Movement-Based Look-Ahead Traffic-Adaptive Intersection Control ⋆

R.T. van Katwijk * **B. De Schutter** ** **J. Hellendoorn** ***

* *Netherlands Organization for Applied Scientific Research, TNO, Delft, The Netherlands (e-mail: ronald.vankatwijk@tno.nl).*
** *Delft Center For Systems and Control & Marine and Transport Technology department, Delft University of Technology, Delft, The Netherlands (e-mail: b@deschutter.info)*
*** *Delft Center For Systems and Control, Delft University of Technology, Delft, The Netherlands, (e-mail: j.hellendoorn@tudelft.nl)*

Abstract: There exist several control approaches for traffic signal control such as fixed-time, vehicle-actuated, or look-ahead traffic-adaptive control. We argue that in order to flexibly deal with varying demand levels movement-based control (which is already common in vehicle-actuated intersection control) is required instead of stage-based control (which is still employed in the state-of-the-art in look-ahead traffic-adaptive control). The movement-based approach is more flexible than the stage-based approach as it allows green for signals in different stages to start sooner if the demand for all conflicting movements in the current stage has cleared. Therefore, we propose a new *movement-based* method for look-ahead traffic-adaptive control. The method uses dynamic programming and branch-and-bound algorithms to determine the optimal traffic signal settings. We illustrate via a simulation example that the new approach can significantly outperform vehicle-actuated and stage-based look-ahead traffic-adaptive control.

Keywords: Traffic Control, Predictive control, Road Traffic, Dynamic Programming, Optimization-Based Control

1. INTRODUCTION

The state-of-the-art in traffic signal control is currently formed by traffic-adaptive systems. However, in practice, the majority of the controlled intersections in most countries are still controlled by vehicle-actuated controllers (e.g., in The Netherlands about 85 %). Vehicle-actuated controllers decide to either extend the active green phase or to switch to the next phase based on whether or not vehicles are still present on the approaches of the active green phase. A vehicle-actuated controller can therefore be considered to suffer from tunnel vision as it does not consider traffic on the other approaches.

Traffic-adaptive control differs from vehicle-actuated control because it can evaluate a set of feasible control decisions and make a decision that is optimal with respect to its control objective. A *look-ahead* traffic-adaptive controller additionally is capable of determining the optimal control decision on the basis of a longer-term analysis, which often incorporates information from further upstream. This allows the look-ahead traffic-adaptive controller to make better decisions in the long run. "Regular" traffic-adaptive control can be considered to be short-sighted compared to look-ahead traffic-adaptive control.

Current approaches to look-ahead traffic-adaptive control, such as PRODYN [Henry et al., 1983], OPAC [Gartner, 1983], UTOPIA-SPOT [Mauro and Di Taranto, 1989], RHODES [Head et al., 1992], and ALLONS-D [Porche et al., 1996], are hampered by the fact that they use a stage-based approach to traffic control as opposed to the movement-based approach employed by the state-of-the-art in vehicle-actuated control. The movement-based approach is more flexible than the stage-based approach as it allows green for signals in different stages to start sooner if the demand for all conflicting movements in the current stage has cleared.

In this paper a new algorithm for look-ahead traffic-adaptive control will be proposed. The algorithm employs a movement-based approach to traffic control, which allows for a significant speed-up of the optimization process since — compared to a stage-based approach — the movement-based approach allows the algorithm to explore a larger search space in the same amount of time. Furthermore, the algorithm integrates the currently best known dynamic programming optimization approach to look-ahead traffic-adaptive control [Sen and Head, 1997] with a branch-and-bound type optimization.

This paper is organized as follows. Section 2 describes the differences between the movement-based approach and the stage-based approach. Next, the new algorithm is proposed in Section 3. In Section 4 we illustrate via a simulation example that the new approach can significantly outper-

⋆ Research funded by the BSIK project "Transition to Sustainable Mobility (TRANSUMO)", the STW VIDI project "Multi-Agent Control of Large-Scale Hybrid Systems" (DWV.6188), the Transport Research Centre Delft, and the Delft Research Center Next Generation Infrastructures.

form vehicle-actuated control and stage-based look-ahead traffic-adaptive control.

2. MOVEMENT-BASED VERSUS STAGE-BASED TRAFFIC CONTROL

Consider the intersection depicted in Figure 1. A *movement* corresponds to a stream of vehicles that could get green or red [1], such as, e.g., movement 11 in Figure 1 which represents the vehicles on the upper arm of the intersection that can drive straight ahead or turn left. Given the set of movements for an intersection, a *stage* is then a (fixed) assignment of red or green indications to each of the movements over a period of time.

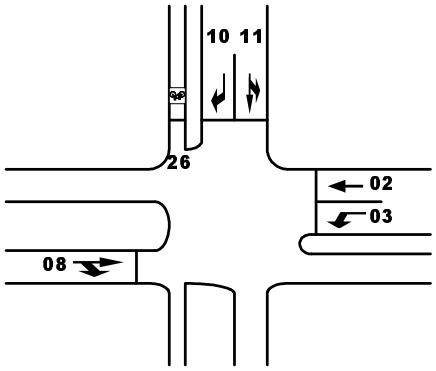


Fig. 1. Intersection with indication of some possible movements.

Although a look-ahead traffic-adaptive control algorithm is able to determine the stage-sequence on-line it is still important to provide the look-ahead traffic-adaptive control algorithm with a control structure (i.e., the composition and the sequence of the stages — see Figure 2 for an example) to work with as it is not possible to evaluate *all* possibilities on-line in real-time. The critical conflict group is the conflict group that, despite being sequenced efficiently, demands the longest cycle time. As the capacity of an intersection is determined by the critical conflict group, the choice of the control structure is often based on the critical conflict group only. Depending on the sequence of realization, the internal lost times will be longer or shorter. This enables the minimization of internal lost times by choosing a sequence of stages with minimum clearance times. Each structure has its own clearance times and so its own minimum cycle times.

As the complexity of an intersection's geometry increases so does the complexity of the control structure. Table 1 shows, e.g., the size of the search space to be evaluated to determine a truly optimal decision for different intersection geometries with increasing complexity for an N-seconds planning horizon. In the table a difference is made between "regular" stages (corresponding to a collection of non-conflicting movements that get green simultaneously) and "maximal" stages (corresponding to a maximal collection of non-conflicting movements that get green simultaneously). This difference is important to make, as the state-of-the-art in vehicle-actuated controllers uses a movement-based approach whereas look-ahead adaptive control still uses a stage-based approach. In order to gain the same level of flexibility with a stage-based approach as with a movement-based approach, non-maximal stages have to be incorporated, which greatly increases the search space.

[1] For the sake of simplicity of the exposition, we do not explicitly consider the yellow phase or the all-red phase, but for the time being we work with green and red only (see also Section 3.2 and [van Katwijk, 2008] for a more detailed explanation).

	Move-ments	Regular stages		Maximal stages	
Geometry	#	#	Search space	#	Search space
cars	12	111	111^N	17	17^N
+ pedestrians	20	2186	2186^N	112	112^N
+ bicycles	28	23362	23362^N	352	352^N
+ public transport	40	105722	105722^N	834	834^N

Table 1. Number of possible stages and the resulting search space for different intersection geometries.

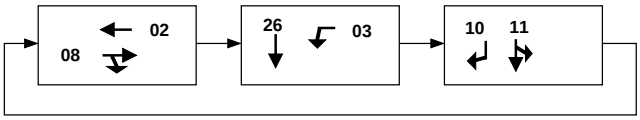


Fig. 2. Control structure for the intersection of Figure 1.

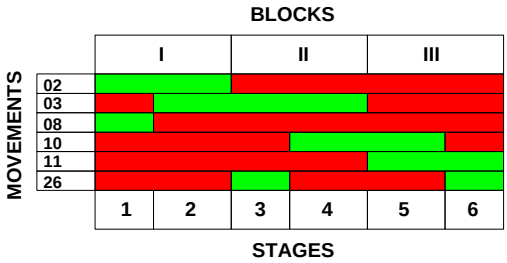


Fig. 3. Number of blocks versus the number of stages for the intersection of Figure 1.

We consider an approach based on blocks that consist of several movements. In contrast to the stage-based approach where a given movements gets green (or red) for the entire duration of the stage, we allow a movement to switch, e.g., from green to red within a block, at which time another non-conflicting movement can get green. This is illustrated in Figure 3. This figure shows a possible timing of green and red intervals for the movements given the structure of Figure 2. Time in the figure progresses from left to right across the page. The period of time in which a movement gets green or red is denoted by a bar that is colored accordingly. The interval of time in which a block is active is depicted at the top side of the picture. The figure shows that as soon as green for movement 08 of block I has terminated, movement 03 of block II is allowed to get green. This is allowed since movement 03 has no conflict with the movements of block I that still get green (in this case, this is movement 02). Similarly, movement 10 of block III is allowed to advance to green as soon as the green phase for movement 26 has ended. By adopting this block/movement-based approach (or movement-based approach for short) the size of the search space can be reduced significantly.

In Figure 3 the advantage is illustrated of employing a movement-based strategy over a stage-based strategy

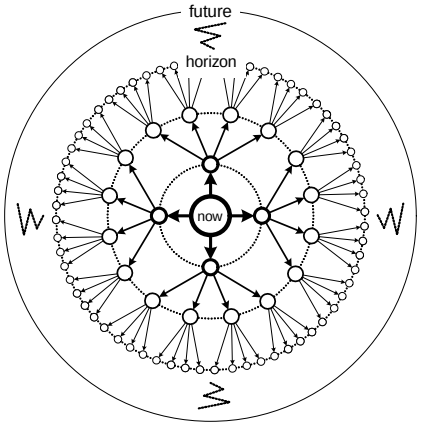


Fig. 4. Decision tree.

for the intersection of Figure 1. As the movement-based strategy allows movements to switch to green as soon as all conflicting movements have cleared, the number of green combinations possible with just three blocks would have required six distinct (non-maximal) stages if instead an equivalent stage-based approach had been applied. These six stages are depicted on the bottom of the picture. Optimizing over the horizon using blocks instead of stages significantly reduces the branching factor of the tree and thus significantly reduces the search space without making sacrifices with regard to the quality of the solution.

3. A MOVEMENT-BASED LOOK-AHEAD TRAFFIC-ADAPTIVE ALGORITHM

We use a moving-horizon approach in which at each step k_0 an optimal control sequence has to be determined over a given horizon $[k_0, k_0 + K)$. Control decisions at each step then involve whether or not to end or to initiate green for movements. This essentially results in an optimization problem involving a decision tree. In Section 3.1 we present the search algorithm we propose to use, and in Section 3.2 we then explain how one can determine when a movement can end, and which of the non-conflicting movements can subsequently start.

3.1 Search algorithm

As the decision space has a discrete structure, the search for the optimal sequence of decisions $u^* = u^*_{k_0}, \ldots, u^*_{k_0+K-1}$ corresponds to building a decision tree. To illustrate the approach consider the decision tree depicted in Figure 4. The tree is rooted by the current state k_0, labeled "now". Four branches lead away from this state, each corresponding to the result of a control decision u_0 that can be taken at this state. The control decision made results in a new state from which again four control decisions can be made. Each decision takes us further into the future and deeper into the tree formed by the states resulting from each possible decision.

Since the 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 not to 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 [Shelby, 2004] several well-known algorithms are assessed based on computational speed and on the quality of the results (in terms of vehicle delay). The search algorithm we present is based on a dynamic programming formulation similar to the one found in [Sen and Head, 1997]. However, the algorithm described below uses movements instead of stages, creating a much more efficient approach. To improve further upon the performance of the algorithm it is extended with techniques from the branch-and-bound algorithm. The algorithm is described below.

Dynamic programming [Dreyfus and Law, 1977, Bertsekas, 2005] is a method for solving problems exhibiting the properties of overlapping subproblems and optimal substructure. A problem is said to have overlapping subproblems if the problem can be broken down into subproblems that can be reused several times whereas a problem is said to have an optimal substructure if its optimal solution can be constructed efficiently from optimal solutions to its subproblems.

In our formulation the problem is to determine the optimal sequence and duration of the blocks over an optimization horizon. In our formulation each decision stage represents the total time allocated to a block. Each decision stage is divided into states. A state encompasses the information required to go from one decision stage to the next. In our formulation the state denotes the total time allocated to the blocks up to and including the current decision stage. Starting with an initial block, the algorithm treats each block as a decision stage, and optimizes over as many cycles as necessary to obtain an optimum.

The following notation is introduced. The cardinality of a set B will be denoted by $|B|$. We also define

- B : set of blocks: $B = \{B_1, B_2, \ldots, B_{|B|}\}$; individual blocks are indexed by i
- B_b : currently active block
- j : index for the decision stage of the dynamic program
- k : state variable containing the total number of allocated time steps
- K : length of the optimization horizon in discrete time steps
- u : control variable denoting the number of time steps allocated to the block
- $U_{j,k}$: set of feasible control decisions for decision stage j and state k
- $F_{j,k,u}$: performance of control u for decision stage j and state k
- $v_{j,k}$: cumulative performance for decision stage j and state k.

Let k_{j+1} denote the successor state of k_j following the implementation of control u_j. For the sake of simplicity we assume without loss of generality that $k_0 = 0$. The first decision stage in our formulation concerns the number of additional time steps to allocate for the block that is currently active. The first decision stage therefore decides about the time to allocate to block B_b, the second decision stage decides about the time to allocate to block $(b + 1) \mod |B|$, etc. The exact number of decision stages used is a by-product of the computations. Given a value for the

state variable k, the control variable u can assume values from the discrete set $U_{j,k} = \{0, 1, \ldots, k\}$.

By allowing u to assume a value of 0, blocks can be skipped and any desired block sequence can thus be generated. The previous state k_{j-1} can be determined on the basis of the current state k_j and chosen control decision u_j as follows:

$$k_{j-1} = k_j - u_j \tag{1}$$

The forward recursion of the algorithm is now presented. In the following k_{j-1} is calculated as a function of k_j and u_j via (1). The algorithm starts with decision stage $j = 1$, and proceeds recursively to $j = 2, 3, \ldots$ At each decision stage, the method calculates the best control decision $u^*_{j,k}$ for each possible value of the state variable k. The performance function (to be minimized) is assumed to take the form $F_{1,k_1,u_1} + F_{2,k_2,u_2} + \ldots$

The optimal performance, denoted by $v_{j,k}$ is a function of the immediate performance $F_{j,k,u}$ of implementing the optimal control decision u^*:

$$v_{j,k} = \min_{u \in U_{j,k}} \{F_{j,k,u} + v_{j-1,k-u}\}$$

The corresponding forward recursion algorithm is provided in Algorithm 1. The recursion ends if there is nothing to be gained from evaluating a new decision stage and at that time the optimal solution can be retrieved. This is the case if all of the previous $|B|$ stages have not improved the performance. Since a later decision stage allows more stage changes for the same value of the state variable, it follows that $v_{j-1,K} \geq v_{j,K}$. Furthermore, note that if $v_{j-1,K} = v_{j,K}$, it follows that there is nothing to be gained by allowing the specific block change associated with decision stage j. This reasoning is applied to the $|B| - 1$ stages preceding decision stage j.

Algorithm 1 Forward recursion

initialize $v_{0,\cdot} \leftarrow 0$, $k \leftarrow 1$
for $k = 0, \ldots, K$ **do**
  $v_{j,k} \leftarrow \min_{u \in U_{j,k}} \{F_{j,k,u} + v_{j-1,k-u}\}$
  record $u^*_{j,k}$, an optimal solution to the above problem
end for
if $(j < |B|)$ **then**
  $j \leftarrow j + 1$, and repeat from Step 2
else {check whether done}
  for $i = 1, \ldots, |B| - 1$ **do**
    if $v_{j-i,K} \neq v_{j,K}$ **then**
      $j \leftarrow j + 1$, break for-loop and repeat from Step 2
    end if
  end for
end if

The optimal solution can subsequently be retrieved by determining the optimal trajectory of states and the associated optimal control decisions. Let J denote the last decision stage for which the value function has been calculated in the forward recursion. Then, we may retrieve an optimal policy by tracing back through the table that has recorded the optimal control decisions, $u^*_{j,k}$ for $j = J - (|B| - 1), J - (|B| - 2), \ldots, 1$. Note that since the forward recursion ends only if $v_{j,K} = v_{j-1,K}$ for $j = J - |B|, \ldots, J$, the controls satisfy $u^*_{j,K} = 0$ for these j, and consequently, it is sufficient to retrieve control decisions starting with decision stage $j = J - (|B| - 1)$.

The formulation of the algorithm thus far relies purely on dynamic programming. To further improve upon the efficiency elementary mechanisms of the branch-and-bound algorithm are also included. The key component of the proposed branch-and-bound algorithm consists in determining when a movement can end, and which of the non-conflicting movements of the next block can be started. Next we explain how this can determined mathematically.

3.2 Scheduling of the movements

The following notation is introduced:

S : set of signals; individual signals are indexed by s
λ_s^{start} : green start lag for signal s
λ_s^{end} : green end lag for signal s
y_s : yellow time for signal s
$r_{r,s}$: all-red time needed to safely switch from signal r to signal s
$\mathbf{q}_s^{\text{in}}$: ordered multi-set of estimated arrival times for signal s
$\mathbf{q}_s^{\text{out}}$: ordered multi-set of estimated departure times for signal s
$\widehat{\mathbf{q}}_{s,j,k}^{\text{in}}$: ordered multi-set of estimated arrival times of arrivals that are queued for signal s for decision stage j and state k
$\widehat{\mathbf{q}}_{s,j,k}^{\text{out}}$: ordered multi-set of estimated departure times for departures of signal s for decision stage j and state k
$\widehat{\mathbf{q}}_{s,j,k,|\mathbf{q}_s^{\text{out}}|}^{\text{out}}$: estimated departure time of the last vehicle $\left|\widehat{\mathbf{q}}_{s,j,k}^{\text{out}}\right|$ departing from signal s
$A_{j,k,u,s}$: demand for signal s in the control interval determined by k and u
$A^1_{j,k,u,s}$: estimated arrival time of the first vehicle arriving at signal s for state k and control decision u
$R_{j,k,u,s,m}$: time after which the m^{th} vehicle will be able to depart at signal s for state k and control decision u
$Q_{j,k,u,s}^{\text{in}}$: (queued) arrivals remaining at signal s after having implemented control decision u to reach state k
$Q_{j,k,u,s}^{\text{out}}$: departures from s after having implemented control u to reach state k
$g_{j,k,s}^{\text{start}}$: time green can start for signal s for state k and control decision u
$g_{j,k,s}^{\text{end}}$: time green can end for signal s.

It will be assumed that the following information is available when calculating the first decision stage: the estimated arrival times for each vehicle approaching a signal ($\mathbf{q}_s^{\text{in}}$) for all $s \in S$, the time of the last vehicle served by a signal ($\widehat{\mathbf{q}}_{s,0,0,|\mathbf{q}_s^{\text{out}}|}^{\text{out}}$) for all $s \in B_b$, and the time green started for a signal ($g_{0,0,s}^{\text{start}}$) for all $s \in B_b$.

The control decision u^* that results in the best performance determines the information that is retained in the calculation of the subsequent stages. The demand for a signal s is determined on the basis of any arrivals remain-

ing ($\widehat{\mathbf{q}}^{\text{in}}_{s,j-1,k-u}$) after having implemented the optimal control decision of the previous decision stage $j-1$, and the arrivals during the interval $[k-u,k)$ specified by the control decision (u):

$$A_{j,k,u,s} = \widehat{\mathbf{q}}^{\text{in}}_{s,j-1,k-u} \cup \left\{a | (a \in \mathbf{q}^{\text{in}}_s) \wedge (a \geq k-u) \wedge (a < k)\right\},$$

with $\widehat{\mathbf{q}}^{\text{in}}_{s,0,0}$ determined as $\widehat{\mathbf{q}}^{\text{in}}_{s,0,0} = \left\{a | (a \in \mathbf{q}^{\text{in}}_s) \wedge (a < 0)\right\}$.

The state of a signal after the application of a control decision is determined by this demand ($A_{j,k,u,s}$) and the time at which a vehicle is able to depart ($R_{j,k,u,s,m}$):

$$Q^{\text{in}}_{j,k,u,s} = \left\{a_m | a_m \in A_{j,k,u,s} \wedge R_{j,k,u,s,m} < k\right\}.$$

The departures for a signal are determined on the basis of the vehicles that have already left ($\widehat{\mathbf{q}}^{\text{out}}_{s,j-1,k-u}$) and the vehicles that depart during the control decision:

$$Q^{\text{out}}_{j,k,u,s} = \widehat{\mathbf{q}}^{\text{out}}_{s,j-1,k-u} \cup \left\{a_m | a_m \in A_{j,k,u,s} \wedge R_{j,k,u,s,m} \geq k\right\}.$$

The time at which a vehicle is able to depart depends on the saturation flow (q^{sat}_s) of the signal, and on the time when the green phase effectively starts ($g^{\text{start}}_{j,k,s} + \lambda^{\text{start}}_s$) *or* on the time of the last departure for the signal ($\widehat{\mathbf{q}}^{\text{out}}_{s,j,k,|\mathbf{q}^{\text{out}}_s|}$) if the green phase for the signal is continued. It is determined as follows: $R_{j,k,u,s,m} =$

$$\sum_{1}^{m-1} \frac{1}{q^{\text{sat}}_s} + \begin{cases} g^{\text{start}}_{j,k,s} + \lambda^{\text{start}}_s & \text{if } s \notin B_{(b+j-2) \mod |B|} \\ \widehat{\mathbf{q}}^{\text{out}}_{s,j,k,|\mathbf{q}^{\text{out}}_s|} + \frac{1}{q^{\text{sat}}_s} & \text{if } s \in B_{(b+j-2) \mod |B|} \end{cases}$$

The saturation flow q^{sat}_s dictates the minimum inter-departure time between consecutive vehicle departures. The time of the most recent departure from a signal $\widehat{\mathbf{q}}^{\text{out}}_{s,j,k,|\mathbf{q}^{\text{out}}_s|}$ ensures that the minimum inter-departure time is respected for a signal that is green in two consecutive blocks.

To calculate how the state of the intersection is affected after the application of a chosen control decision it is important to know when the green signal for a movement starts. Green starts for signal s if it has demand ($|A_{j,k,u,s}| \neq 0$) and if it is active in the block ($s \in B_{(b+j-1) \mod |B|}$). In order not to waste the available green time, green starts no sooner than necessary to allow the first arrival ($A^1_{j,k,u,s}$) to pass without delay. Of course, the green signal is allowed to start only after any conflicting movements have cleared. The algorithm used to determine the time green starts is given in Algorithm 2.

The time green can start for a movement depends on the time that conflicting movements have cleared. Note that green can start before the start time of the block it is part of if the conflicting movements have cleared before that time. The time conflicting signals have cleared depends on the time the green phase for the conflicting movement has ended.

The time green can end for a signal is determined after having evaluated all $u \in U_{j,k}$. It is determined for the best performing control decision using Algorithm 3. The time green ends for a signal depends on: whether it is active in the block ($s \in B$), whether it served any vehicles ($\widehat{\mathbf{q}}^{\text{out}}_{s,j,k,|\mathbf{q}^{\text{out}}_s|} \geq g^{\text{start}}_{j,k,s}$), the time it has started ($g^{\text{start}}_{j,k,s}$), the minimum green time ($g^{\min}_s + y_s$), and the time the last vehicle was served ($\widehat{\mathbf{q}}^{\text{out}}_{s,j,k,|\mathbf{q}^{\text{out}}_s|}$).

Algorithm 2 Compute $g^{\text{start}}_{j,k,s}$, the time green can start for signal s

```
if |A_{j,k,u,s}| ≠ 0 then {there is demand}
    g^start_{j,k,s} ← A^1_{j,k,u,s} {green starts no sooner than necessary}
    for all r ∈ B_{(b+j-2) mod |B|} do {signals that are green in the previous block}
        g^start_{j,k,s} ← max{g^start_{j,k,s}, g^end_{j-1,k-u,r} + r_{r,s}} {green starts no sooner than conflicting signals have cleared}
    end for
else {there is no demand}
    g^start_{j,k,s} ← k {as there is no demand, green is skipped}
end if
return g^start_{j,k,s}
```

Algorithm 3 is used to determine the time green ends.

Algorithm 3 Compute $g^{\text{end}}_{j,k,s}$, the time green can end for signal s

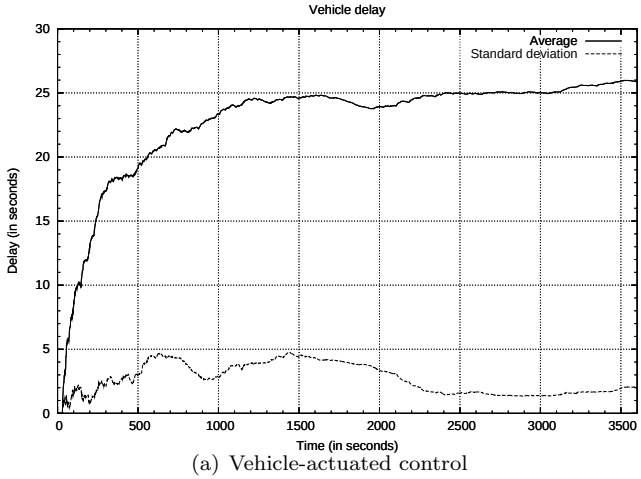
```
Require: The departures (q̂^out_{s,j,k}) that result after having implemented the optimal control decision have been determined
if q̂^out_{s,j,k,|q^out_s|} ≥ g^start_{j,k,s} then {demand has been served}
    g^end_{j,k,s} ← max{g^start_{j,k,s} + g^min_s + y_s, q̂^out_{s,j,k,|q^out_s|}} {determine end of green phase}
else {no demand has been served}
    g^end_{j,k,s} ← k − u {as no demand has been served, the green phase was skipped}
end if
return g^end_{j,k,s}
```

While optimizing the performance of an intersection certain constraints should be respected. The algorithm ensures that constraints are respected with respect to, e.g., minimum and maximum green and red times, protection of dilemma, option, and comfort zones, prior commitments, block skipping and termination in the presence of demand, and maximum allowable queue lengths.

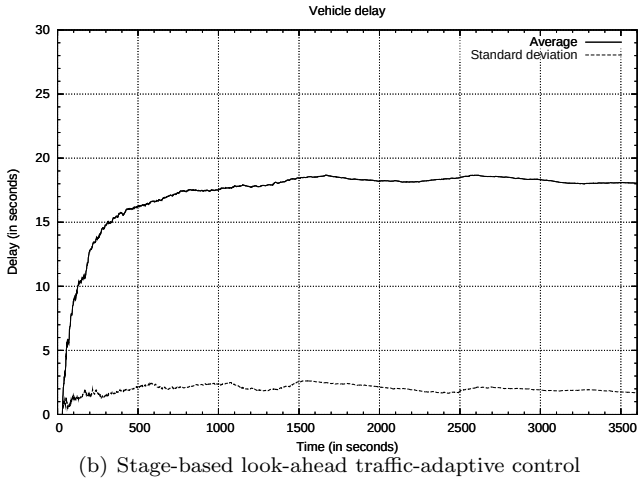
4. EXAMPLE

In order to test the performance of the developed algorithm the traffic management test bed described in [van Katwijk et al., 2005] is used. This test bed enables us to interface the adaptive control algorithm of Section 3 with the microscopic traffic simulation models Paramics and AIMSUN. The simulations were performed for a 4-arm intersection. Each of the twelve possible movements on the intersection has a separate, single, approach lane. The total demand for the intersection is set to 4400 vehicles per hour, which is distributed over the movements in proportion to the saturation flow rate of each movement. Maximum green times for each stage and block were subsequently determined using Webster's method [Webster and Cobbe, 1966].

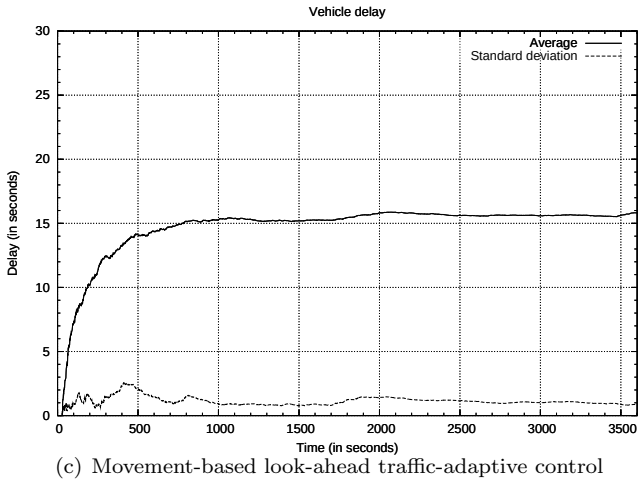
The results have been obtained from a number of one-hour simulations each with a different random seed and are displayed in Figure 5. The plots show the average delay per



(a) Vehicle-actuated control



(b) Stage-based look-ahead traffic-adaptive control



(c) Movement-based look-ahead traffic-adaptive control

Fig. 5. Performance for an intersection with 4 blocks.

vehicle as it evolves over a one-hour period when the intersection is operating under (a) vehicle-actuated control, (b) *stage-based* look-ahead traffic-adaptive control, and (c) *movement-based* look-ahead traffic-adaptive control.

The average delay encountered by a vehicle in the vehicle-actuated controlled case stabilizes at about 25 s per vehicle (see Figure 5(a)) whereas the average delay encountered by a vehicle in the stage-based traffic-adaptive controlled case stabilizes at about 18 s per vehicle (see Figure 5(b)). This proves that planning for future arrivals can substantially improve the performance of an intersection. The *movement-based* look-ahead traffic-adaptive controller results in an even lower average delay of about 16 s per vehicle (see Figure 5(c)). Note that this reduction in delay can be obtained without significantly increasing the number of computations as the movement-based look-ahead traffic-adaptive control algorithm developed in this paper allows to evaluate a larger number of possible signal timings without increasing the size of the search space.

5. CONCLUSIONS

A new algorithm has been presented for look-ahead traffic-adaptive intersection control. The algorithm integrates the currently best known dynamic programming optimization approach to look-ahead traffic-adaptive control [Sen and Head, 1997] with a branch-and-bound type optimization, and it applies the more flexible movement-based approach to traffic signal control as opposed to the stage-based approach employed by the current state of the art in look-ahead adaptive control. This enables the algorithm to analyze a larger number of possible signal timings without further increasing the size of the search space.

REFERENCES

D. P. Bertsekas. *Dynamic Programming and Optimal Control – Volume I.* Athena Scientific, 3nd edition, 2005.

S.E. Dreyfus and A.M. Law. *Art and Theory of Dynamic Programming.* Academic Press, Orlando, FL, USA, 1977.

N.H. Gartner. OPAC: A demand-responsive strategy for traffic signal control. *Transportation Research Record*, 906:75–81, 1983.

K.L. Head, P.B. Mirchandani, and D. Sheppard. Hierarchical framework for real-time traffic control. *Transportation Research Record*, 1360:82–88, 1992.

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

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

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

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

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

R. T. van Katwijk. *Multi-Agent Look-Ahead Traffic Adaptive Control.* PhD thesis, Delft University of Technology, Delft, The Netherlands, January 2008.

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.

F.V. Webster and B.M. Cobbe. Traffic signals. *Road Research*, 56, 1966.