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

Haijema, R.

Publication date
2008

Appendix D

Optimization of fixed cycle

D.1 Determining the minimal cycle length

Under FC the green periods are fixed to say $\alpha_1, \ldots, \alpha_C$, where C is the number of combinations. The switch-over times are fixed to γ slots. During the first β slots the lights show yellow and cars still leave the queue (whenever present). Departures happen at rate 1 per slot per queue (that has a green or yellow signal).

A bottleneck stream of a combination is a queue that brings the highest workload over all streams in that combination. Further let λ_c denote the arrival intensity at the bottleneck stream of combination c.

The length of a cycle D is fixed given the sum of the green periods and the switch-over times. The what-we-call Minimal-cycle-length algorithm provides the minimal cycle length D and minimal green times $\alpha_1, \ldots, \alpha_C$, under which FC is just stable.

Minimal-cycle-length algorithm

1. $D' \leftarrow C \cdot (1+\gamma)$; $\quad \{$*suppose a minimum green time of 1 slot*$\}$
2. repeat
3. $\quad D \leftarrow D', \quad D' \leftarrow C \cdot \gamma$;
4. $\quad \forall c \in \{1, \ldots, C\}$:

   $\quad\quad \alpha_c \leftarrow 1 + (\lceil \lambda_c D - (1+\beta) \rceil)^+, \quad \{$*minimal green time given* $D\}$

   $\quad\quad D' \leftarrow D' + \alpha_c$;
5. until $D' = D$;
6. $\rightarrow$ minimum cycle length is D slots, with green times $\alpha = \alpha_1, \ldots, \alpha_C$.

D.2 Incremental search

Starting at the shortest cycle possible the algorithm, which we call the Optimal-fixed-cycle algorithm, increases the cycle length successively by 1 slot. Every time that a better cycle is found the best one found so far is update. Especially in the beginning of the search, an increase by one may result in a very bad or even instable cycle. Therefore we continue increasing the cycle length until all green times are increased proportionally to the workload the respective flows bring to the intersection. Hence after each improvement M successive increments are considered, with $M = \sum_c \lambda_c / \lambda_g$ and $g = \arg\min_c \lambda_c$, the 'less- busiest' combination. If an improvement stays out the search is terminated.

We formalize the algorithm for finding a nearly optimal FC in pseudo-code using the notations from the previous section. We introduce $EW(D, \alpha)$ being the expected overall average waiting time under FC with cycle length D and green times $\alpha = \alpha_1, \ldots, \alpha_C$. The function EW can be evaluated by simulation or by solving a Markov chain. The vector $\mathbf{e}_c$ is the c-th unit vector with all elements set to 0, except for a 1 at the c-th position; u, m, b and g are auxiliary variables.

Optimal-fixed-cycle algorithm

1. Set cycle length D and green times α by the Minimal-cycle-length algorithm, such that FC is just stable.
2. $D^* \leftarrow D, \quad \alpha^* \leftarrow \alpha, \quad n \leftarrow 0; \qquad \{n = \#\ \textit{iterations no improvement}\}$
3. $g \leftarrow \arg\min_c \lambda_c,$

   $M \leftarrow \sum_c \lambda_c/\lambda_g; \qquad \{M = \textit{maximal}\ \#\ \textit{iterations no improvement}\}$
4. repeat
5. $\quad w \leftarrow EW(D, \alpha), \quad n \leftarrow n + 1;$
6. $\quad \forall c \in \{1, \ldots, C\} : \qquad \{\textit{select green period}\}$

   $\quad$ if $EW(D + 1, \alpha + \mathbf{e}_c) \leq w$

   $\quad\quad$ then $\quad w \leftarrow EW(D + 1, \alpha + \mathbf{e}_c), \quad b \leftarrow c;$
7. $\quad D \leftarrow D + 1, \quad \alpha \leftarrow \alpha + \mathbf{e}_b; \qquad \{\textit{lengthen green period combination } b\}$
8. $\quad$ if $EW(D^*, \alpha^*) > EW(D, \alpha) \qquad \{\textit{an improvement}\}$

   $\quad\quad$ then $\quad D^* \leftarrow D, \quad \alpha^* \leftarrow \alpha, \quad n \leftarrow 0;$
9. until $n = M$;
10. $\rightarrow$ best cycle length is D^* slots, with green times $\alpha_1^*, \ldots, \alpha_C^*$.