Delft University of Technology

Delft Center for Systems and Control

Technical report 07-020

Look-ahead traffic-adaptive signal control*

R.T. van Katwijk, B. De Schutter, and J. Hellendoorn

If you want to cite this report, please use the following reference instead:
R.T. van Katwijk, B. De Schutter, and J. Hellendoorn, "Look-ahead traffic-adaptive signal control," *Proceedings of the 6th European Congress on Intelligent Transport Systems and Services (ITS'07)*, Aalborg, Denmark, 12 pp., June 2007. Paper 2417.

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.51.19 (secretary)
fax: +31-15-278.66.79
URL: `http://www.dcsc.tudelft.nl`

*This report can also be downloaded via `http://pub.deschutter.info/abs/07_020.html`

LOOK-AHEAD TRAFFIC-ADAPTIVE SIGNAL CONTROL

R.T. van Katwijk[*†], B. De Schutter[†‡], and J. Hellendoorn[†]

[*] Business Unit Mobility and Logistics, Netherlands Organization for Applied Scientific Research (TNO), Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands, phone: +31-15-269 74 73, Ronald.vanKatwijk@tno.nl

[†] Delft Center for Systems and Control (DCSC), Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands, email: b@deschutter.info, j.hellendoorn@tudelft.nl

[‡] Marine and Transport Technology department, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands

ABSTRACT

The design of a look-ahead traffic-adaptive control system is as well a science as an art. Along the way compromises have to be made to end up with a workable system that is not only able to come up with good signal timings, but also to deliver them on time. We propose a taxonomy of look-ahead traffic-adaptive control algorithms based both on their underlying principles and the compromises that were made to come up with a workable, albeit less optimal system. A new hybrid algorithm is subsequently defined that combines the strengths of all current approaches and incorporates constraint programming techniques.

INTRODUCTION

A common function of a traffic controller is to seek to minimize the delay experienced by vehicles through manipulation of the traffic signal timings. There are various levels of sophistication in traffic signal control system applications. With recent advances in communication network, computer, and sensor technologies, there is increasing interest in the development of look-ahead traffic-adaptive signal control systems. Numerous systems have been proposed including PRODYN ([9], [1], [8]), UTOPIA-SPOT ([10]), OPAC ([7], [5], [6]), RHODES ([16], [11]), SPPORT ([2], [3]) and ALLONS-D [13], [14], [15]. This overview is based on these references and the references contained therein.

The design of a look-ahead traffic-adaptive control system is as well a science as an art. Along the way compromises have to be made in order to end up with a workable system that is not only able to come up with good signal timings, but is also able to deliver them on time. Each system mentioned above has a different approach in dealing with the computational complexity of determining the best set of signal timings.

All systems therefore have their own specific strengths and weaknesses that make that system more - or - less suited for particular networks and traffic demand patterns. In this paper we

propose a taxonomy of the various look-ahead traffic-adaptive systems based both on their underlying principles and the compromises that were made to come up with a workable, albeit less optimal system.

A TAXONOMY OF APPROACHES

Traffic signal control essentially comes down to making the right decisions at the right time. As such the traffic signal control problem solved by all look-ahead traffic-adaptive systems can be formulated in the form of a general decision problem. This decision problem in turn can be represented as a simple decision tree. The root of a decision tree represents the current state $s_i, s_i \in S$, where i is the current time index and S is the set of all states. The cost involved in order to transition to the subsequent state, s_{i+1} when deciding for an action u_i is denoted by c_i.

In general, the nodes of a search tree represent choices. These choices are mutually exclusive and therefore partition the search space into two or more simpler sub-problems. At each time step, the controller observes the system's current state s_i, and selects a control action, $u \in U_i$, where u is the action and U_i is the finite set of actions available to a controller in state s_i. When the controller chooses an action $u \in U_i$, the cost incurred by taking that action and subsequently transition to state s_j with probability $p_{i,j}(u)$, is denoted by $c(i)$. The objective of a look-ahead traffic-adaptive system is to find an optimal sequence of actions.

Looking at the various look-ahead traffic-adaptive systems we can discern the following features on which they differ:

- the *optimization method*. Is the optimal sequence of actions found by searching the decision tree using a rule-based method, or an approach based on dynamic programming or branch-and-bound?
- the possible *actions* (u_i) considered in the optimization. Is the order in which phases can be given green to predetermined or can this be determined (and optimized) on-line?
- the length and resolution of the *planning horizon* over which an optimal sequence of actions is sought (i.e., the depth of the decision tree). Is the length of the horizon fixed (e.g., 2 minutes) or dependent on current traffic conditions? Is the resolution static (e.g., is the horizon divided into 5 seconds intervals) or is it dynamic (e.g., dependent on projected arrival times)?
- the *update frequency*. How often can the optimization be done (i.e every 0.5 seconds or every 5 seconds)?
- the *delay model*. How is the performance (c_i) of each evaluated action (u_i) evaluated? How accurate is the model used in optimizing the signal timings? Is a fast vertical queuing model used instead of a slow but possibly more accurate simulation model?

The following sections elaborate on each of these features and how each system differs in how these features are filled in.

OPTIMIZATION METHOD

The objective of the system is to operate such that the total cost over the entire planning horizon is minimized. Thus, the task of the controller is to obtain a sequence of control actions $[u_0, u_1, \ldots u_T]$, also referred to as a policy or control trajectory, such that the expected cost is

minimized. In the case of an infinite planning horizon, a discount factor, $\gamma < 1$, is typically applied to future costs to obtain a finite estimate of the *cost-to-go* from the current state *i*, denoted by $f(i)$. The optimal cost-to-go value, denoted by $f^*(i)$, is a function of the immediate cost of applying the control plus the expected cost-to-go from the subsequent state, a relationship encapsulated in the following recursive expression (also known as Bellman's Equation [4]):

$$f^*(i) = \min_{u \in U(i)} \left\{ c_i(u) + \gamma \sum_{j \in S} p_{i,j}(u) f^*(j) \right\} \tag{1}$$

As the decision space has a tree-like structure, the search for the optimal sequence of decisions corresponds to building the tree. An exhaustive search of the entire decision space results in a full tree being built. Since search space size grows exponentially with problem size, it is not possible to explore all assignments except for the smallest problems. The only way out is to not look at the whole search space. Efficiency in searching the decision space is considered by the degree to which the entire tree will not have to be built to find an optimal path. In [17] several well-known algorithms are assessed based on computational speed and on the quality of the results (in terms of delay).

Dynamic programming and branch-and-bound (and combinations thereof) are the techniques that are predominantly used in look-ahead traffic-adaptive systems.

Dynamic Programming

The applicability of the approach depends on the opportunities for state aggregation within the decision tree. The strength of dynamic programming is that it can prevent that optimal solutions to subproblems it has already solved are recomputed. In order to do this, the solutions to already solved problems are saved. This approach is called memoization (not memorization, although this term also fits). It is however only possible to reuse a previous solution when states and thus the corresponding subproblems can be considered equal. RHODES, PRODYN, OPAC and SPPORT all employ dynamic programming as their method of optimization.

Branch-and-Bound

Branch-and-bound is a general method for finding optimal solutions of various optimization problems, especially in discrete and combinatorial optimization. It belongs to the class of implicit enumeration methods. One way to do this is by proving that certain areas of the space contain no solutions. The core of the approach is the simple observation that (for a minimization task) if the lower bound for a sub-problem A from the search tree is greater than the upper bound for any other (previously examined) sub-problem B, then A may be safely discarded from the search. This is the bounding-part of the branch-and-bound approach. Of the systems reviewed only ALLONS-D and SPOT employ the branch-and-bound method in its pure form. RHODES employs a hybrid system in which branch-and-bound techniques are applied within a dynamic programming framework.

In order to obtain a tight upper bound an initial path must be established through the search tree for which it is most likely to obtain a good solution. This involves that initially parts of the search space that are unlikely to contain good solutions are ignored. This is done by using *heuristics*. Heuristics are used to explore promising areas of the search tree first. This can

be done by using problem specific knowledge (often borrowed from current practices in tuning traffic responsive and vehicle-actuated controllers) or by reusing information gained from previous optimizations.

ACTION SPACE

The width of the tree to be searched is dependent on the number of decisions that can be made at each point in time. In its simplest form the choice available is that between extending the current phase or switching to the next phase. This is the approach taken in OPAC, ALLONS-D, PRODYN and SPPORT. Although this approach significantly reduces the number of options to consider, it does not allow arbitrary phase sequencing. In its most elaborate form the choice available is that between phases. This approach allows the arbitrary sequencing of phases but comes at a cost in the width of the search tree. This is the approach chosen by UTOPIA-SPOT. A compromise between these two extremes is found in allowing phase skipping. When the skipping of phases is allowed any phase sequence can be attained. This is the approach taken in the COP-system. The downside of this approach is that when the initial phase sequence is chosen wrong the gain in width is counteracted with an increase in tree depth. This is not strictly necessary however. It is in principle possible to choose the next phase dynamically.

PLANNING HORIZON

Look-ahead traffic-adaptive systems employ a traffic model to evaluate alternative traffic signal timings over a planning horizon. The length of the planning horizon as well as how the horizon is split up into successive intervals differs between each system. Typically however the horizon has a fixed length (of typically 1 to 2 minutes) and is subdivided into fixed intervals. From their descriptions we can deduce that OPAC, PRODYN, SPPORT, and ALLONS-D all use or have used 5-second time-steps.

If the horizon is chosen too short and the optimization algorithm is faced with the choice whether to a) completely serve a phase discharging at a slow rate, or b) preempt that phase in order to switch to a phase with a higher discharge rate, it would chose for the latter as this brings about the biggest benefits in the short term. This is why many of the systems that employ shorter horizons have introduced terminal costs in order to penalize residual queues at the end of the horizon [12], [17].

The ALLONS-D algorithm takes a different approach wherein the length of the horizon depends on the current traffic situation. The ALLONS-D algorithms enlarges the horizon until it finds a solution in which all projected arrivals are cleared. Although the idea of a horizon that shrinks or grows dependent on the traffic situation sounds attractive, it might not turn out this way in the case of the ALLONS-D algorithm. In saturated conditions - with many projected arrivals - the length of the horizon might become so large that the optimization method used by ALLONS-D might be unable to come up with an answer in time.

The approach where both the length of the planning horizon and the length of the time intervals in which it is subdivided are variable is not applied by any of the algorithms reviewed.

UPDATE FREQUENCY

Look-ahead traffic-adaptive systems rely on predicted arrivals. As the distance over which these arrivals are predicted increases the reliability of these predictions often decreases. This is why a rolling horizon is often applied. The concept of a rolling horizon originated in operations research and is used to determine a short term policy based on a longer term analysis. All systems reviewed that depend on arrival predictions employ the concept of a rolling horizon. These algorithms implement only the first (few) action(s) of the control plan after which a new optimization is performed.

The amount of time that passes between each subsequent optimization (the roll - or - commitment period) is, for all systems reviewed, equal to the length of the intervals which subdivide the planning horizon. For most systems reviewed the length of the interval is typically equal to 5 seconds. Waiting 5 seconds between each optimization can however have a significant impact on delay. Note that, as all systems choose their commitment period equal to the length of the interval in which the planning horizon is subdivided, switching from a 5-second to a 1-second decision resolution increases the number of time-steps in the planning horizon by a multiple of 5. This imposes too much of an increase in computational effort for many algorithms to solve in real-time. Thus, the typical trade-off is to also decrease the duration of the planning horizon.

DELAY MODEL

All systems reviewed consider individual vehicles in determining the control delay brought about by a chosen control plan. In that respect all models can be considered to use a meso- to microscopic model. However, because the delay model is applied many times when exploring the search tree, all models have to make some sacrifices with respect to the level of detail on the employed model.

Known, commercially available, microscopic simulation models like Paramics, VISSIM, and AIMSUN are unfit for use in real-time optimization. This is why simple event-based and cellular automaton models are predominantly used within these systems. At first these systems employed vertical queuing models, but many systems have since switched to using horizontal queuing models so that queue spillbacks to upstream intersections can explicitly be considered in the optimization.

A NEW HYBRID ALGORITHM

At the basis of our new hybrid algorithm we use a modified version of the dynamic programming approach as defined by [16]. In the analysis performed in [17] of the various traffic control algorithms this algorithm proved to be one of the best performing algorithms. The algorithm uses a dynamic programming approach in which each stage represents a phase as opposed to a approach in which each stage represents an interval of time. The exact number of stages used in the DP is a by-product of the computations. For the delay model we use a modified version of the delay model as defined in [14]. This delay-model considers the weighted time delay for each vehicle as opposed to other approaches which use the number of vehicles in the queue multiplied by the interval of time in which the vehicles were queued. In the following a 'signal group' denotes the signals that are specific for a traffic stream that has its own queue, whereas the term 'phase' is used to denote the set of signal groups which can be served at the same time.

The term 'stage' is exclusively used to denote a stage in the algorithm; it is thus not used as a synonym for 'phase'.

THE OPTIMIZATION METHOD

Notation. We introduce the following notation

$$
\begin{array}{rcl}
j & \equiv & \text{Index for stages of the DP.} \\
t & \equiv & \text{Variable denoting the total number of allocated time-steps.} \\
T & \equiv & \text{Total number of discrete time-steps. Each period of length } \Delta \text{ is indexed by } t \in [0,T]. \\
P & \equiv & \text{Set of phases. The cardinality of this set will be denoted by } |P|. \\
\rho & \equiv & \text{Index for phases in } P. \\
\Omega & \equiv & \text{Set of signal groups. The cardinality of this set will be denoted by } |\Omega|. \\
\omega & \equiv & \text{Index for signal groups in } \Omega. \\
f^{\omega}_{\text{saturation}} & \equiv & \text{The saturation flow for a signal group.} \\
\gamma^{\omega} & \equiv & \text{The minimum green time for a signal group} \\
\kappa^{\omega} & \equiv & \text{The maximum green time for a signal group} \\
r^{\omega} & \equiv & \text{The effective all-red interval for a signal group (integer number of time-steps)} \\
r(\omega_{j-1},\omega_j) & \equiv & \text{The effective all-red interval for a signal group (integer number of time-steps) when switching from signal group } \omega_i \text{ to } \omega_j \\
U_j(t) & \equiv & \text{Set of feasible control decisions, given } t. \\
u_j(t) & \equiv & \text{Control variable denoting the amount of time allocated to stage } j \\
u^*_j(t) & \equiv & \text{Optimal value for the control variable} \\
c_j(t,u_j) & \equiv & \text{Cost of control } u_j \text{ when applied at time } t \text{ in stage } j \\
f_j & \equiv & \text{Value function (cumulative value of prior optimal costs) at time } t \text{ in stage } j \\
A_{\omega}(a,b) & \equiv & \text{A function of scalars } (a,b) \text{ denoting the arrivals in the time interval } [a,b\rangle \text{ and requesting signal group } \omega. \text{ If } a=b, \text{ we put } A_{\phi}(a,b)=0. \\
L_{j,t,\omega} & \equiv & \text{Queued arrivals at time } t \text{ in stage } j \text{ resulting from applying control variable } u_j(t) \\
L^*_{j,t,\omega} & \equiv & \text{Queued arrivals at time } t \text{ in stage } j \text{ resulting from applying the optimal control variable } u^*_j(t) \\
i & \equiv & \text{Index for vehicles arriving at the intersection for signal group } \omega. \\
l^{\omega}_i & \equiv & \text{Predicted arrival time of vehicle } i.
\end{array}
$$

In general the values control variable $u_j(t)$ can assume is bounded by a minimum and maximum. These can be calculated from the minimum and maximum green times per signal group and the red time needed for vehicles so safely clear the intersection:

$$
\begin{array}{rcl}
u^{\min}_j & = & \max\{r(\omega_{j-1},\omega_j)+\gamma^{\omega}_j | \omega_{j-1}\in\rho_{j-1} \wedge \omega_j \in \rho_j\} \\
u^{\max}_j & = & \min\{\kappa^{\omega}_j | \omega_j \in \rho_j\}
\end{array}
$$

Given a value for the state variable t_j, the control variable u_j can assume values from the following discrete set:

$$
U_j = \begin{cases} \{0\} & \text{if } t_j < u^{\min}_j \\ \{0, u^{\min}_j, \ldots, u^{\max}_j\} & \text{otherwise.} \end{cases}
$$

We shall assume that the performance index has the following form

$$c_1(t_1,u_1) \circ c_2(t_2,u_2) \ldots \circ \ldots$$

where the operator $\circ$ can be the $+$-operator when minimizing delay or number of stops or the max-operator when minimizing queue length

The forward recursion of the DP-algorithm is presented in algorithm 1. To initialize, the value function $f_0(0) \leftarrow 0$. The DP algorithm starts with stage $j = 0$, and proceeds recursively to $j = 1,2,\ldots$. At each stage, the method calculates the best control decision $u_j^*(t)$ for each possible value of the state variable t. This process is summarized in algorithm 1.

Algorithm 1 Forward recursion

$j \leftarrow 0$
$f_j(0) \leftarrow 0$
while $\neg$ stop **do**
  for $t_j \in T$ **do**
    $f_j^*(t) \leftarrow \infty$
    for $\forall u_j \in U_j(t)$ **do**
      $f_j(t_j) \leftarrow c_j(t_j,u_j) \circ f_{j-1}(t_j - u_j)$
      if $f_j(t_j) < f_j^*(t_j)$ **then**
        $f_j^*(t_j) \leftarrow f_j(t_j)$
        $u_j^*(t_j) \leftarrow u_j$
        $L_{j,t,\omega}^*(t_j) \leftarrow L_{j,t,\omega}$
      end if
    end for
  end for
  $j \leftarrow j+1$
end while

The optimization procedure is stopped when it can be determined that there is nothing to be gained by adding an extra phase to the signal plan. This is the case when the values of the value function f_j is equal to that of the previous $|P|-1$ phases.

Algorithm 2 Stopping criterion

if $(j \geq |P|) \wedge (\forall \rho \in \{1,\ldots,|P|-1\}(f_{j-\rho}(T) = f_j(T)))$ **then**
  return TRUE
else
  return FALSE
end if

The procedure to retrieve the optimal trajectory is provided in algorithm 3.

THE DELAY MODEL

In general for stage j, we assume that the quantities $L_{j-1,t_{j-1},\omega}$ are available for all values of t_{j-1}. We now discuss how we calculate the cost and the queues that result when a control action is applied for any given value of t_j. The resulting queues $L_{j,t,\omega}$ are recorded as permanent queues $L_{j,t,\omega}^*$ if a new u_j^* is identified.

Algorithm 3 Retrieval of optimal policy

```
t_j^* ← T
for j = j to 0 do
    if j > 1 then
        t_{j-1}^* ← t_j^* − u_j^*(t_j^*))
    end if
end for
```

The vehicles affected for the chosen control u_j can be determined as follows:

$$L_{j,t,\omega} = L_{j-1,t_{j-1},\omega} \cup A(t_{j-1}, t_j)$$

Each of the vehicles in $L_{j,t,\omega}$ can contribute to delay when the signal is red, green, or yellow. A vehicle stopped at a red light contributes an amount of delay equal to the duration of the red light. In addition, delay can be accumulated by vehicles at a green light. For instance, for long (standing) queues, only a small portion in the front of the queue will clear the intersection when the light turns green. Even for the vehicles that are close enough to the intersection to pass through, there will still be some delay incurred. The latter type of delay accounts for the minimum amount of time required for the vehicle immediately in front to respond and move through the intersection. This minimum time required for each car to depart will be referred to as the headway constraint. It is calculated using the saturation flow for a signal group. The maximum number of vehicles that can leave an intersection during time period, is dependent on the saturation flow and the length of the interval of green light. The saturation flow is signal group specific.

Notation. We introduce the following variables:

o_i^ω $\equiv$ Time at which the i^{th} vehicle waiting at signal group ω is able to enter the intersection

k^ω $\equiv$ Number of vehicles waiting at signal group ω that are able to enter the intersection

which are initialized as follows:

$$o_i^\omega = t - (u - r^\omega) + \sum_0^i h^\omega, \text{ where } h^\omega = \frac{1}{f^\omega_{\text{saturation}}}$$
$$k^\omega = f^\omega_{\text{saturation}} \cdot ((u - r^\omega))$$

Four indicator variables are subsequently calculated that can be used to determine whether a vehicle is:

- *q*ueued up at the start of the green light:
$$q_i^\omega = \begin{cases} \text{true}, & l_i^\omega \le t - (u - r^\omega) \\ \text{false}, & \text{else} \end{cases}$$
- able to *d*epart during the green period:
$$d_i^\omega = \begin{cases} \text{true}, & i \le k^\omega((u - r^\omega)) \\ \text{false}, & \text{else} \end{cases}$$
- partially *c*onstrained by the headway of vehicles in front:
$$c_i^\omega = \begin{cases} \text{true}, & l_i^\omega \le o_i^\omega \\ \text{false}, & \text{else} \end{cases}$$

- constrained by the headway of vehicles in front of it *and p*rohibited from leaving during the green period:
$$p_i^\omega = \begin{cases} 1, & t \leq o_i^\omega \\ 0, & \text{else} \end{cases}$$

Delay is subsequently calculated as follows when a vehicle can be determined to

1. wait at a red light $f_i^\omega = \min\left\{u, t - l_i^\omega\right\}$,
2. be queued at a light that turns green but that cannot leave during the green period due to vehicles ahead of it ($q_i^\omega \wedge \neg d_i^\omega$), i.e., $f_i^\omega = (u - r^\omega) + \min\left\{r^\omega, t - (u - r^\omega) - l_i^\omega)\right\}$,
3. be queued at a light that turns green and can leave but only after waiting for the vehicles ahead of it to depart ($q_i^\omega \wedge d_i^\omega$): $f_i^\omega = (i-1) * h + \min\left\{r^\omega, t - (u - r^\omega) - l_i^\omega)\right\}$,
4. arrive during a green period and cannot leave due to vehicles ahead of it ($\neg q_i^\omega \wedge p_i^\omega$), and $f_i^\omega = t - l_i^\omega$,
5. arrive during a green period and can leave but only after waiting for the vehicles ahead of it to depart ($\neg q_i^\omega \wedge c_i^\omega \wedge \neg p_i^\omega$): $f_i^\omega = o_i^\omega - l_i^\omega$

The cost of the control action is determined by summing the cost for all vehicles:

$$c = \sum_{\forall \omega \in \Omega} \sum_{\forall l_i^\omega \in L^\omega} f_i^\omega$$

The queues that remain $L_{j,t,\omega}$ after the application of the control u_j is easily determined by removing vehicles that were able to depart during the control-interval: $L^\omega = L^\omega \setminus \{l_i^\omega | d_i^\omega = 1\}$.

The strategy is as follows:

- Given t_j calculate based on the existing queues $L_{j-1,t_{j-1},\omega}$ and pending arrivals $A(t_{j-1}, t_j)$ the performance index c_{j,t_j,u_j}. The queues that remain $L_{j,t,\omega}$ after the application of the control u_j are calculated as a side-effect
- The queues $L_{j,t,\omega}$ are recorded as permanent queues $L^*_{j,t,\omega}$ if a new u^*_j is identified.

ANALYSIS

In order to test the performance of the developed algorithm we used the traffic management test-bed described in [18]. This test-bed enabled us to interface the control algorithm above with the Paramics simulation model developed by Quadstone Ltd.

SCENARIO

The simulations were performed for a 4-arm intersection with a separate, single, approach lane for each turn. There are 12 signal groups in total (one for each turn). Depending on the simulation the intersection operated under either 4- or 8-phase traffic control. The phases are shown in 1. For simplicity, the right turns are omitted and assumed to proceed with the through movements.

The total demand for the intersection is set to 4400 vehicles per hour, which is distributed over the signal groups in proportion to the saturation flow rate of each signal group. Maximum green times for each phase were subsequently determined according to Webster's method.

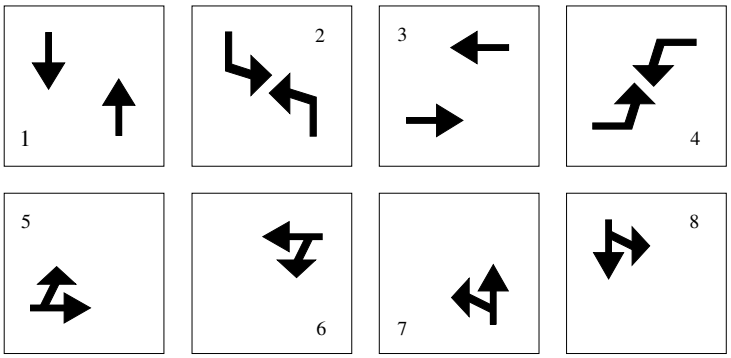
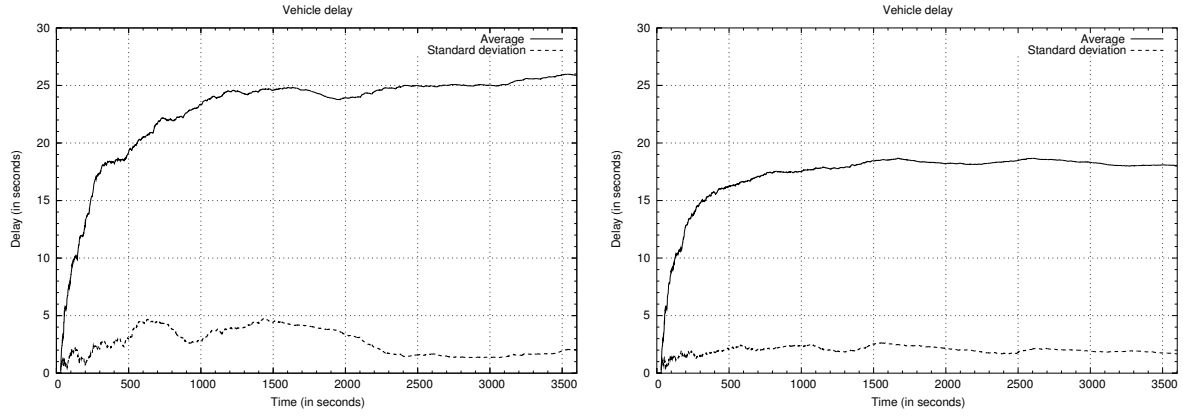


Figure 1: The phases used in the optimization of the intersection

RESULTS

The results have been obtained from a number of one-hour simulations each with a different random seed. The figures show the average delay per vehicle as it evolves over a one hour period. As each simulation starts with an empty network some time is needed before vehicles start to arrive at the intersection and the intersection becomes fully stressed. This is the warm-up period of a simulation. We can be sure the warm-up period has ended when the plot of the average vehicle delay stabilizes into a line that is more-or-less horizontal.

In Figure 1 the average delay of a vehicle is shown when the intersection is operating under a) traffic-actuated control and b) look-ahead traffic-adaptive control. The average delay encountered by a vehicle in the traffic-actuated controlled case can (in Figure 2(a)) be seen to stabilize at about 25 seconds per vehicle whereas the average delay in the traffic-adaptive controlled case can (in Figure 2(b)) be seen to stabilize at about 18 seconds per vehicle. This proves that there is certainly something to be gained by planning for future arrivals.



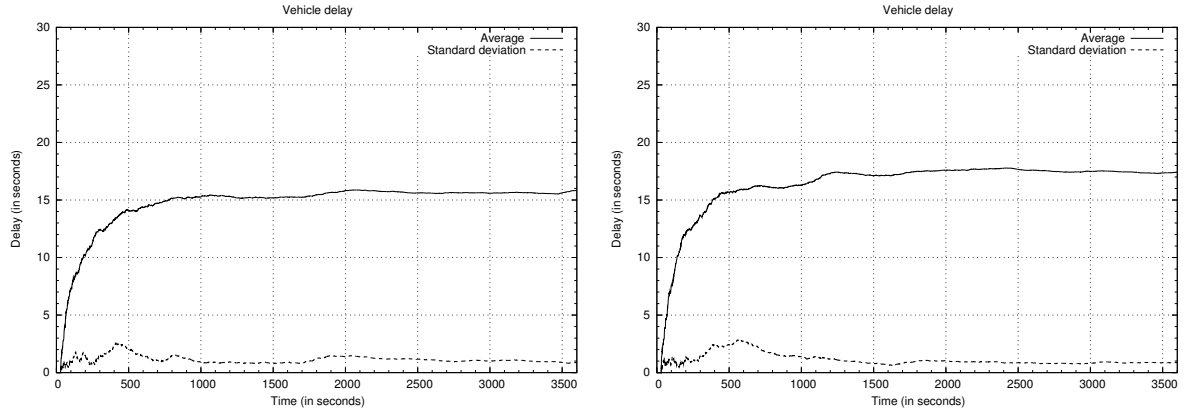
(a) Vehicle actuated control

(b) Look-ahead traffic-adaptive control

Figure 2: Performance for an intersection with 4 phases

In Figure 2(b) the average delay encountered by a vehicle was plotted for a 4-phase traffic-adaptive controller. When the number of phases is extended to 8 (i.e., the number of signal group configurations supported by a typical 2-ring NEMA traffic-actuated controller) the average delay can (in Figure 3(a)) be seen to have reduced even further to about 16 seconds per vehicle. When computational complexity demands a longer update-interval (e.g., 5 seconds) the average delay per vehicle can (in Figure 3(b)) be seen to rise again to about 18 seconds per vehicle. This clearly shows the impact of the trade-off made regarding the choice of the

action space, planning horizon, update frequency and delay model. The better the optimization method employed is, the less comprises have to be made.



(a) 1-second update interval

(b) 5-second update interval

Figure 3: Performance for an intersection with 8-phase traffic-adaptive control

CONCLUSIONS AND FURTHER WORK

The previous has shown that there are many different ways to configure a traffic-adaptive system. They differ with respect to the search algorithm applied, the length and resolution of the planning horizon, the update frequency and the delay model used. Unfortunately computational boundaries still prevent the configuration of a traffic-adaptive system in which no compromises have to be made in order to end up with a workable system that is a) able to come up with good signal timings and b) is able to deliver them on time.

In order to gain the full advantage of traffic-adaptive control, the system should be carefully tuned. Computationally complexity, geometry of an intersection, and demand patterns should be carefully considered. This is why it is so difficult to judge the merits of an algorithm based on the benchmark studies. More often than not comparisons are made between algorithms that have been tuned recently and those that have not been tuned for some time. We are currently trying to find a site which allows us to make a fair comparison.

ACKNOWLEDGMENTS

Research funded by the TNO spearhead program ”Sustainable Mobility Intelligent Transport Systems (SUMMITS)”, the Transport Research Centre Delft, and the BSIK project “Towards Sustainable Mobility (TRANSUMO)”.

REFERENCES

[1] J.F. Barriere, J.L. Farges, and J.J. Henry. Decentralization vs. hierarchy in optimal traffic control. In *Proc. of the 5th IFAC/IFIP/IFORS Symposium on Control in Transportation Systems*, Vienna, Austria, July 1986.

[2] F. Dion and B. Hellinga. A methodology for obtaining signal coordination within a distributed real- time network signal control system with transit priority. In *Proc. of the 80th Annual Meeting of the Tranportation Research Board (TRB'01)*, May 2001. 80th Annual Meeting of TRB.

[3] F. Dion and B. Hellinga. A rule-based real-time traffic responsive signal control system with transit priority: Application to an isolated intersection. *Transportation Research Part B*, 36:325–343, May 2002.

[4] S.E. Dreyfus and A.M. Law. *The Art and Theory of Dynamic Programming*. Academic Press, New York, 1977.

[5] N. H. Gartner, C. Stamatiadis, and P. J Tarnoff. Development of advanced traffic signal control strategies for intelligent transportation systems: Multilevel design. *Transportation Research Record*, (1494):98–105, 1995.

[6] Nathan H. Gartner, P.J. Tarnoff, and C.M. Andrews. Evaluation of the optimized policies for adaptive signal control strategy. *Transportation Research Record*, (1683):105–114, 1999.

[7] N.H. Gartner. Opac: A demand-responsive strategy for traffic signal control. *Transportation Research Record*, (906):75–81, 1983.

[8] J.J. Henry and J.L. Farges. Prodyn. In *Proc. of the 6th IFAC/IFIP/IFORS Symposium on Control, Computers, and Communications in Transportation*, Paris, France, September 1989.

[9] J.J. Henry, J.L. Farges, and J. Tuffal. The prodyn real time traffic algorithm. In *Proc. of the 4th IFAC/IFIP/IFORS Symposium on Control in Transportation Systems*, pages 307–312, Baden-Baden, Germany, April 1983.

[10] V. Mauro and C. Di Taranto. Utopia. In *Proc. of the 2nd IFAC-IFIP-IFORS Symposium on Traffic Control and Transportation Systems*, pages 575–597, 1989.

[11] P. Mirchandani and L. Head. A real-time traffic signal control system: architecture, algorithms and analysis. *Transporation Research Part C*, 9:415–432, 2001.

[12] G.F. Newell. The rolling horizon scheme oftraffic control. *Transportation Research Part A*, 32:39–44, 1998.

[13] I. Porche, M. Sampath, R. Sengupta, Y.-L. Chen, and S. Lafortune. A decentralized scheme for real-time optimization of traffic signals. In *Proc. of the 1996 IEEE International Conference on Control Applications*, September 1996.

[14] I.R. Porche. *Dynamic Traffic Control: Decentralized and Coordinated Methods*. PhD thesis, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, Michigan, 1998.

[15] Isaac Porche and Stephane Lafortune. A game-theoretic approach to signal coordination. *Submitted to Transportation Research B*, 2005.

[16] S. Sen and K.L. Head. Controller optimization of phases at an intersection. *Transportation Science*, 3:5–17, 1997.

[17] Steven G. Shelby. Single intersection evaluation of real-time adaptive traffic signal control algorithms. In *Proc. of the 84rd Annual Meeting of the Tranportation Research Board (TRB'04)*, January 2004.

[18] R.T. van Katwijk, P. van Koningsbruggen, B. De Schutter, and J. Hellendoorn. Test bed for multiagent control systems in road traffic management. *Transportation Research Record*, 1910:108–115, 2005.