (6.12) one reads the n-th approximation of the direct externality costs, $E_n(a, \mathbf{q}')$, when making a transition from $\mathbf{q}$ to $\mathbf{q}^{\mathrm{T}}$ instead to $\mathbf{q}'$:

$$E_n(a, \mathbf{q}') = 2 \cdot V_n(a, \mathbf{q}^{\mathrm{T}}) - 3 \cdot V_n(a, \mathbf{q}^{\mathrm{T}} - \mathbf{q}^{\mathrm{O}}) + V_n(a, \mathbf{q}^{\mathrm{T}} - 2\mathbf{q}^{\mathrm{O}}). \tag{6.13}$$

Since in the infinite horizon MDP, the direct cost must be stationary, the approximation of $E(a, \mathbf{q}')$ is fixed after a large number of, say M, iterations. For the dynamic control of traffic lights M is set to 100 slots (or more than 3 minutes in real time).

6.2.7 An SA algorithm with extrapolation of Externality costs

A complete successive approximation scheme to compute an optimal policy is the following:

Step 1. First, select a sufficiently large value of M, such that the externality costs are approximated over an horizon of sufficient length of M slots.
Next, for $n = 0$ to $M-1$ successively compute for all states $(x, \mathbf{q})$ with $\mathbf{q} \leq Q \cdot \mathbf{1}$:

$$V_{n+1}(x, \mathbf{q}) = c(\mathbf{q}) + \min_{a \in \mathcal{A}(x,\mathbf{q})} \left(\sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}') \; \left[E_n(a, \mathbf{q}') + V_n(a, \mathbf{q}^{\mathrm{T}}) \right] \right) \tag{6.14}$$

starting with $V_0(x, \mathbf{q}) = 0$, and $E_n(a, \mathbf{q}')$ as defined in (6.13).

Step 2. Store the resulting value vector $\mathbf{V}_M$ and compute the expected direct cost:

$$\mathbb{E}C(a, \mathbf{q}) = c(\mathbf{q}) + \sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}') \cdot E(a, \mathbf{q}') \tag{6.15}$$

using the following approximation of the externality costs

$$E(a, \mathbf{q}') \approx 2 \cdot V_M(a, \mathbf{q}^{\mathrm{T}}) - 3 \cdot V_M(a, \mathbf{q}^{\mathrm{T}} - \mathbf{q}^{\mathrm{O}}) + V_M(a, \mathbf{q}^{\mathrm{T}} - 2\mathbf{q}^{\mathrm{O}}). \tag{6.16}$$

Step 3. Starting with $n = M$ and $\mathbf{V}_M$ the value vector lastly computed in Step 1., recursively compute for all states $(x, \mathbf{q})$ with $\mathbf{q} \leq Q \cdot \mathbf{1}$:

$$V_{n+1}(x, \mathbf{q}) = \min_{a \in \mathcal{A}(x,\mathbf{q})} \left(\mathbb{E}C(a, \mathbf{q}) + \sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}') \; V_n(a, \mathbf{q}^{\mathrm{T}}) \right) \tag{6.17}$$

until the $span(\mathbf{V}_{n+1} - \mathbf{V}_n)$ is sufficiently small, say smaller than ϵ.

Suppose this happens at the N-th iteration. (One may show that convergence is guaranteed as the system empties every now and then, so that all states communicate.)

Step 4. If no optimizing actions are stored in Step 3., then a (nearly) optimal stationary strategy π is obtained from:

$$\pi(x, \mathbf{q}) = \arg \min_{a \in \mathcal{A}(x,\mathbf{q})} \left(\mathbb{E}C(a, \mathbf{q}) + \sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}') \; V_{N+1}(a, \mathbf{q}^{\mathrm{T}}) \right). \tag{6.18}$$

The average costs under π can be approximated by any element of $\mathbf{V}_{N+1} - \mathbf{V}_N$, which estimates the minimal long-run average number of cars in the system.

6.2.8 Computational complexity

Since we have made the state space finite, in principle the computations in (6.14) to (6.18) can be done in finite time. However, the multi-dimensionality of the system (one dimension per flow) usually leads to tremendous state spaces so that in general a straightforward dynamic programming approach is not possible in a reasonable amount of time. The computational complexity is primarily set by the number of queue states, which is exponential in the number of flows F. When no more than Q cars are admitted to each queue, then the number of queue states is $(1+Q)^F$. The number of traffic light states is $C \cdot (1+3)$ (under cyclic control).

When the workload of the intersection is not so high, a reasonable bound on the queue lengths might be 9 cars. Then, for an intersection with 12 flows, one has to consider thus 10^{12} queue states. When the flows are served in 4 combinations, the total number of states, including the $4 \cdot (1+3)$ traffic light states, thus becomes $1.6\,10^{13}$, which is far too many to solve the MDP in a reasonable time. Another limitation in implementing the successive approximation scheme may be the available RAM memory: for all states $(x, \mathbf{q})$ with $\mathbf{q} \leq Q \cdot \mathbf{1}$ at least the values of $V_n(x, \mathbf{q})$, $V_{n+1}(x, \mathbf{q})$ need to be stored.

Although we compensate in the MDP model for overflow of the queues, Q must be high enough such that overflow happens rarely. Furthermore, Q should be high enough since the SA algorithm derives in principle no optimal actions for states were one or more queues do overflow. When the overflow is limited to at most $\frac{1}{2}Q$, nearly-optimal actions can be derived by extrapolation of the value function using known values of the value vector. The quality of such estimates cannot be guaranteed.

6.3 One-step policy improvement (RV1)

Clearly the computational complexity of the MDP makes it practically impossible to solve many real-life MDPs. Therefore we suggest to apply a one-step policy improvement approach that is based on the decomposition of the state space. As outlined in Chapter 5, over the years such an approach has been applied successfully to a number of multi-dimensional decision problems (see [105], [165], [80], [81], [17], [18], and [124]).

Approach

The one-step policy improvement approach improves an initial policy for which the relative values are known. Under FC the relative values of the states can be determined after the decomposition of the state space. Therefore FC is taken as the initial strategy that will be improved through a one-step policy improvement algorithm. The following four steps are distinguished in our approach:

Step 1. Obtain a good FC:
Find a FC with cycle length D and effective green times d_s that results in a low overall long-run average waiting time per car. Therefore an incremental search algorithm is used in which the average waiting costs per car is computed flow-by-flow by evaluation of the underlying Markov chains. The Markov chain is thus decomposed into F periodic Markov chains, one for each flow.

Step 2. Derive the relative values of the D slots within the fixed cycle:
For each of the F Markov chains, one computes the relative values of the states under FC. Summing the relative values for all flows for given numbers of queued cars $\mathbf{q}$, one obtains the relative appreciation of each slot within the cycle.

Step 3. Set an improvement rule (RV1):
Formulate a one-step policy improvement rule using the relative values of the states under FC. We call the resulting policy RV1, with RV for relative value.

Step 4. Evaluate RV1 by Simulation:
Finally, RV1 is evaluated by simulation, and compared to a number of basic traffic light control strategies and, where possible, to the optimal MDP policy.

The four steps are discussed in greater detail in the next subsections. The simulation results of Step 4 are reported in Section 6.4.

6.3.1 Step 1 – Fixed Cycle control (FC)

The reason for choosing FC as initial control strategy is that the relative values of states under FC can be computed flow-by-flow. In the next subsection the numerical computation of the relative values is discussed for a single flow. In this section we focus on finding a good configuration of FC that results in a low average waiting time per car.

The Optimal-fixed-cycle algorithm

At this point we are just interested in setting a good or (nearly) optimal fixed cycle. For this we use a simple incremental search algorithm: the *Optimal-fixed-cycle algorithm*, which is reported in detail in Appendix D.2. The search starts with calling the *Minimum-cycle-length algorithm* to find a cycle of minimum length by which all queues are just stable under FC: all queues get assigned just enough green and yellow slots to be stable. The Minimum-cycle-length algorithm is reported in Appendix D.1. A configuration of FC is evaluated by numerically solving a set of F Markov chains. How each Markov chain (MC) is formulated and how it is solved is discussed in the next section.

The Optimal-fixed-cycle algorithm increases successively the cycle length D by 1 slot, according to the best increase of the effective green period for one of the combinations. This process of incremental search is stopped when the last-so-many increases did not yield a better FC. The best FC so far is considered as the (locally) optimal FC.

Numerical example

In Figure 6.3, the search process of the Optimal-fixed-cycle algorithm is illustrated for the F12C4 intersection of Figure 6.1(b)) with identical arrival rates ($\forall f : \lambda_f = 0.2$). The Minimum-cycle-length algorithm as reported in Appendix D.1 indicates that the cycle length must be at least 24 slots: during 4 different slots all lights show red. The effective green time is 5 slots for each combinations: the lights show green for 3 slots and yellow for 2 slots.

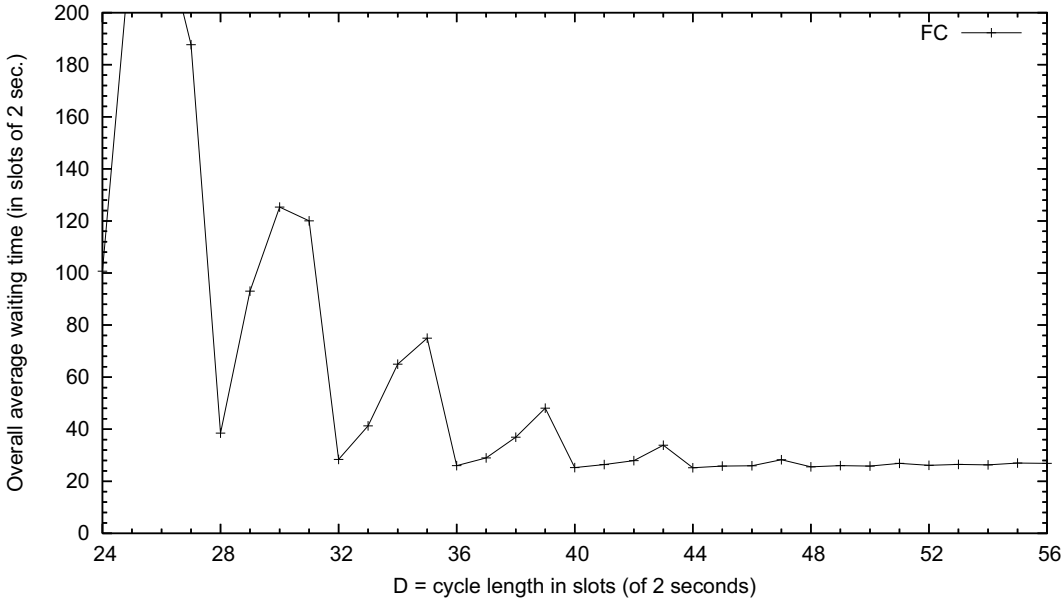
The Optimal-fixed-cycle algorithm successively increases the cycle length, starting with a cycle of length $D = 24$ slots. The length of the green period for a single combination is extended by one slot, such that the cycle length becomes 25 slots. Consequently, the red period for all other combinations increased by 1 slot, which may cause some queues to become instable. Whereas all queues are (just) stable when $D = 24$, some queues are

instable at D is 25, 26, and 27 slots[2]. At $D = 28$, the best FC is the one where the green periods of all four combinations are extended by 1 slot compared to the FC with $D = 24$.

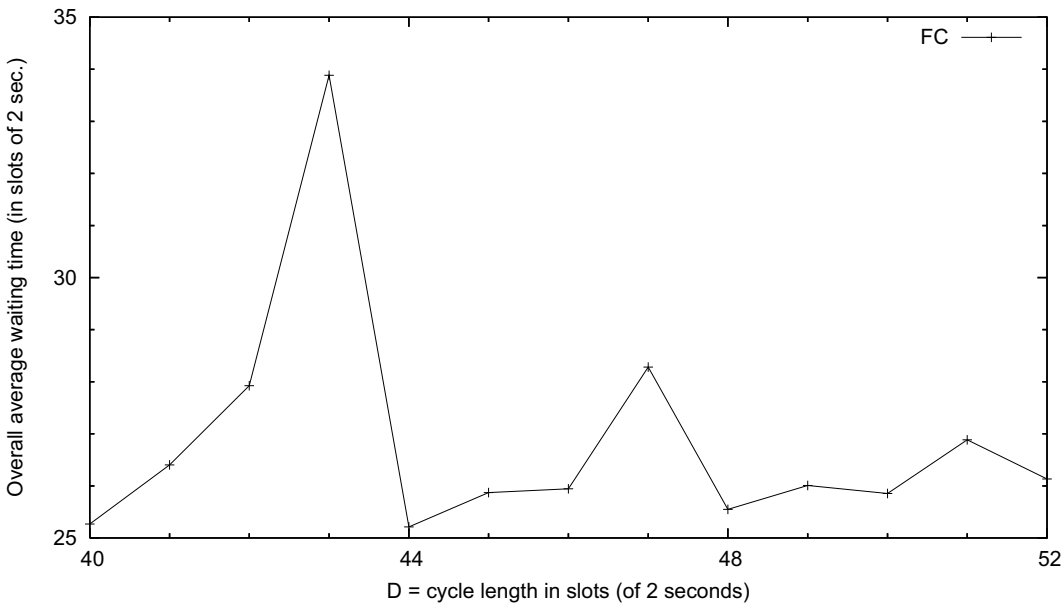
The long-run average waiting times reported in Figure 6.3 are for the best FC given cycle length D. The figures are accurate up to 4-5 decimals and obtained by numerically solving MCs. Clearly the response curve, depicting the relation between the long-run overall average waiting time and the cycle length, is far from convex and contains many local optima. When one would only connect the local optima in Figure 6.3(a) one obtains a curve that is quite flat for high values of D. Apparently it is better to end-up with a FC with a relatively high cycle length, than with a FC with a cycle length that is relatively short. In Figure 6.3(b) we have zoomed to show that cycle length 40 and 44 give the best results, although 52 slots still yields a low average waiting time. We deliberate on the sensitivity with respect to the chosen cycle length in Section 6.4.5.

Since in this example the arrival rates are identical, a locally optimal FC appears to have green periods of identical length. Locally optimal FCs are thus found for D being a multiple of $C = 4$. This is not the case when the arrival rates differ, as will be illustrated in Figure 8.12 in Chapter 8. This complicates the stopping criterion of the search algorithm. The incremental search algorithm can be stopped when k successive increments do not yield an improved FC. When all arrival rates are identical, k can be set to C. But, when some combination(s) have a higher workload than others, it may be necessary to set k higher depending on the skewness in the arrival rates. We do not guarantee the algorithm to end in the global optimum, but numerical experiments have shown that the Optimal-fixed-cycle algorithm generates a good FC.

[2]In the evaluation of a Markov chain for an instable queue the iterative process is cut-off at some maximum number of iterations.



(a) Sensitivity of the waiting time under FC when the cycle length ranges from 36 to 56 slots.



(b) Sensitivity around the optimal value of the cycle length.

Figure 6.3: Application of the Optimal-fixed-cycle algorithm to find optimal value of the cycle length for F12C4 with identical arrival rates ($\lambda_f = 0.2$). (The average waiting times are computed by solving a Markov chain for each flow. For an instable queue the iterative process is cut-off at some maximum number of iterations.)

6.3.2 Step 2 – Relative values of FC

Under FC each flow perceives a periodic green-yellow-red cycle of fixed length, independent of the queue lengths of the various flows. Let D be the duration of a cycle or the *cycle length*. Then the *effective green time* or the *departure time* of flow f is, under FC, a fixed number of d_f slots during which the lights of flow f show green or yellow. The last yellow slot is followed by $r_f = D - d_f$ red slots. Since cars depart from a queue only during green or yellow slots, we call these d_f slots *departure slots*.

Thus for flow f the fixed cycle results in a periodic MC with d_f departure slots and r_f red slots. The state of the chain can be described as a pair (t, q) with q the number of cars present at the beginning of a slot and t the number of the slot within the cycle, $t \in \{1, \ldots, D\}$. Whereas in the MDP model the traffic light state was $x = (s, i)$, the state of the lights under FC is simply the slot number t.

The equilibrium distribution, and thus the average costs per time unit, can be computed using a generating function approach, see Darroch [33] and Van Leeuwaarden [153]. But what we need are the relative values of the states to compare some time slot τ against slot t. The analysis of such a periodic MC and in particular the computation of these relative values can be done numerically by stochastic dynamic programming (SDP). As the state space for one flow is small (only 2-dimensional), one may set the bound on the queue length Q high enough such that the probability of rejecting an arrival plays hardly any role.

The total number of feasible states in each of the F MCs is $D \cdot (1+Q)$, which is considerably less than the number of states in the MDP model. Hence the relative values of states under FC can be computed quickly by value iteration. The immediate costs in state (t, q) is simply the number of cars present: q. In addition, one may incur expected externality costs to compensate for the error involved with the truncation of the queue lengths at Q. Although externality costs are computed in the programs developed, we omit the computation of any externality cost in the discussion below: when Q is set very high, such that the number of queued cars in the real systems stays virtually always well below Q, externality costs play hardly any role.

Let $v_n^f(t, q)$ be the total expected costs (=number of cars waiting) at queue f over n slots when starting in (t, q). For the evaluation of the MC for a single flow and for the computation of the relative values of states, one proceed as follows.

Computation of relative values under FC

For all flows $f = 1, \ldots, F$ execute the following four steps: Step 2a – Step 2d.

Step 2a. Define $v_0^f(t,q) = 0$ for all t and q and let ϵ take a very small value, e.g. 10^{-10}.

Step 2b. For all t and q recursively compute $v_{n+1}^f(t,q)$ as follows:

- if t is a departure slot for f:

$$v_{n+1}^f(t,q) = q + \lambda_f \cdot v_n^f(t+1,q) + (1-\lambda_f) \cdot v_n^f(t+1,(q-1)^+), \quad (6.19)$$

- if t is a 'red slot' to flow f:

$$v_{n+1}^f(t,q) = q + \lambda_f \cdot v_n^f(t+1,q+1) + (1-\lambda_f) \cdot v_n^f(t+1,q) \ , \quad (6.20)$$

  where $v_n^f(D+1,\cdot)$ should be read as $v_n^f(1,\cdot)$.

until $span(\mathbf{v}_{n+D}^f - \mathbf{v}_n^f) < \epsilon$. Suppose this happens at first at the N-th iteration. For all t and q the difference $(v_{n+D}^f(t,q) - v_n^f(t,q))$ converges exponentially fast to the long-run average costs per cycle $(g^{(f)})$, as the D-step Markov chains are all irreducible.

Step 2c. Compute the long-run average waiting time (EW_f) at queue f in slots by applying Little's law: $EW_f = EL_f/\lambda_f$, with $EL_f =$ the long-run average number of cars present at the start of a slot:

$$EL_f = \frac{(v_{N+D}^f(t,q) - v_N^f(t,q))}{D}.$$

The overall average waiting time (EW) is simply the weighted sum:

$$EW = \frac{\sum_f \lambda_f \cdot EW_f}{\sum_f \lambda_f} = \frac{\sum_f EL_f}{\sum_f \lambda_f} = \frac{\sum_f (v_{N+D}^f(t,q) - v_N^f(t,q))/D}{\sum_f \lambda_f} \quad (6.21)$$

Step 2d. Choose a fixed reference state, $(D, 0)$, and compute the relative value vector $\mathbf{v}^f_{rel}$:

$$\mathbf{v}^f_{rel} \equiv \frac{\mathbf{v}^f_N + \cdots + \mathbf{v}^f_{N+D-1}}{D} - \frac{v^f_N(D,0) + \cdots + v^f_{N+D-1}(D,0)}{D} \cdot \mathbf{1} \ , \quad (6.22)$$

where $\mathbf{1}$ is a vector with all elements equal to 1.

Given the reference state $(D, 0)$, the relative values are unique. As discussed in Section 1.3.2 and Appendix B, the relative value definition in (6.22) is correct. The differences $v^f_N(q,t) - v^f_N(D,0)$ cannot be used for this purpose as FC is a periodic policy and consequently these differences change periodically with N.

Example – Consider the F4C2 intersection of Figure 6.1(a) where 4 flows are served in two combinations: flows 1 and 3 constitute C_1, and flows 2 and 4 constitute C_2. Suppose both combinations get green during 3 slots. Switching takes another 3 slots, so the duration of a cycle is 12 slots: $D = 12$. A typical relative value curve of flow 1 is depicted in Figure 6.4 for the case that 4 cars are waiting at queue 1. Slots 1 to 5 are departure slots for flow 1, the next 7 slots the signals of flow 1 show red. The worst position in the cycle is the start of slot 6, since all 4 cars have to wait for at least 7 slots. Apparently, the best position is the start of slot 1, because then the remaining green period last longest.

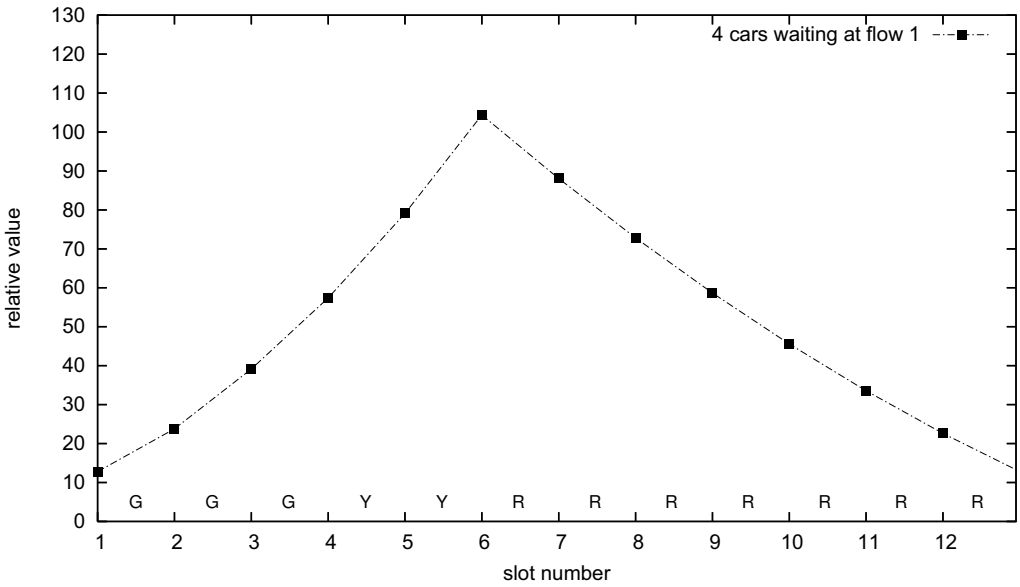


Figure 6.4: Relative value curve when 4 cars are waiting at flow 1.

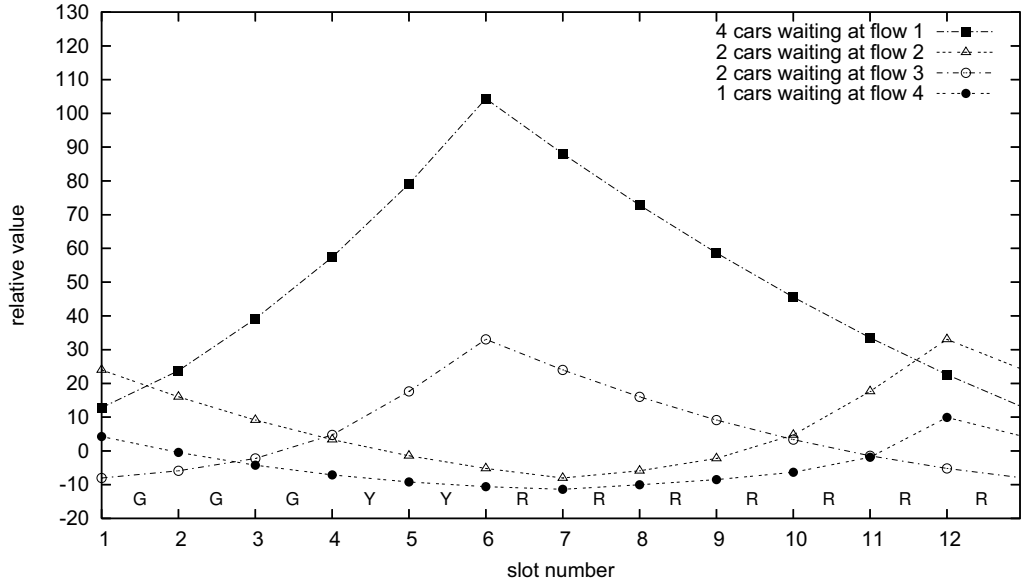


Figure 6.5: Relative value curves for flows 1 to 4 when $\mathbf{q} = (4, 2, 2, 1)$ cars are waiting.

Similar curves can be drawn for all flows and any number of cars in $\{0, 1, \ldots, Q\}$. In Figure 6.5 the relative value curves for all four flows are drawn when $\mathbf{q} = (4, 2, 2, 1)$ cars are waiting. Each flow has it own optimal slot number, which is usually at (or close to) the start of its green period. The curves of flow 2 and flow 4 show the difference in relative values between having 2 cars queued and only 1 car queued.

6.3.3 Step 3 – One-step policy improvement rule (RV1)

Now let us return to the system as a whole. Under FC the system's state is described by the pair $(t, \mathbf{q})$, with $\mathbf{q} = (q_1, \ldots, q_F)$ the numbers of queued cars and t the slot within the cycle. Related to t are the colors of the lights. In the example of the previous section, $t = 1$ is the first green slot of C_1, $t = 4$ is the first yellow slot of C_1, and $t = 7$ is the first green slot to C_2. Note that $t = 6$ and $t = 12$ relate to an all-lights-red slot. Slot 13 does not exist, since after slot $t = D = 12$ the cycle resumes at $t = 1$. During any slot t at most one combination has green or yellow, all other combinations have red.

Under FC the slots are thus visited in the order $1, 2, \ldots, D,\ 1, 2, \ldots$. When this order is interrupted, the state and color of the lights are changed accordingly. Now, let us return to the numerical example. Suppose $t = 2$ at the start of a slot (hence C_1 has right of way), then one may change the state of the light t from 2 into 4, such that the green period of C_1 is ended. In fact by allowing the dynamic adjustment of the state of the traffic lights

t, one may choose to resume FC at some slot τ rather than at slot t. The decision that one selects immediately affects the traffic lights and has indirectly impact on the number of cars waiting at each queue.

The one-step policy improvement rule looks for the best time jump within the cycle from t to τ. In state $(t, \mathbf{q})$, only time jumps to a slot $\tau \in \mathcal{T}(t, \mathbf{q})$ are allowed. The set $\mathcal{T}(t, \mathbf{q})$ contains all feasible time jumps when $\mathbf{q}$ cars are waiting at the start of slot t. $\mathcal{T}^{\mathrm{C}}(t, \mathbf{q})$ denotes the action space when the policy has to be cyclic. The action spaces are similar to $\mathcal{A}(x, \mathbf{q})$ in the MDP model. When t is a green slot to C^s, one may choose any τ that keeps the lights green to C^s, or to the τ that indicates the first yellow slot to C^s. The action space consists of a single element, i.e. t, when t refers to the start of the second yellow slot for some flow or to the start of an all-red slot. When t indicates that a new green period is about to start for combination s, one has to distinguish cyclic from acyclic control:

- Under cyclic control any τ that grants green to combination s, or set τ to $t-1$ such that the all-red period is extended by 1 slot;
- under acyclic control any τ that grants green to any combination may be selected, next to slots that imply red to all flows.

Selection of best time jump

In state $(t, \mathbf{q})$ every $\tau \in \mathcal{T}(t, \mathbf{q})$ is evaluated by summing the relative values related to τ: $\sum_{f=1}^{F} v_{rel}^{f}(\tau, q_f)$. Immediately thereafter, the lights are adjusted according to the value of τ that minimizes the sum of the relative values under FC. When evaluating a time jump, one assumes that after the jump FC is resumed at slot τ, although this will not be the case when decisions are revised at the start of every slot.

Hence the one-step optimal decision (time jump) in state $(t, \mathbf{q})$, follows from:

$$\sigma(t, \mathbf{q}) = \arg \min_{\tau \in \mathcal{T}(t, \mathbf{q})} \sum_{f=1}^{F} v_{rel}^{f}(\tau, q_f) \ . \tag{6.23}$$

These $\sigma(t, \mathbf{q})$ for all states $(t, \mathbf{q})$ together constitute the improved strategy RV1 that is hopefully close to optimal. Since a decision is updated every slot, RV1 breaks FC: the green periods under RV1 are not of a fixed length, but their lengths change dynamically.

Example – Revisit the previous example, in Section 6.3.2. To illustrate the concept, we show in Figure 6.6 the sum of the relative value curves over all flows with $\mathbf{q} = (4, 2, 2, 1)$ cars are waiting at flows 1 – 4. The best position in the cycle (yielding the lowest sum of relative values) is slot 1: the start of green to C_1. Although switching to this point seems to be best, it is not reachable from all points in the cycle. When C_2 has green the best feasible decision is to switch from green to yellow, since the curve has a local minimum at slot 10.

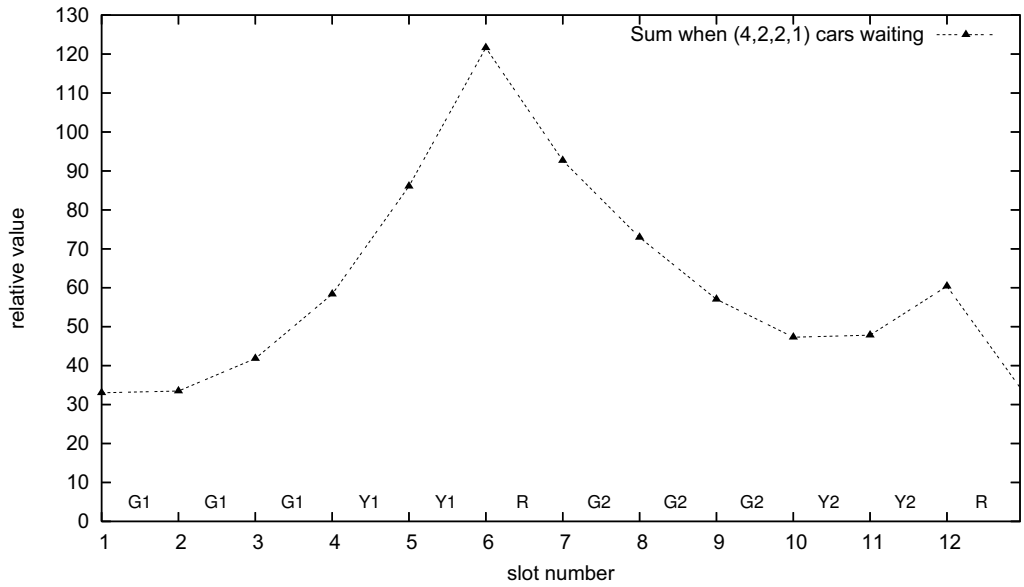


Figure 6.6: Sum of the relative value curves when $\mathbf{q} = (4, 2, 2, 1)$ cars are waiting at flow 1 to 4.

6.3.4 Step 4 – Simulation of RV1

To check the performance of RV1 one may in principle formulate a MC. For most intersections, the MC of RV1 cannot be analyzed, since the number of states $(t, \mathbf{q}) = (t;\, q_1, q_2, \ldots, q_F)$ is far too large. Since the actions and transitions do depend on the entire state description, the state space under RV1 cannot be decomposed; hence the flows cannot be analyzed in isolation of each other.

For large problems we can only evaluate RV1 by Simulation. Therefore a simulation model is developed and implemented in Delphi-Pascal. The process of cars arriving and leaving the queues and the control of the traffic lights is simulated in discrete time with time jumps equal to one slot (=2 seconds). At the start of every slot the number of cars

waiting at the queues ($\mathbf{q}$) is observed as well as the state of the traffic lights (t). Next, a decision τ is taken according to the policy under consideration, e.g. FC, RV1, or any exhaustive control policy. (Under FC $\tau = t$, under RV1 $\tau = \sigma(t, \mathbf{q})$.)

The decision τ is implemented instantaneously. Any cars arriving during a slot are assumed to do so at the beginning of that slot, any departures from a queue are modeled at the end of that slot. A car arriving at an empty queue and facing a green or yellow light passes the stopping line during the very same slot and hence contributes no waiting costs. At the start of the next slot the state of the traffic light is $\tau + 1$, and the above process repeats based on the new state of the queues.

Before the start of the simulation of RV1, the relative value curves for each flow and for all possible queue lengths are computed and stored. This takes hardly any time compared to solving the MDP, since the F Markov chains are of a much lower dimension: each MC has only 2 dimensions under FC compared to $F + 1$ dimensions for the MDP. When the simulation is started, every decision epoch the F relative value curves are summed based on the actual state $(t, \mathbf{q})$, as in Figure 6.6. Although based on FC, the improved policy RV1 is no longer a fixed cycle, since the green periods are interrupted (lengthened or shortened) every slot.

Remark – Since the relative values are computed flow-by-flow, the computation time remains low even when the upper bound Q on the queue lengths is very high: say 100 cars or even more. In the simulation model we assume that queues may contain an infinite number of cars. Although not likely when Q is set high enough, it may happen, at least theoretically, that the number of cars waiting at a queue exceeds Q. In that case, we need to estimate the relative value of that state since the relative value curves are known only for queues lengths $q \leq 100$. Then the relative value is estimated by quadratic extrapolation, see Appendix E.

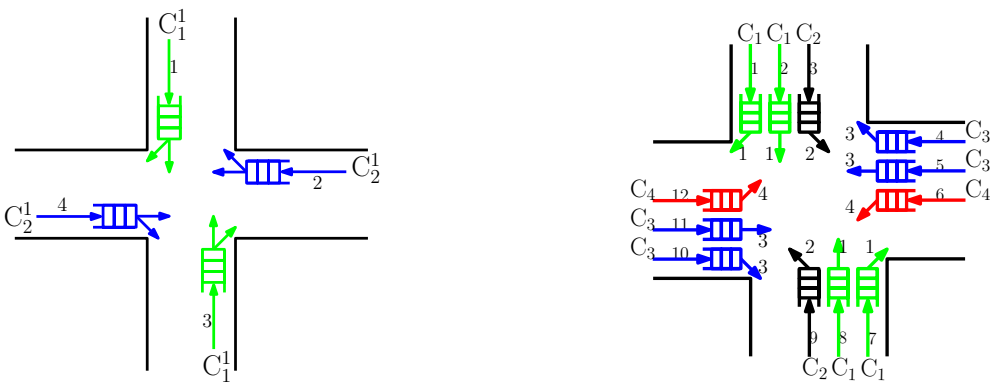
6.4 Test cases and results

To assess the quality of RV1, its performance is compared against FC, and some exhaustive control policies. For small problems the optimal MDP policy can be evaluated. Throughout this section we stick to cyclic control, hence $\mathcal{T}(t, \mathbf{q})$ in Equation (6.23) should be replaced by the action space for cyclic control $\mathcal{T}^{\mathrm{C}}(t, \mathbf{q})$. Acyclic control is studied in Sections 6.5 and 6.6.

6.4.1 Test cases – Infrastructures

Throughout this and the next chapters, the different policies are applied to two infrastructures: the F4C2 and F12C4 intersections that are displayed in Figure 6.7. These intersection were already introduced in Section 5.1.1. Whereas the F4C2 intersection may be realistic but simple, the F12C4 intersection has some interesting features. F12C4 has separate lanes for left turning traffic and non-identical numbers of flows per combination (C_1 and C_3 consists of twice as many flows as C_2 and C_4). Other infrastructures are studied in [54] and [55]. Instead of reporting results for many infrastructures, we believe its is at least as insightful to answer specific question for the F4C2 and F12C4 intersections under different workloads.

For each infrastructure, we consider different scenarios concerning the workload of the intersection and the symmetry in the arrival rates. In Table 6.1, we introduce the following terminology: *fully-(a)symmetric* and *partly-(a)symmetric*. In a, what-we-call, *'symmetric' case*, all combinations consist of an identical number of flows. F4C2 is thus a symmetric



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

(b) "F12C4" serve 12 flows in 4 asymmetric combinations

Figure 6.7: Two typical infrastructures.

Table 6.1: Terminology regarding the symmetry of test case with respect to the number of flows per combination and the arrival rates.

	# flows per combination	
	Identical for all C_s	Non-identical for all C_s
Arrival rates	'symmetric' intersection	'asymmetric' intersection
Identical $\forall f : \lambda_f = \lambda$	'fully-symmetric' case	'partly-asymmetric' case
Non-identical	'partly-symmetric' case	'fully-asymmetric' case

case, whereas F12C4 is asymmetric. When, in addition, the arrival rates in a symmetric case are identical for all flows, we deal with a *fully-symmetric case*; whereas when the arrival rates differ between flows, we speak of a *partly-symmetric case*. Asymmetric cases with identical arrival rates are called *partly-asymmetric cases*. When the arrival rates differ between flows, we call an asymmetric case a *fully-asymmetric case*.

For each infrastructure we formulate test cases in which the arrival rates are identical for all flows. In addition, we investigate a few cases with non-identical arrival rates, to find out whether 'thin' combinations are overruled by 'thick' combinations, given the definition of thickness in Section 6.1.2.

In the next section we report results for the two infrastructures under different work loads ρ (=0.4, 0.6, and 0.8), as defined in (6.2). Note that, as explained in Section 6.1.2, the effective workload is higher than ρ, since definition (6.2) does not include the all-lights-red time when switching to another combination. We do not consider cases where the effective workload is 1 or higher.

Simulation horizon

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 4,000 hours in the real system. At the start of each run a warming-up period of 450 slots (=15 minutes) applies. The reported mean waiting times are accurate up to (at least) 2 to 3 figures; to safe space we do not report confidence intervals.

6.4.2 Simple intersection F4C2

Identical arrival rates (F4C2)

In Table 6.2 the results are presented for the fully-symmetric F4C2 intersection at varying workloads. In the cases $\rho = 0.4, 0.6, 0.8$ all flows have identical arrival probabilities per slot of 0.2, 0.3, 0.4 respectively. The Optimize-fixed-cycle algorithm suggests cycle lengths of 8, 12 and 22 slots respectively, as reported in seconds in the next-to-last row. The departure times are the lengths of the effective green period under FC. RV1 is based on FC with these cycle lengths and departure times.

Table 6.2: Overall mean waiting time (in sec.) 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 sec.) and departure times per combination	16 (6, 6)		24 (10, 10)		44 (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 a few percent better. FC and XC yield on average 15% and 27% higher waiting times; when the load is high ($\rho = 0.8$) the differences are greatest. On average RV1 is just a little bit better than the anticipating exhaustive variants (XC-1 and XC-2), but the difference is small for this simple fully-symmetric case.

Non-identical arrival rates (F4C2)

Two F4C2 cases with non-identical arrival rates are considered:

Case I. The flows within the same combination have identical arrival rates, but C_2 is three times as thick as C_1: $\lambda_1 = \lambda_3 = \lambda$ and $\lambda_2 = \lambda_4 = 3\lambda$. (Hence $\rho_2 = 3\rho_1$.)

Case II. Flow 1 is a third as thick as the other flows: $\lambda_1 = \lambda/3$ and $\lambda_2 = \lambda_3 = \lambda_4 = \lambda$. (Hence $\rho_2 = \rho_1$.)

Table 6.3: Mean waiting times (in sec.) for two partly-symmetric F4C2 cases at $\rho = 0.6$.

Rule	EW	% above RV1	EW_1	EW_2	EW_3	EW_4
Case I						
$(\lambda_1, \ldots, \lambda_F) = (0.15, 0.45, 0.15, 0.45)$						
RV1	5.9		10.5	4.4	10.4	4.4
FC departure times 6, 14 sec.	6.9	+17%	11.2	5.4	11.2	5.4
XC	7.5	+27%	11.0	6.3	11.0	6.3
XC-1	6.6	+12%	8.6	5.9	8.6	5.9
XC-2	7.3	+24%	7.7	7.2	7.7	7.2
MDP (cyclic)	5.9	-0%	10.4	4.4	10.4	4.4
Case II						
$(\lambda_1, \ldots, \lambda_F) = (0.1, 0.3, 0.3, 0.3)$						
RV1	6.5		6.1	5.6	8.3	5.7
FC departure times 10, 10 sec.	8.0	+23%	5.2	8.3	8.3	8.3
XC	7.7	+18%	6.8	7.4	8.7	7.4
XC-1	6.5	+0%	5.4	6.2	7.3	6.2
XC-2	6.7	+3%	4.7	6.7	7.6	6.7
MDP (cyclic)	6.3	-3%	5.2	5.9	7.5	5.9

The results are reported in Table 6.3 for a workload of $\rho = 0.6$ (hence for Case I $\lambda = 0.45$ and for Case II holds $\lambda = 0.3$). The average waiting time, EW_f, is presented in the last four columns for each flow f. The overall average waiting time per car is reported in the second column. Again, next to the overall average waiting time per car, the relative differences compared to RV1 are reported. FC yields a waiting time that is for Case I and II respectively 17% and 23% above that under RV1. The overall average waiting time under all policies has reduced in both cases compared to the fully-symmetric case.

The impact of the arrival rates on the best configuration of FC, is read through the reported departure times: in Case I the cycle length remains $6 + 14 + 2 \cdot 2 = 24$ seconds. In Case II both the cycle length and the departure times are unaffected, since both combinations are equally thick.

By comparing the results for the two cases, we observe that flows that are part of a thicker combination experience a lower waiting time. This might seem counterintuitive, but a thicker combination gets a better treatment since it puts more weight to the scale. If a thin flow and a thick flow are together in one combination, then the thin flow benefits but the thick flow suffers a bit compared to when it would be part of a thicker combination.

6.4.3 Computation times on F4C2: MDP versus RV1

The computation complexity of the SA algorithm for computing an optimal MDP strategy is $\mathcal{O}(C^2(1+Q)^{2F})$. The complexity order is much lower for evaluating the Markov chains under FC, which is input to RV1. The computation of the relative values for all F flows is $\mathcal{O}(F \cdot D \cdot (1+Q)^2)$.

To illustrate the difference in complexity order, we report some running times on an in-2007-modern PC, an Intel Pentium 2.8GHz processor and 2Gb RAM- memory. The RAM-memory is sufficient to registers about 10^8 states. Depending on the implementation and the available time one can execute a successive approximation scheme for cases with millions of states. We discuss the running time for a simple example using the program, written in Delphi-Pascal, developed to generate results that are included in this thesis.

Consider the F4C2 intersection. When the load of the intersection is high, say 80%, the number of waiting cars virtually never exceeds 18, as may be checked by simulation. Hence we truncate queues at $Q = 18$ cars. The total number of states is about a million (or to be precise $(1+18)^4 \cdot (2 \cdot (1+3)) = 1,042,568$). The computation of the optimal cyclic MDP policy takes about 12 hours for 520 iterations, or 86 seconds per iteration of the SA algorithm. The computation time can be reduced significantly, by setting the upper bound Q somewhat lower. This is only possible because we accurately compute the externality costs by quadratic extrapolation. Further, one may note that a nearly optimal MDP policy is found far before iteration 520, but the SA algorithm continues iterating until the gain of the related Markov chain is closely approximated.

Computing for all flows the relative values of all states under FC takes only a few seconds, depending on the required accuracy. The computation time needed to obtain a nearly optimal MDP policy is thus much higher for a problem of the same size.

6.4.4 Complex intersection F12C4

For the F12C4 intersection the optimal MDP strategy cannot be computed because the number of states is prohibitively large. Even when in the MDP model queues are truncated at 2 cars, the total number of states is still very large: $8.5\,10^6$ 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 4 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 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 (and XA-2), green lights are turned into red as soon as at each of the flows that have right of way 2 or less cars are queued.

We consider both a partly-asymmetric and a fully-asymmetric F12C4 case. For the partly-asymmetric case (where all arrival rates are identical), we discuss consecutively:

- the impact of the asymmetry and the workload on the overall average waiting time, and
- how the asymmetry of the combination affects the waiting time distribution and the queue length distribution.

For the fully-asymmetric F12C4 case (with non-identical arrival rates), we test whether a very thin combination is getting under-served under a high workload. Therefore we assume that C_2 is 6 times as thin as C_1 and C_3, and three times as thin as C_4.

In Section 6.4.5, we discuss the impact of the sub-optimal cycle lengths D on the performance of FC and RV1.

Identical arrival rates (partly-asymmetric F12C4)

In Table 6.4 the results for varying workloads at a F12C4 intersection are presented for the case where all flows have identical arrival intensities (0.1, 0.15, and 0.2 for $\rho = 0.4$, 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 $\rho = 0.8$ XC-2 yields an average waiting time that is 28% higher than under RV1. Then FC performs even better than XC-2.

Table 6.4: Overall mean waiting time (in sec.) 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 sec.)	32		40		88	
FC departure times (in sec.)	(6, 6, 6, 6)		(8, 8, 8, 8)		(20, 20, 20, 20)	

In Table 6.5, we study the waiting time at the different flows when the workload is 0.8. Although the arrival rates λ_f are identical for all flows, the mean waiting times differ 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 times per car under FC are identical since all flows experience a departure time (or an effective green time) of 20 seconds (10 slots) per cycle.

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

Table 6.5: Mean waiting times (in sec.) 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 sec.	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

Waiting time and queue length distribution (partly-asymmetric F12C4)

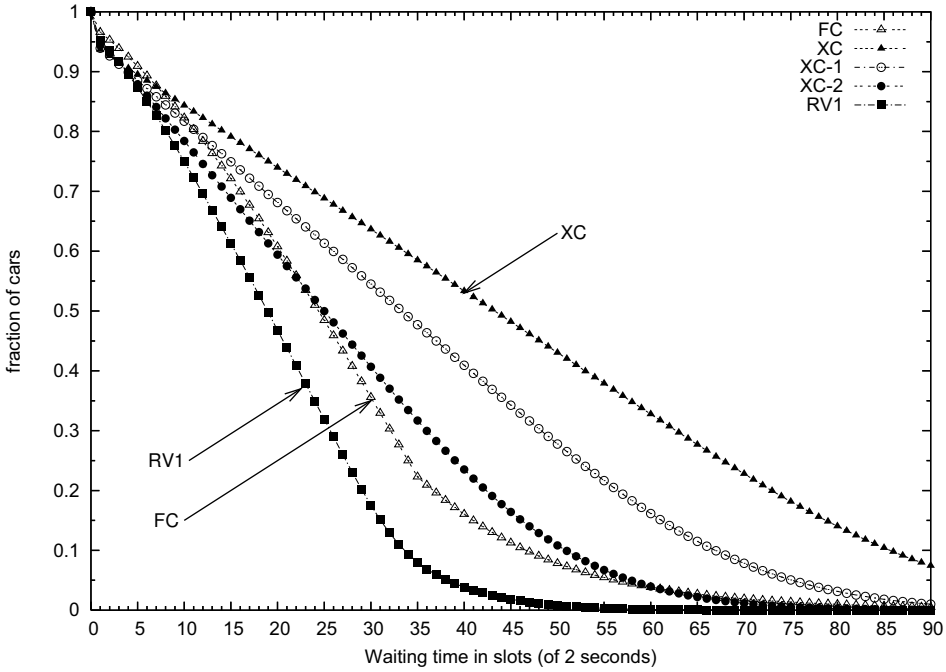
Waiting time distribution

Although our primary criterion is the average waiting time, we are also interested in the distribution of the waiting time. From the waiting time distribution one reads the fraction of cars that experience a waiting time above some level. Therefore we have computed the inverse cumulative distribution or the tail distribution of the waiting time based on the individual waiting times that are registered by the simulation program. Related to the distribution of the waiting time is the distribution of the queue length at the start of a slot: the more often a high queue length is observed the more cars experience a high waiting time.

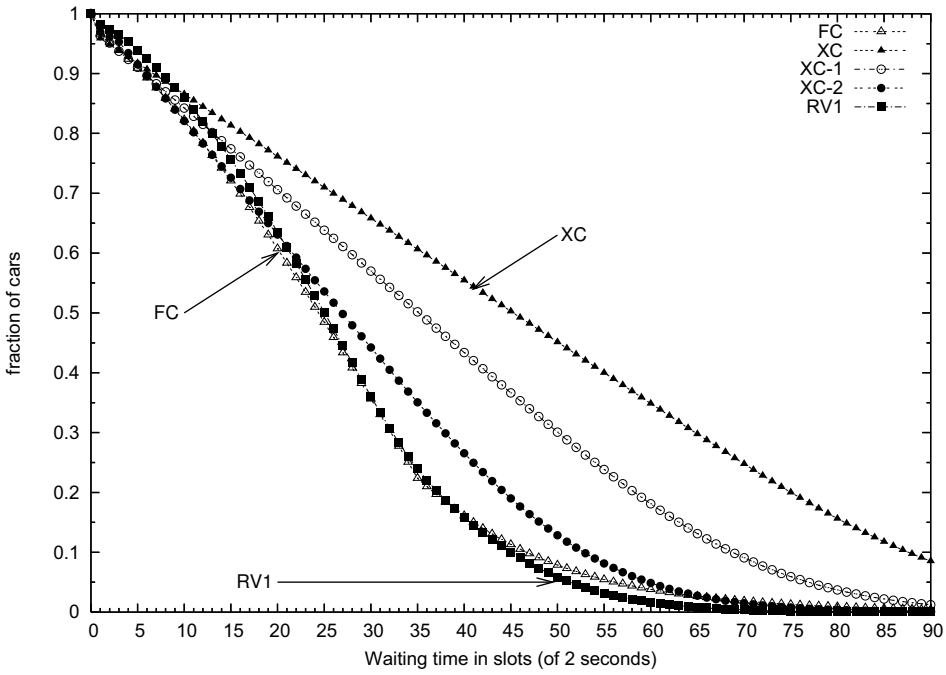
In Figures 6.8 and 6.9 we show the tail distributions of both the individual waiting time and the queue length at the start of a slot under the various control strategies: RV1, FC, and XC, XC-1, XC-2. Since the arrival rates are identical at all queues the only difference between flows is whether they are part of a thick or a thin combination. The distributions are computed after having simulated the policies for 4,000 hours (100 runs of 72,000 slots each), during which on average 1.44 million cars arrive at each flow. On the horizontal axis of Figure 6.8(a) and 6.8(b) one reads the individual waiting time which is an integer number of slots ranging from 0 to 90 slots (3 minutes). Vertically one reads the fraction of cars experiencing a waiting time greater than or equal to the value read at the horizontal axis.

Figure 6.8(a) depicts the waiting time distribution for the 8 flows part of the thick combinations. The probability of excessive waiting time, is lowest under RV. For example only 17% of the cars arriving at combination 1 or 3 experience a waiting time exceeding 1 minute (30 slots), while under FC still about 36% of the cars has to wait for at least 1 minute. Under pure-exhaustive control about 65% has to wait for 1 minute and often much more. Although the differences are smaller we conclude from Figure 6.8(b) that flows part of thin combinations do favor RV1 since it avoids excessive waiting times better than all other strategies.

Hence RV1 is superior not only in its mean waiting time but also in its distribution. A great reduction in the average waiting time at most of the queues is observed, while excessive waiting times at the other queues occur less frequently than under FC.



(a) Tail distribution of waiting time of cars in thick combinations: $f \in \mathcal{C}(1) \cup \mathcal{C}(3)$.



(b) Tail distribution of waiting time of cars in thin combinations: $f \in \mathcal{C}(2) \cup \mathcal{C}(4)$.

Figure 6.8: Inverse cumulative distributions (tail distributions) of the waiting time for flows f of thick and thin combinations of partly-asymmetric F12C4 ($\rho = 0.8$ and $\lambda_f = 0.2$).

Queue length distribution

In Figure 6.9 we show for the several policies the tail distributions of the queue lengths at the start of a slot. Based on the queue length distribution Figures 6.9(a) and 6.9(b), we draw similar conclusions: the queues of thick combinations are shorter under RV1 than under any of the other policies. At the other queues there is only a small difference in queue lengths between FC and RV; again the tail under RV1 is less heavy than under FC.

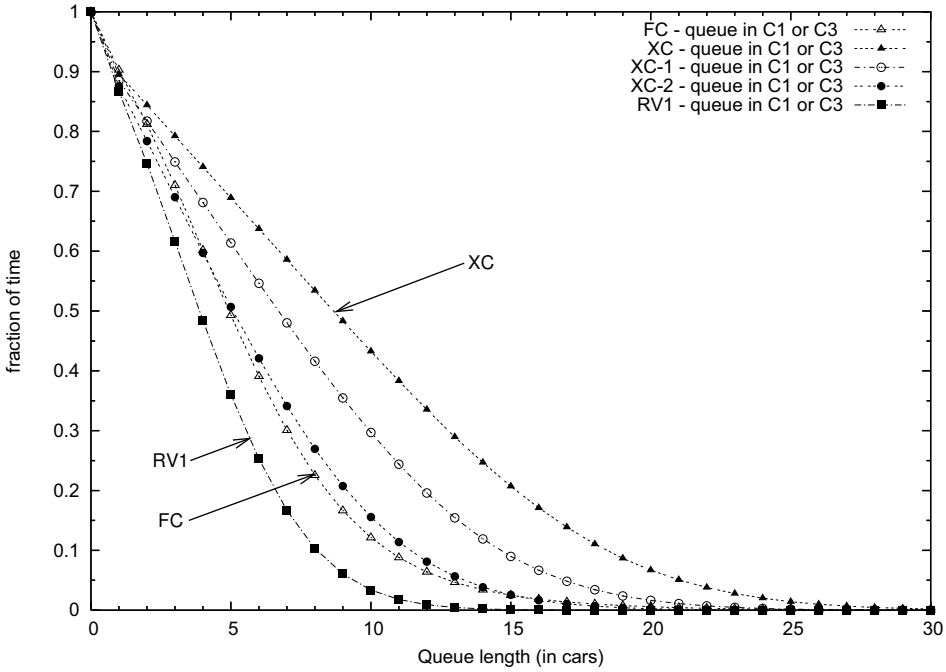
When comparing the waiting time curves of the FC policy to the queue length distribution under FC, we observe that the waiting time curve shows a bend at 35 slots, while the queue length distribution does not show a similar irregularity. This bend is explained by noting in this case an all-red period lasts 34 slots. Cars that that could not be served during the first green period that they encounter, need to wait at least another 34 slots higher for the start a next green period. The curves under the dynamic control policies do not show such an bend, since the all-red period has no fixed length under dynamic control. The queue length distribution under FC does also not show such an irregularity since cars arrive uniformly over time.

Non-identical arrival rates (fully-asymmetric F12C4)

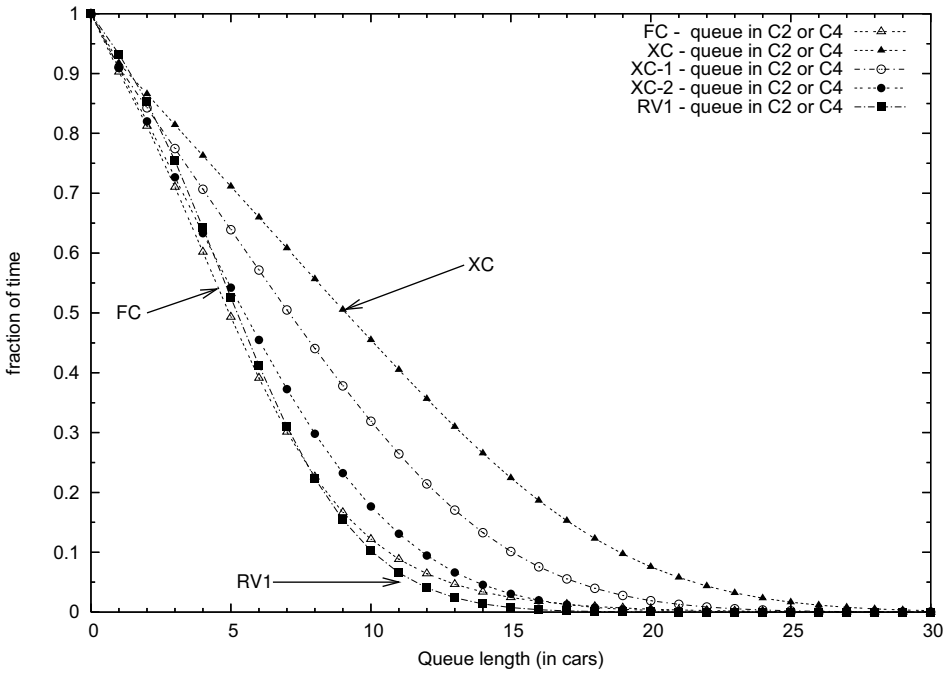
To study the impact of non-identical arrival rates in combination with an asymmetric number of flows per combination, we study the fully-asymmetric F12C4 intersection with workload 0.8. We assume C_2, which consists only 2 flows (flow 3 and 9) to have an arrival rate that is only a third of all other 10 flows. The arrival rates are thus 0.08 for flows 3 and 9 and 0.24 for all other flows. Then the overall load is again 0.8. Thus, C_2 is $(4 \cdot 0.24)/(2 \cdot 0.08) = 6$ times as thin as C_1 and C_3, and C_2 is three times as thin as C_4. Since C_2 adds little weight to the scale, one may expect C_2 to suffer high waiting times. In Table 6.6 the average waiting time per car is reported for all flows.

Table 6.6: for F12C4 at $\rho = 0.8$: C_2 is six times as thin as C_1 and C_3.

Rule	*EW* overall		*EW* C_1,C_3	*EW* C_2	*EW* C_4
RV1	39.4		34.9	66.6	48.7
FC dep. times 22, 8, 22, 22 sec.	47.1	+20%	45.6	69.4	45.6
XC	85.1	+116%	82.8	107.8	86.9
XC-1	66.6	+69%	64.6	84.8	68.3
XC-2	50.5	+28%	48.8	65.0	52.6



(a) Tail distribution of queue length of queues in thick combinations: $f \in \mathcal{C}(1) \cup \mathcal{C}(3)$.



(b) Tail distribution of queue length of queues in thin combinations: $f \in \mathcal{C}(2) \cup \mathcal{C}(4)$.

Figure 6.9: Inverse cumulative distributions (tail distributions) of the queue lengths for flows f of thick and thin combinations of partly-asymmetric F12C4 ($\rho = 0.8$ and $\lambda_f = 0.2$).

The waiting time of cars in C_2 is (considerably) lower than under FC, XC and XC-1, although the waiting time at C_2 is indeed higher than at all other flows. Apparently, C_2 is not under-served. XC-2 performs marginally better for C_2, but the average over all queues is 28% higher than under RV1. Pure-exhaustive control performs very bad, since it does not anticipate departures during the yellow slots. FC performs better than XC-2 but yields a waiting time that is still 20% above that of RV1.

We conclude that, also when arrival rates are non-identical, RV1 is superior, and that RV1 does not result in excessive waiting times at queues of thin combinations.

6.4.5 Sensitivity regarding the cycle length D

In Section 6.3.1, we have observed already that the curve of the average waiting time as a function of the cycle length D, is quite flat around the optimal value of D. For the partly-asymmetric F12C4 case we have seen that the local minima are where D is a multiple of $C = 4$ slots. We now return to that example and show that RV1 is even less sensitive to the chosen cycle length of the underlying FC.

In Figure 6.10 we have plotted the average waiting times for D ranging from 24 to 56 slots with increments of 4 slots. Although the overall long-run average waiting time under FC is high when D is set low, say $D = 24$, the average waiting is much lower under RV1. It appears that RV1 is almost insensitive to the cycle length of the underlying FC for which the relative values are computed.

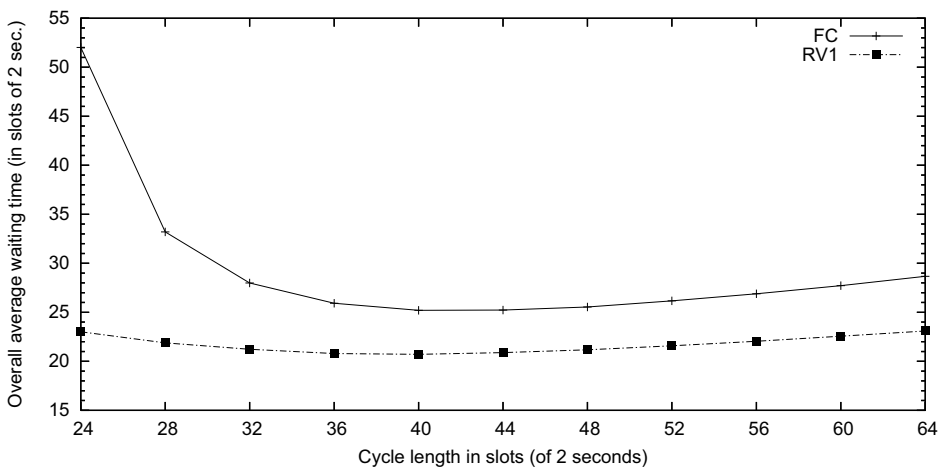


Figure 6.10: Sensitivity of FC and RV1 policy to cycle length for F12C4 at workload 80%

6.4.6 Conclusions

The optimal dynamic control of traffic lights, given the number of cars waiting at each queue, involves a high-dimensional state space. Therefore an optimal MDP policy can only be found for small intersections such as the F4C2 intersection. For more complex intersections such as the F12C4, one relies on heuristic and approximate solutions, such as RV1.

The newly developed policy RV1 is based on a one-step policy improvement algorithm over FC. Since RV1 improves FC, first one needs to determine a good configuration of FC, i.e. nearly optimal departure times and the related nearly-optimal cycle length D. Even when the selected FC is sub-optimal, RV1 shows very good improvements; it is quite robust for a broad range of cycle lengths D.

RV1 improves FC and exhaustive control policies at varying loads. At a high workload of the intersection, the control scheme has to be sophisticated. RV1 shows the greatest improvement over the other policies when the workload is high, say 0.8. Also when the intersection is asymmetric in the number of flows in each combination, like in F12C4, the control rule has to be more sophisticated than FC or exhaustive control. The F4C2 and F12C4 cases show that RV1 reduces the waiting time under FC by about 20%. For the F4C2 infrastructure, we have shown that RV1 performs nearly optimal. Especially for the F12C4 infrastructure, (anticipative) exhaustive control policies may perform even worse than FC. Clearly the definition of when to switch green lights into yellow should be refined when multiple queues can be served simultaneously. RV1 is refined at this point, and easy to implement.

6.5 More-advanced improvement rules

Although the results for RV1 are promising, we are interested in more advanced policies that make use of the relative values of the states under FC. First, we develop a variant called RV2 in which two decisions are evaluated simultaneously to find a best time jump. Next, three acyclic control policies RV1S, RV1M and RV2M are constructed.

6.5.1 Adjust two green periods (RV2)

The new policy RV1, developed and tested in the previous sections, performs very well, while it is based on only a single improvement step over FC. RV1 adjusts the length of the actual green period by making a time jump in the cycle: to evaluate a time jump one assumes that FC is resumed from that point onwards. In order to select a time jump RV1 ignores future time jumps that will be taken in the real system. Although for the F4C2 infrastructure, we have checked that RV1 performs nearly optimal, this cannot be checked for the F12C4 infrastructure. For the F12C4 infrastructure, decisions based on the queue lengths are more refined, therefore one may expect to get better decisions when a time jump is evaluated by also taking a next time jump into account.

A one-step policy improvement over RV1 cannot be executed, because the relative value vectors cannot be derived for RV1 since the state space under RV1 cannot be decomposed. Therefore we propose to include a second time jump, which plans where to resume FC after the next switching phase. In the evaluation of a first time jump, the second time jump allows thus to shorten the length of the next green period. We refer to this new policy as RV2. In Figure 6.11 the two time jumps are depicted. Suppose at time t the lights are green to some combination. Then an immediate time jump to τ_1 is evaluated by assuming that the next green period will be started at slot τ_2 instead of at slot r. (e and r denote respectively the end of the green period and the end of the next all-red slot, which is the start of the next green period under FC.)

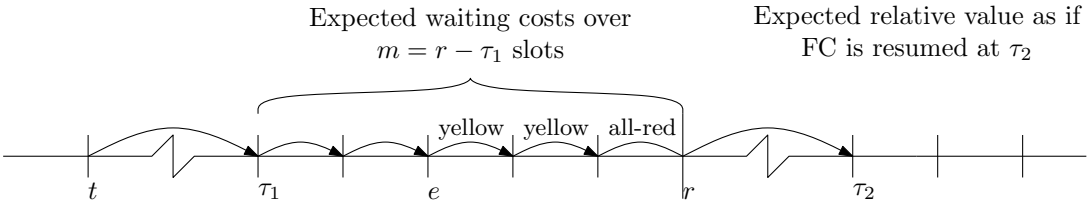


Figure 6.11: Setting RV2 actions requires evaluating two jumps τ_1 and τ_2

RV2 adjusts the lights according to a time jump τ_1, but for the evaluation of the time jump, RV2 optimizes over the length of the next green period by selecting a second time jump τ_2. Given the actual state $(t, \mathbf{q})$ at the start of a slot, RV2 derives the best pair (τ_1, τ_2) that minimizes the long-run average waiting cost assuming that FC is resumed at τ_2. When evaluating a time jump RV2 assumes FC to continue from slot τ_1 to the end of the next all-red slot, after which a second time jump τ_2 is selected, as illustrated in Figure 6.11. The end of the all-red slot coincides with the start of slot r at which a next green period would start (if not interrupted by τ_2). RV2 thus implements only the first time jump from t in to τ_1. The second time jump in RV2 is introduced solely to obtain a more accurate evaluation of jumping from t to τ_1.

For example, $\tau_1 = t-1$ implies to lengthen the current green period by one slot for the flows that have right of way at slot t. Under RV1 all cars queued at the other combinations need to wait one more slot. Under RV2 however, the next green period may be shortened, just as what may happen in the real system, and consequently the red period of combinations visited thereafter may be shortened. A time jump that may look not beneficial under RV1 could thus be beneficial in the real system. RV2 better resembles the real system since it may anticipate the action taken at the end of the next all-red slot.

Computations of RV2 actions

The process depicted in Figure 6.11 can be modeled as a MC. Except for the jump from r to τ_2, the process resembles an MC of FC. The MC is characterized by the state $(t, \mathbf{q})$, and the time jump τ_2. The process at queue f is independent of the arrival and departure processes at the other queues. The relative values for this MC may be computed flow-by-flow. Let $u_m^f(\tau_1, q_f, \tau_2)$ denote the relative value for flow f of starting, m slots before the start of the next green period, the MC in slot τ_1 with q_f cars queued and breaking FC at the start of a next green period by jumping to slot τ_2.

For the computation of the relative values for the interrupted FC, one may use the approximation of the relative values, $\mathbf{v}_{rel}^f$, of FC. In words, $v_{rel}^f(\tau_2, q_{f,m})$ is used as terminal costs when ending in state $(\tau_2, q_{f,m})$; $q_{f,m}$ is a stochastic variable indicating the number of cars waiting at queue f after m slots. We add to $v_{rel}^f(\tau_2, q_{f,m})$, the expected waiting costs over the first m slots. Since we have truncated the (in principle) infinite horizon in the approximation of the relative values by $\mathbf{v}_{rel}^f$, we need to compensate for unequal horizons by subtracting m times the average costs per slot under FC, $g^{(f)}$.

RV2 algorithm – The following algorithm is used to derive RV2 actions: Steps 1 – 3 can be computed off-line. Step 4 is executed run time: every slot a best RV2 action is read from Equation (6.27). The algorithm implicitly computes the relative values of states for Markov chains similar to the one presented in Figure 6.11.

Step 1. Approximate for all flows f, the relative values of the states under FC, $\mathbf{v}^f_{rel}$, and the gain, $g^{(f)}$, of the MC for flow f.

Step 2. Set for all flows f, for all $q_f \in \{0, 1, \ldots, Q\}$, and for all $\tau_1, \tau_2 \in \{1, \ldots, D\}$:

$$u^f_0(\tau_1, q_f, \tau_2) = v^f_{rel}(\tau_2, q_f). \tag{6.24}$$

Step 3. Compute recursively for for all flows f, for $m = 1$ to $d_f + 1$:
for $\tau_1 := 1$ to D, for all $q_f \in \{0, 1, \ldots, Q\}$:

- if τ_1 implies red to flow f:

$$\begin{aligned} u^f_m(\tau_1, q_f, \tau_2) \;=\; q_f + & \quad \lambda_f \quad u^f_{m-1}(\tau_1 + 1, q_f + 1, \tau_2) + \\ + & \quad (1 - \lambda_f) \quad u^f_{m-1}(\tau_1 + 1, q_f, \tau_2) \\ - & \quad g^{(f)} \end{aligned} \tag{6.25}$$

  (when $q_f = Q$, then $u^f_{m-1}(\tau_1 + 1, q_f + 1, \tau_2)$ is approximated by quadratic extrapolation),

- if τ_1 grants green or yellow to flow f:

$$\begin{aligned} u^f_m(\tau_1, q_f, \tau_2) \;=\; q_f + & \quad \lambda_f \quad u^f_{m-1}(\tau_1 + 1, q_f, \tau_2) \\ + & \quad (1 - \lambda_f) \quad u^f_{m-1}(\tau_1 + 1, (q_f - 1)^+, \tau_2) \\ - & \quad g^{(f)}. \end{aligned} \tag{6.26}$$

Step 4. An optimal RV2 action τ_1 in state $(t, \mathbf{q})$ is online computed by:

$$\arg \min_{\substack{\tau_1 \in \mathcal{T}(t, \mathbf{q}) \\ \tau_2 \in \mathcal{T}(r, \mathbf{q})}} \sum_{f=1}^{F} u^f_{r-\tau_1}(\tau_1, q_f, \tau_2), \tag{6.27}$$

(with r = slot number at which a next green period would start when FC would be continued).

Before presenting results for RV2 (in Section 6.6), we first introduce a few more-advanced acyclic RV policies.

6.5.2 More-advanced acyclic RV policies (RV1M and RV2M)

In Section 6.4 we have studied results for cyclic control policies. By using action space $\mathcal{T}(t, \mathbf{q})$ instead of $\mathcal{T}^{\mathrm{C}}(t, \mathbf{q})$, acyclic control policies are considered. In the evaluation of a feasible time jump τ, RV1 assumes FC to resume at τ. Hence RV1 does not allow to evaluate a decision by assuming a different cyclic order in which the combinations receive green.

Therefore we have developed another RV policy: RV1M. Under RV1M a time jump to τ is evaluated more carefully by allowing to modify the cyclic order in which the combinations receive green. RV1M selects the best τ based on the best cyclic order at which FC may resume. Since decisions are updated every slot, only decision τ is implemented every slot.

Below we deliberate on the acyclic counterpart of RV1, and on RV1M.

Acyclic RV1

For example, consider the F12C4 case and the green period of C_3 has just ended. Now, the controller needs to select a combination to serve next. When τ grants green to C_1, FC is resumed in the order C_1-C_2-C_3-C_4 - C_1-C_2-etc. Hence under RV1 C_4 is skipped and thus served after all other combinations are visited. Since this might be too bad to C_4, a jump to τ might be rejected under RV1, although it could be optimal to serve C_1 before C_4.

RV1M

Under RV1M we may modify the cyclic order to better evaluate the jump to τ. (The 'M' in RV1M stands for Modify the cyclic order.)When τ grants green to C_1, RV1M not only considers the cyclic order C_1-C_2-C_3-C_4 - C_1-C_2-etc. Instead RV1M evaluates τ as if FC continues at the best of the $((C-1)! = 3! =)$ six possible cyclic orders:

Cyclic order 1:	C_1-C_2-C_3-C_4 - C_1-C_2-...
Cyclic order 2:	C_1-C_2-C_4-C_3 - C_1-C_2-...
Cyclic order 3:	C_1-C_3-C_2-C_4 - C_1-C_3-...
Cyclic order 4:	C_1-C_3-C_4-C_2 - C_1-C_3-...
Cyclic order 5:	C_1-C_4-C_2-C_3 - C_1-C_4-...
Cyclic order 6:	C_1-C_4-C_3-C_2 - C_1-C_4-...

For the selection of the best τ from the relative values of the states under FC, we need to consider in this case six potential new intended cycles ϕ. The set of all possible cycles, is denoted by $\Phi(\tau)$ and depends on the slot number τ in the base cycle C_1-C_2-C_3-C_4. In principle RV1M is still a one-step policy improvement based on the relative values of states under FC, but next to τ one selects a best cyclic order, ϕ, of combinations by which FC resumes. Under RV1M the best decision is thus the best pair (τ, ϕ), with $\phi \in \Phi(\tau)$ being the new intended cyclic order.

In order to value a decision (τ, ϕ), one needs the relative values of the states under FC. The relative values of states under FC with cyclic order ϕ can be derived from the relative values (6.22) of states under FC with cyclic base order C_1-C_2-C_3-C_4. Instead of saving $(C-1)!$ relative value tables, we suggest to save on memory usage, using the fact that the relative values of states for different ϕ are only shifted in the slot number t.

The relative values $\mathbf{v}^f_{rel}$, in Equation (6.22), relate to the original cycle (C_1-C_2-C_3-C_4). Suppose that under this cyclic order flow f receives green from slot $t = r_f$ to slot e_f. When the cyclic order changes, f receive green at other slot numbers: the start of a green period may not no longer be at slot r_f, but say at slot $r_f + \delta_f(\phi)$. The relative value curve of flow f is shifted by a constant δ_f which depend on the new intended cycle ϕ.

Example – Consider a case where $C = F = 4$ (one flow per combination) and τ grants green to C_1. The cyclic base order is (C_1-C_2-C_3-C_4), and the green period of flow f starts at slot r_f. When one evaluates the cyclic order $\phi =$ (C_1-C_3-C_2-C_4), then the green period of C_2 is postponed by $\delta_2 = r_4 - r_3$ slots: which is the effective green time of C_3 plus one all-red slot. C_3 gets green $r_3 - r_2$ slots earlier than under the cyclic base order. The delay δ_3 is thus negative for C_3, since the time until it receives green is shortened: $\delta_3 = r_2 - r_3$. The order of C_1 and C_4 is preserved, hence $\delta_1 = \delta_4 = 0$.

For this particular order ϕ the delays are thus $\delta(\phi) = (0, r_4 - r_3, r_2 - r_3, 0)$, which are the shifts for looking up the right relative values.

Actions under RV1M – The optimal actions under RV1M follow from Equation (6.28), where $\tau + \delta_f(\phi)$ should be read as $[(D + \tau + \delta_f(\phi)) \mod D]$ whenever $\tau + \delta_f(\phi)$ falls not in $\{1, 2, \ldots, D\}$.

$$\sigma^{\mathrm{RV1M}}(t, \mathbf{q}) = \arg \min_{\tau \in \mathcal{T}(t,\mathbf{q})} \left(\min_{\phi \in \Phi(\tau)} \sum_{f=1}^{F} v^f_{rel}(\tau + \delta_f(\phi), q_f) \right). \tag{6.28}$$

RV2M

Similarly one may investigate multiple cyclic orders ϕ under RV2: we call the resulting policy RV2M. Under RV2M one thus selects the best triple (τ_1, τ_2, ϕ). We skip a formal definition of RV2M since it involves complicated notations.

Complexity of RV1M and RV2M

When a feasible time jump is to be evaluated over all possible cyclic orders, the computational complexity grows exponentially in the number of combinations. The combination to serve first is set by the time jump (τ or τ_1). The remaining $C-1$ combinations can be visited in $(C-1)!$ different orders. (If the combination that has been served most recently has to wait until all other combinations are served, then the number of cyclic orders to consider is $(C-2)!$.)

For the F12C4 intersections one deals with $C = 4$ combinations. The number of possible cycles is small enough for considering all permutations. As a result simulating RV2 takes considerable more time than simulating RV1. For cases where the number of combinations is (much) larger, one could save time by considering not all permutations of the $C-1$ combinations, but only those in which two combinations are pulled forward in the intended cycle. Then instead of $(C-1)!$ permutations only $(C-1) \cdot (C-2)$ possible cycles are considered. The simulation program in which we test RV1M and RV2M allows for considering a subset of all permutations.

6.6 Evaluation of advanced RV policies

With the introduction of the new policies we are interested in comparing cyclic against acyclic control. Furthermore, we question whether the more advanced rules RV2, RV1M and RV2M yields much lower average waiting time than RV1. In this section we investigate the quality of these policies and compare them against FC and a number of (anticipative) exhaustive control policies. For the F4C2 case acyclic policies are not of interest, since F4C2 consists of only two combinations. Therefore the focus is on the infrastructure F12C4, under different loads. Again all results are obtained by simulating each policy for 4,000 hours (100 runs of 72,000 slots), such that the results reported are accurate up to 2 or 3 digits.

6.6.1 F12C4 with identical arrival rates

In Table 6.7 one reads for varying workloads at a partly-asymmetric F12C4 intersection, the long-run average waiting time per car overall flows. The top part of the table, containing the results for the cyclic policies, greatly coincides with Table 6.4, except that the results for RV2 are added. RV2 appears to perform equally well as RV1: although RV2 is based on a more accurate evaluation, it hardly improves RV1.

When the control of the lights may be acyclic, RV1 shows an improvement of 7.7% over cyclic RV1, when the workload is low (0.4). However when the workload is low, anticipative-exhaustive control XA-2 performs even better. (Remember: under XA, XA-1, and XA-2 one switches to the combination with the longest queue at the end an all-red slot.) The lowest waiting time at $\rho = 0.4$ is obtained under RV1M and RV2M: 12.4% below RV1. Apparently, modifying the cyclic order is important when the workload is low.

When the workload is high ($\rho = 0.8$), the more advanced acyclic rules perform hardy any better than RV1. When the load is high it becomes critical not to waste much time on switching or on serving queues at which not much traffic is waiting. The best anticipative exhaustive control policy yields a waiting time that is about 28% above that of RV1.

Conclusions – One may conclude that when the workload is low one needs to select carefully which combination to serve next, while when the workload is high one should take care in formulating when to switch. For the best results RV1M (or RV2M) is favored, but when the control has to be cyclic RV1 is favored because of it requires less and more simple computations than RV2.

Table 6.7: Overall mean waiting time (in sec.) at different loads for F12C4 ($\lambda_f = \rho/4$).

Rule	$\rho = 0.4$		$\rho = 0.6$		$\rho = 0.8$	
Cyclic policies:						
RV1	13.5		19.3		41.8	
RV2	13.5	0.0%	19.4	+0.1%	41.6	−0.5%
FC	15.0	+10.9%	23.7	+23%	50.5	+20.8%
XC	19.2	+41.9%	33.4	+73%	89.8	+114.8%
XC-1	14.9	+10.1%	25.1	+30%	70.1	+67.2%
XC-2	13.5	+0.0%	19.6	+2%	53.3	+27.6%
FC cycle length (in sec.)	32		40		88	
FC departure times (in sec.)	(6, 6, 6, 6)		(8, 8, 8, 8)		(20, 20, 20, 20)	
Acyclic policies:						
RV1	12.5	−7.7%	18.9	−2.1%	41.7	−0.2%
RV1M	11.9	−12.4%	18.3	−5.3%	41.7	−0.2%
RV2M	11.9	−12.4%	18.3	−5.4%	40.9	−2.1%
XA	18.7	+37.9%	33.4	+72.6%	90.0	+115.4%
XA-1	14.0	+3.1%	25.1	+29.7%	70.2	+68.1%
XA-2	12.3	−9.1%	19.1	−1.0%	53.8	+28.8%

Waiting time per flow

Although the arrival rates per flow are identical (0.1, 0.15, 0.2 when the workload is 0.4, 0.6 and 0.8 respectively), the mean waiting times may differ per combination. C_1 and C_3 are 'thicker' than 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. Under a policy that takes the number of queued cars into account, such as the RV policies, we may expect that car drivers in C_1 and C_3 experience a lower waiting time than car drivers in C_2 and C_4.

In Table 6.8 detailed results are reported for $\rho = 0.8$. RV1 and RV2 yield a much lower overall average waiting time than FC, but not at the expense of a high average waiting time at the queues of the thin combinations C_2 and C_4. Under cyclic control, the average waiting times at C_2 and C_4 are just as high as under FC. The waiting time at the other queues is reduced by more than 25% from 50.5 seconds to about 37 seconds.

Under acyclic control, thin combinations may be skipped or their service may be ended too early, in favor of serving a thick combination. Nevertheless, RV1, RV1M and RV2M yield average waiting times that do not differ much from those under FC. The waiting time at C_2 and C_4 stays very close to that under FC, while the waiting time at the other

Table 6.8: Mean waiting times (in sec.) for F12C4 at $\rho = 0.8$ ($\forall f : \lambda_f = 0.2$).

Rule	*EW* overall		*EW* C_1, C_3	*EW* C_2, C_4
Cyclic policies:				
RV1	41.8		37.4	50.6
RV2	41.6	−0.5%	37.1	50.3
FC dep. times 20, 20, 20, 20 sec.	50.5	+20.8%	50.5	50.4
XC	89.8	+114.8%	88.5	92.4
XC-1	70.1	+67.2%	68.9	72.4
XC-2	53.3	+27.6%	52.1	55.8
Acyclic policies:				
RV1	41.7	−0.2%	37.3	50.6
RV1M	41.7	−0.2%	36.2	52.9
RV2M	40.9	−2.1%	35.2	52.5
XA	90.0	+115.4%	88.0	94.0
XA-1	70.2	+68.1%	68.3	74.1
XA-2	53.8	+28.8%	51.8	57.7

queue is as low as 35.2 seconds: a reduction of 31%. Compared to acyclic exhaustive control the waiting times at all queues is much lower under the RV policies. Even the best anticipative acyclic exhaustive control policies yield waiting times that are higher than under FC.

The difference between cyclic and acyclic control, in terms of the average waiting times, is very small.

6.6.2 Non-identical arrival rates (F12C4)

Hypothesis

Flows that contribute little to the total workload may suffer high waiting times under acyclic control, since they may be skipped or their green period is ended too early in favor of serving thick combinations. Under the RV policies, we hope and expect that this will not happen too much, since letting many cars waiting at a thin combination is penalized by the definition of the relative values.

Ending a green period too early implies that all cars left behind at that queue need to wait for the next green period. A thin combination has a shorter green period in FC than a thick combination. Letting say 5 cars waiting at a thin combination is consequently less

Table 6.9: Mean waiting times (in sec.) for fully-asymmetric F12C4 at $\rho = 0.8$ – C_2 is six times as thin as C_1 and C_3.

Rule	*EW* overall		*EW* C_1,C_3	*EW* C_2	*EW* C_4
Cyclic policies:					
RV1	39.4		34.9	66.6	48.7
RV2	39.3	−0.2%	34.4	66.3	48.9
FC dep. times 22, 8, 22, 22 sec.	47.1	+19.5%	45.6	69.4	45.6
XC	85.1	+115.9%	82.8	107.8	86.9
XC-1	66.6	+68.9%	64.6	84.8	68.3
XC-2	50.5	+28.1%	48.8	65.0	52.6
Acyclic policies:					
RV1	38.7	−1.7%	33.6	75.8	46.5
RV1M	38.3	−2.9%	32.5	82.5	46.9
RV2M	37.6	−4.5%	31.8	79.8	46.4
XA	82.7	+109.7%	73.8	201.9	78.7
XA-1	64.8	+64.4%	57.7	158.1	62.3
XA-2	49.7	+26.2%	44.0	121.1	48.8

desirable according to the relative values than letting 5 cars waiting at a thick combination. Since under FC some cars may have to wait for even two or more cycles before getting served, when some queue become long, the RV policies aim at keeping all queues short.

We expect thus very thin combinations not to be skipped under the RV policies, although they may experience higher waiting times than cars at the other queues. We test this hypothesis for the F12C4 case with the arrival rates of the queues in C_2 being only a third of the arrival rate at the other ten queues. The workload of C_1 and C_3 is thus six times as high, since these combinations consist of four flows each. When the total work load (ρ) is 0.8 the arrival probability for the flows in C_2 is 0.08, for the other ten flows it is 0.24. The results are reported in Table 6.9.

Results

Under cyclic control a thin combination has to be served before a next thick combination can is helped, but it service may be stopped (well) before its queues become exhausted. Nevertheless, the long-run average waiting time at a thin combination is not very high under RV1 or RV2. The long-run average waiting time at C_2 is higher than at the other queues, but it is (considerably) lower than under FC, XC and XC-1. Indeed the relative

values prevent long queues at the thin combinations. The lowest waiting time to C_2 is experienced under XC-2, but the difference with cyclic RV1 and RV2 is small. The overall average waiting time under XC-2 is 28% above that under cyclic RV1 and RV2.

Under acyclic control the service of C_2 may be postponed in order to serve a thicker combination. Since the visiting order is no longer cyclic, the service of a thin combination is no longer enforced in the action space. Because of this, one observes a higher average waiting time at C_2, than under cyclic control.

The difference in the average waiting time between cyclic and acyclic control is more visible, now that not all combinations have identical loads. Moreover RV2M yields a somewhat lower waiting time than RV1. Compared to FC the waiting time under RV2M at the eight queues of C_1 and C_3 has dropped by more than 30% (14 seconds), while the waiting time at C_2 has increased by only 15% (8 seconds). Acyclic anticipative-exhaustive control XA-2 performs very bad, since C_2 is only served when the length of one it's queues is the longest over all 12 queues; due to the low arrival rate this may take a relatively long time during which any cars present have to wait.

All RV policies yield thus fairly low waiting times even at the thin flows, apparently the relative values avoid queues to become very long.

6.6.3 Conclusions for F12C4 cases:

- The relative values used in the RV policies prevent queues to become long, even for a thin combination, which contributes little to the total workload.
- When the total workload is low, acyclic control may be preferred; when the total workload is high the difference between cyclic RV policies and acyclic RV policies is small.
- RV1 is the preferred policy for cyclic control, since RV2 is more complicated while it does not lead to significantly lower average waiting times.
- For acyclic control the best option seems to be RV2M, although RV1 or RV1M may be may be favored for requiring less computations.

6.7 Conclusions and Discussion

6.7.1 Conclusions

The dynamic control of traffic lights, such that the long-run average waiting time over all flows is minimized, can be formulated as an MDP. However an optimal solution can be computed only for intersections where the number of queues is small. For most realistic intersections, one relies on heuristics such as FC, and vehicle actuated control, which is a mixture between exhaustive control and FC. Under FC the system has a well-structured state space, which can be decomposed into subspaces for each flow. The relative values of states under FC can thus be computed numerically for each flow in isolation of all others. These relative values are used to develop what we call RV policies, in which a best position (slot number) in the cycle is determined given the queue lengths.

RV1 is based on a one-step policy improvement algorithm with FC as the initial policy. RV2 optimizes every slot over two successive time jumps, although only the first time is implemented. RV1M and RV2M do allow to modify the cyclic order while evaluating the time jumps in a fixed cycle. Based on a number of test cases for two different infrastructures we conclude:

- RV1 yields a great improvement of FC and exhaustive control policies.
- RV1 is robust in the chosen cycle length for the underlying FC: RV1 performs also very good when based on a sub-optimal FC. This is important since finding an optimal FC is in general difficult.
- When the workload is high, the more advanced RV policies (such as RV2, RV1M, and RV2M) show only a minor improvement compared to RV1.
- When the workload is low, acyclic control, e.g. by RV1M, is favored.
- The one-step policy improvement algorithm, which results in policy RV1, is promising for reducing the waiting times at signalized intersections.

6.7.2 Discussion

The quality and robustness of the RV policies have been investigated and compared based on several interesting cases. In this section we briefly discuss how the RV policies can deal with additional objectives and with a number of complexities that arise in practice.

Different objectives in cost function

We have focussed on minimizing the overall long-run average waiting time. Another criterion, commonly studied in traffic light engineering, is to minimize the number of stops made by the cars. The cost structure can easily capture this criterion by including cost for each stop a car makes in the queue. The question to minimize the number of stops may be more prevalent when dealing with networks of intersection.

Serving pedestrians and cyclists

In the cases that we have studied we excluded flows of pedestrians and cyclists. The departure process of pedestrians and also that of cyclists differ from the departure process of cars, since when the light turns green all pedestrians waiting at a cross walk start crossing the street more or less at the same time. Furthermore, the number of pedestrians may not be known to the controller: the controller only receives information about whether there are any pedestrians are waiting. The same may hold to cyclists: the number of cyclists waiting at an intersection may not be known to the controller and cyclist do not leave one-by-one. At some intersections cameras or induction loops are present to count or estimate the number of cyclists waiting, but usually their waiting time cannot be estimated accurately, since only the first moment of pressing a button to call for service is registered.

Therefore cyclists and pedestrians require a special treatment. In practice the green period is fixed to a pre-specified time that a cyclist or pedestrian needs to cross an intersection, which is independent of the actual number of cyclists or pedestrians waiting at a light. The service of cyclists or pedestrians can be included in FC as special combinations. In FC fixed periods are then included during which cyclists or pedestrians are served. These fixed period may be skipped only when no cyclist or pedestrian is present. Furthermore, the RV policies are not allowed to shorten the green period of these special combinations.

Under acyclic dynamic control we may need to specify a 'soft' bound b on their waiting time; when pedestrians or cyclist are waiting more than b slots, they are the next

combination to get green. The process of serving car traffic is then continued until some pedestrian(s) or cyclist(s) wait for more than (at least) b slots. Then, as soon as a green period to car traffic ends, the respective lights for pedestrians and cyclists turn green.

Public transport

Trams and busses often drive on separate lanes in Dutch cities, and at some places they get priority over car traffic for political and environmental reasons. Green periods to cars are ended instantaneously when a bus or tram approaches an intersection. The RV policies can of course be interrupted in this way, but for the minimization of the waiting time we suggest to let RV1M decide about when to end a green period based on the number of cars waiting at the queues. Public transport thus gets priority as soon as an all-red slot is visited. Such a priority setting fits better to the objective of minimizing the long-run average waiting time, while it still provides priority to public transport.

When at an intersection busses and trams arrive very frequently one may want to include one or more special combinations in FC, just as for cyclists and pedestrians. When queueing of trams and busses is allowed, one may even compute relative values of the queue lengths for public transport. To give buses and trams a higher weight one may incur (much) higher cost for busses and trams for each slot that they are queued. One may even introduce a cost structure that is progressive in the number of queued busses and trams.

More complex cycles and sub-cycles in FC

In the cases that we have studied, the set of flows $\mathcal{F}$ was split into C mutually exclusive subsets, called combinations. Under cyclic control each flow is visited once every cycle, and an all-red slot applies when switching between two combinations. In general, a flow may be part of multiple combinations, and the lights of one or more flows may stay green while switching form one combination to another. The cyclic RV1 and RV2 policies can easily be modified to cases where a flow is visited multiple times in a cycle, since one only needs to know during which slots of FC its lights are green, yellow, and red.

In the practice of traffic light engineering, a cycle may contain two (or more) sub-cycles: e.g. in both sub-cycles all or just a part of the flows is served, but the first sub-cycle may be longer than the second. The RV policies can easily be extended to make time jumps in a cycle of multiple sub-cycles: one then allows to jump from a short cycle to a longer

cycles. Since RV1 is hardly sensitive to changes in the cycle length we do not expect great improvements of considering sub-cycles.

Serving conflicting flows simultaneously

Thus far we have presumed that conflicting traffic flows are not in the same combination and thus do not get right of way simultaneously. In practice, some conflicting flows may receive green simultaneously, in which case basic traffic rules apply to solve any conflicts. For example, left-turning traffic flow f may be served together with opposite thru traffic originating from flow h. Flow f than needs to give right of way to flow h, whenever cars leave queue h.

In deriving the relative values of states, one thus needs to acknowledge that the departure process of f depends on the number of cars present at queue h. The transitions become a bit more involved. Suppose that only flow h has priority over f, and no flow in the same combination has priority over h. When slot t grants green to flows f (and h), Equation (6.19) then becomes:

$$\begin{aligned}
V^f_{n+1}(t, q_f, q_h) \;=\; q_f \;&+\; \lambda_f \cdot \lambda_h \;\cdot\; V^f_n(t+1, q_f+1, q_h) \\
&+\; \lambda_f \cdot (1-\lambda_h) \;\cdot\; \big[\, V^f_n(t+1, q_f+1, q_h-1) \cdot I(q_h \geq 1) \\
&\qquad\qquad + V^f_n(t+1, q_f, 0) \cdot I(q_h = 0) \,\big] \\
&+\; (1-\lambda_f) \cdot (1-\lambda_h) \;\cdot\; \big[\, V^f_n(t+1, q_f, q_h-1) \cdot I(q_h \geq 1) \\
&\qquad\qquad + V^f_n(t+1, (q_f-1)^+, 0) \cdot I(q_h = 0) \,\big] \\
&+\; (1-\lambda_f) \cdot \lambda_h \;\cdot\; V^f_n(t+1, q_f+1, (q_h-1)^+).
\end{aligned}$$

The value vector of flow h can be computed as before as, using Equation (6.19), since its departure process does not depend on any other flow. The state space is thus decomposed in a somewhat different way. The relative values are computed in a very similar way as before, but the relative value vector of flow f is three-dimensional, while that of h is two dimensional: $v^f_{rel}(t, q_f, q_h)$ versus $v^h_{rel}(t, q_h)$.

Outlook

In the next chapters, the RV policies are extended to include information on near future arrivals, and the policies are applied to networks of intersections.