Title
A Paramics Plugin for Actuated Signal Control and First Generation UTCS

Authors
Gomes, Gabriel
Skabardonis, Alexander

Publication Date
2006-07-01

CALIFORNIA PATH PROGRAM
INSTITUTE OF TRANSPORTATION STUDIES
UNIVERSITY OF CALIFORNIA, BERKELEY

A Paramics Plugin for Actuated Signal Control and First Generation UTCS

Gabriel Gomes
Alexander Skabardonis

California PATH Working Paper
UCB-ITS-PWP-2006-8

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation, and the United States Department Transportation, Federal Highway Administration.

Report for Task Order 5322

July 2006

ISSN 1055-1417

A Paramics Plugin for Actuated Signal Control and First Generation UTCS

Gabriel Gomes
gomes@path.berkeley.edu

Alexander Skabardonis
dromeas@berkeley.edu

University of California at Berkeley,
Institute of Transportation Studies,
California Partners for Advanced
Transit and Highways (PATH)

July 2006

A Paramics Plugin for Actuated Signal Control and First Generation UTCS

Gabriel Gomes and Alexander Skabardonis

Abstract

This report serves as a user manual for a plugin developed under the Paramics API for simulating standard surface street traffic controllers. The strategies included are time-of-day, actuated signal control, traffic responsive, and traffic responsive with critical intersection control.

Keywords : signal control, traffic simulation, Paramics.

1 Introduction

This document describes a plugin developed in Paramics API V.5 for simulating the first generation Urban Traffic Control System (UTCS) and standard actuated signal control. The UTCS is a set of algorithms developed under a program of the FHWA beginning in 1967 (Honeywell 1979, Kay 1975, Henry 1976). The objective of this program was to create and test a host of microprocessor-based urban signalling methods. The control strategies that resulted were classified into three generations, the first generation consisting of methods for selecting an appropriate timing plan from a stored library. The second and third generations automatically generate new timing plans based on measured data.

The strategies included in the first generation UTCS are time-of-day, traffic responsive, and critical intersection control. These were first tested in downtown Washington D.C. between 1972 and 1974. They have since become standard features in many NEMA and

170 type controllers. Most urban intersections in the U.S. use either time-of-day or actuated signal control, many others use traffic responsive and even adaptive strategies. However, there is no simple recipe for choosing the best approach for a given site. Traffic signal timing remains somewhat of an art.

Microsimulation modeling has recently emerged as an excellent tool for selecting a control strategy and tuning its parameters prior to deployment. While some microsimulation softwares such as CORSIM include actuated signal control as a basic feature, Paramics includes only pre-timed control. A plugin for actuated control was developed previously at the University of California at Irvine (Chu 2004). However, limitations in the older version of the Paramics API precluded some important features. The plugin described here includes an updated implementation of actuated signal control, as well as the three strategies of the first generation UTCS.

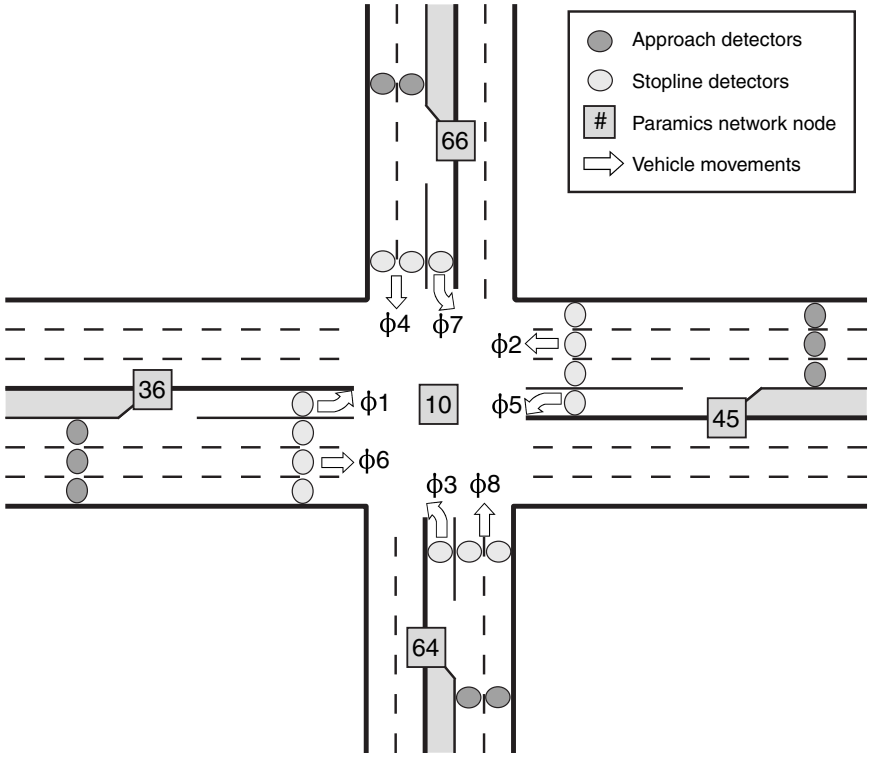

Figure 1: Layout of a typical intersection

2 Basic setup

The assumed layout for each signalized intersection is shown in Figure 1. Three-legged intersections are also supported. The eight through and left-turn phases are numbered according to the NEMA convention. Each of these can be equipped with a set of *approach* detectors, and/or a set of *stopline* detectors. It is assumed that the single-lane loop detector model of Paramics is used.

The function of the approach detector in actuated control is to renew the green light extension and to count vehicles for calculating the initial green time. Approach detection is used in traffic responsive control to generate the traffic signature, while in critical intersection control it is used to compute green demands. These detectors are typically located about 200 feet from the stopline.

Stopline detectors are usually placed within a few feet of the stopline, and are used in actuated signal control to place calls for service on particular phases. Related to the placement of stopline detectors, we have noticed a behavior in Paramics that can cause the plugin to fail. Vehicles wanting to turn left at an intersection which are unable to change lanes into the dedicated left-turn lane before reaching the stopline, will often change lanes from a standstill and move laterally onto the stopline detector. These vehicles, because they do not trigger the Paramics detector in the normal way, do not place a call for service. They also prevent other vehicles from going over the detector. The left turn phase is therefore always skipped, and an unboundedly long queue can form. To avoid this problem, we recommend placing the stopline detectors a bit farther upstream.

The main input file for the plugin is `param_main.txt`. This file provides geometric, output, and high level control information for the simulation. An example input file is provided in Figure 2. Each line in the file begins with a case-sensitive token, followed by information separated by blank spaces or tab characters. Comments can be inserted using the percentage symbol (`%`).

```
controller    ASC
outputperiod  60
timeconstant  30
%-------------------------------------------------------
node 10                                 % Walnut St.
phase2nodes 45 36
phase4nodes 66 64

% phase#    1  2  3  4  5  6  7  8
protected   1  1  1  1  1  1  1  1
permissive  0  0  1  0  0  0  1  0

det 1 S n10_1_S_1
det 2 A n10_2_A_1 n10_2_A_2 n10_2_A_3
det 2 S n10_2_S_1 n10_2_S_2 n10_2_S_3
det 4 A n10_4_A_1
det 5 S n10_5_S_1
det 6 S n10_6_S_1 n10_6_S_2 n10_6_S_3
det 6 A n10_6_A_1 n10_6_A_2 n10_6_A_3
det 8 A n10_8_A_1

%-------------------------------------------------------
node 11                                 % Eschelman St.
phase2nodes 46 48
phase4nodes 50 47

% phase#    1  2  3  4  5  6  7  8
protected   1  1  0  1  1  1  0  1
permissive  0  0  0  0  0  0  0  0

det 1 S n11_1_S_1
det 2 A n11_2_A_1 n11_2_A_2 n11_2_A_3
det 2 S n11_2_S_1 n11_2_S_2 n11_2_S_3
det 3 S n11_3_S_1
det 4 S n11_4_S_1
det 5 S n11_5_S_1
det 6 P n11_6_S_1 n11_6_S_2 n11_6_S_3
det 6 A n11_6_A_1 n11_6_A_2 n11_6_A_3
det 7 S n11_7_S_1
det 8 P n11_8_S_1
```

Figure 2: Main input file (`param_main.txt`)

The first line in the file determines the control strategy. The accepted values for the `controller` token are:

`TOD`	... time-of-day
`ASC`	... actuated signal control
`TRSP`	... traffic responsive control
`TRSPCIC`	... traffic responsive with critical intersection control.

The `outputperiod` token defines the aggregation period for the loop detector output file in seconds. This value does not affect the controller behavior in any way. If omitted, the output file will not be generated.

Traffic responsive and critical intersection control use *smoothed* measurements, as opposed to raw detector measurements. The smoothed data is generated with a first-order filter:

$$\begin{aligned} \text{vol}^s[k] &= (\tau)\,\text{vol}^s[k-1] + (1-\tau)\,\text{vol}^r[k] \\ \text{occ}^s[k] &= (\tau)\,\text{occ}^s[k-1] + (1-\tau)\,\text{occ}^r[k] \end{aligned}$$

where the superscript s and r indicate smoothed and raw variables. Raw measurements are 1-minute average values. τ is a filter parameter related to the time constant T_c (defined as the time to reduce the filter error by 63%) by $\tau = e^{-\Delta t/T_c}$, where Δt is the simulation step size. The time constant T_c is defined by the user in seconds with the `timeconstant` token.

The rest of the file describes the signalized intersections. Each intersection begins with the `node` token, followed by the ID number for the Paramics network node. This number is the "node name" in Paramics Modeller. Although Paramics allows character strings for its node name, this plugin requires a positive integer-valued node name. Also, the plugin overrides all signal timing and priority settings defined in Paramics.

The `phase2nodes` and `phase4nodes` tokens are used to orient the NEMA numbering scheme with respect to the intersection. The two numbers following `phase2nodes` and `phase4nodes` are respectively the *from* and *to* nodes for phases $\phi 2$ and $\phi 4$. To illustrate, the nodes from the sample input file are shown in Figure 1. For three-legged intersections, the *to* node for either `phase2nodes` or `phase4nodes` should be set to -1.

The `protected` and `permissive` tokens define the basic signalling features of the intersection. They are followed by a sequence of eight 0/1 values corresponding to the eight vehicle phases. Protected phases are those that are directly controlled by the traffic signal. Permissive phases are left turns that are allowed to proceed without a green light when the opposing through phase has the right-of-way. For example, setting $\phi 1$ to permissive means that vehicles in that phase are allowed to turn left during the green interval of $\phi 2$. Only left-turn phases can be made permissive.

Loop detectors are assigned to phases using the `det` token. The syntax for this line is:

```
det [phase number 1-8] [detector type S|A] [list of detector names]
```

The detector names are those defined in Paramics Modeller. Phases are not required to have stopline or approach detectors. Furthermore, a single detector may be associated with more than one phase.

Column #	Value
1	Start time for the aggregation period
2	Node ID
3	phase ID
4	1=Approach ; 0=Stopline
5	Lane
6	Vehicle count [veh]
7	Occupancy $\in$[0,1]

Table 1: Column headers for `output_loop.txt`

2.1 Output files

Aggregated raw measurements from each of the loop detectors listed in `param_main.txt` are exported to `output_loop.txt`. This file contains a matrix with 7 columns and a row for every detector and aggregation interval. The column headers are given in Table 1.

The program also produces a log file called `output_log.txt`, which collects error and warning messages generated during the simulation. Cycle lengths and green times are ex-

Column #	Value
1	Cycle start time
2	Intersection node number
3	Cycle length
4-11	Green time for phases 1 through 8

Table 2: Column headers for `output_control.txt`

ported to `output_controldata.txt`. The column headers for this file are given in Table 2.

3 Signal Timing Plans and Time-of-day control (TOD)

Under TOD control, every intersection operates according to a predefined plan with fixed cycle lengths, splits, and phase sequences. Paramics offers an interface for creating preset plans, however we provide another here which is used by the traffic responsive controller described in section 5. The input file for TOD control is `param_tod.txt`. A sample file is shown in Figure 3. A set of plans is defined, each identified by an integer following the `plan` token. The `todstart` and `todplan` tokens are used to set the times for switching between plans. For example, the sample input file will tell the controller to switch from plan 1 to plan 2 at $t = 300$ seconds, and back to plan 1 at $t = 400$ seconds.

Actual control devices do not change plans instantaneously. This would result in truncated green and red intervals, which would be dangerous for drivers and pedestrians. Instead, they transition in a gradual manner, with small adjustments to each consecutive cycle. Different controller manufacturers employ different transitions procedures. This plugin does not attempt to reproduce those procedures, but instead inserts a delay between the decision to change plans and the realization of the change. A nominal delay value is defined by the user with `transdelay`. This nominal value is rounded up to an integer multiple of the current cycle length. In the example, plan 1 will actually be in effect from $t = 0$ to $t = 390 = 300 +$ one plan 1 cycle. Plan 2 will then be activated from $t = 390$ to $t = 520 = 400 +$ two plan 2 cycles.

Each network plan consists of a number of intersection-specific plans, which contain stage sequences and corresponding green, yellow, and red clearance times. It is left to the

user to ensure that the phases combined in each stage are compatible. The start time for the intersection plan is shifted with respect to the master clock by `offset` seconds. The master clock rotates with cycle length `cyclelength` in seconds. As in `param_main.txt`, each signalized intersection is identified by `node`. The sequence of stages is set with the `stage` token. The syntax for this line is:

```
stage [phase ID] [phase ID] [green sec] [yellow sec] [red clear sec]
```

In the case that a stage involves only one phase, the second phase ID should be set to zero. The sum of the green, yellow, and red clearance times for each intersection must not exceed the cycle length. All signals are set to red during the time remaining after the end of the last stage. Actual traffic controllers usually give this time to the major through movements. To mimic this behavior, it is recommended that an additional stage be added to fill in any leftover time.

```
todstart      0   300   400
todplan       1     2     1
transdelay   80
%================================================
plan 1
cyclelength 90
% ..............................................
node 10
offset 0
%       phaseA   phaseB   green   yellow   red clear
stage      1        5       15       3         3
stage      2        6       25       4         3
stage      3        7       15       3         3
stage      4        8       17       4         3
% ..............................................
node 11
offset 0
%       phaseA   phaseB   green   yellow   red clear
stage      1        5       22       4         3
stage      2        6       30       4         3
stage      4        8       25       4         3
%================================================
plan 2
cyclelength 60
% ..............................................
node 10
offset 0
%       phaseA   phaseB   green   yellow   red clear
stage      1        5       10       3         3
stage      2        6       27       4         3
stage      3        7       10       3         3
stage      4        8       25       4         3
% ..............................................
node 11
offset 0
%       phaseA   phaseB   green   yellow   red clear
stage      1        5       17       4         3
stage      2        6       25       4         3
stage      4        8       35       4         3
```

Figure 3: Input file for time-of-day control (`param_tod.txt`)

4 Actuated signal control (ASC)

The plugin implements a fully actuated eight-phase dual-ring traffic controller. Under actuated control each intersection operates locally, with no communication to adjacent intersections. The controller architecture has two levels: an upper level represented by a *dual-ring controller* (section 4.1) and a lower level that executes the individual phases (section 4.2).

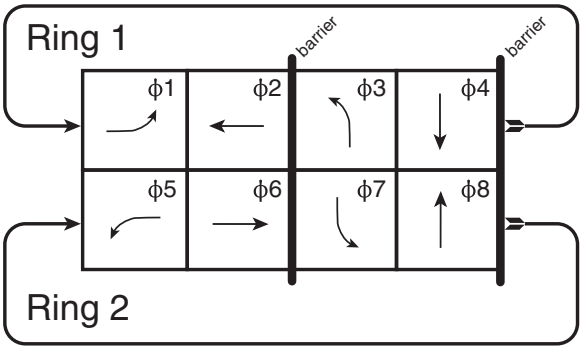

Figure 4: Dual-ring controller

4.1 The dual-ring controller

The objective of the dual-ring controller is to maintain safe conditions by allowing only compatible vehicle movements to enter the intersection at any time. Compatible movements or phases are those that 1) belong to different rings, and 2) are on the same side of the barriers (see Figure 4). Phases on opposite sides of a barrier are in conflict, and can therefore not be combined. For example, the only compatible options for $\phi 2$ in Figure 4 are $\phi 5$ and $\phi 6$.

The controller advances by initiating phases and, upon termination, selecting the next one to execute. The upcoming phase is found by searching the ring for the next protected phase which has either registered a vehicle presence or has been designated as a *recall* phase. Unprotected phases are always skipped. A recall phase cannot be skipped, even if no vehicle is registered by its stopline detectors. Recalls are typically applied to the through phases of major streets. They are also often used on cross streets lacking stopline detection. If no vehicle is registered on a non-recall phase, that phase may be skipped.

Left turns can be either *leading* or *lagging*, depending on their position in the ring with respect to the opposing through phase. All of the left-turn phases in Figure 4 are leading. Phase $\phi 1$ is made a lagging left turn by swapping its position with $\phi 2$.

All of these features are included in the plugin. The main task of updating the active phases is implemented in two steps. First, whenever one of the two active phases terminates, the next serviceable phase in the ring is found with the NextPhase() function. This function is passed an active or inactive phase, and returns the next protected phase with a vehicle call or a recall status. It returns the input phase if it is still active. It also returns the input phase if none of the other three phases in its ring require service. The pair of phases found with NextPhase() may not be compatible. The second step is to find the number of barriers crossed in each ring in the transition from the current phase to the NextPhase(). This number – 0, 1, or 2 – may not be the same for both rings. The following logic is then applied to adjust the selected pair of phases so that they remain compatible:

```
if( both rings jump the same number of barriers )
     - transition to NextPhase() on both rings
else
     if( one ring jumps zero barriers )
          - that ring transitions to NextPhase()
          - the other remains in its current phase
     else      /* the only remaining case is one jumps one barrier, the other two */
          - the one that jumps one barrier goes to NextPhase().
          - the other goes to the compatible through phase.
```

4.2 Interval timing

The execution of a phase is illustrated in Figure 5. The dual-ring controller initiates the phase at "START". This point is synchronized with the transition from green to yellow of the previous phase ("END"). The initial wait period is equal in length to the yellow and red clearance intervals of the previous phase. Thus, the transition from wait to green is simultaneous with the transition from red clearance to idle of the outgoing phase. The green interval is divided into two portions: the initial green interval and the extension interval. The duration of the initial green interval is calculated based on the maximum number of vehicles registered by the approach loops during the preceding red interval. Each detected vehicle increases the green period by `addpervehicle`. The result is limited by `mingreen` and

`maxinitial`:

$$\text{initial green} = \min\Big\{ \max\{ \text{(largest count)} \times \texttt{addpervehicle} \, , \, \texttt{mingreen} \} \, , \, \texttt{maxinitial} \Big\}$$

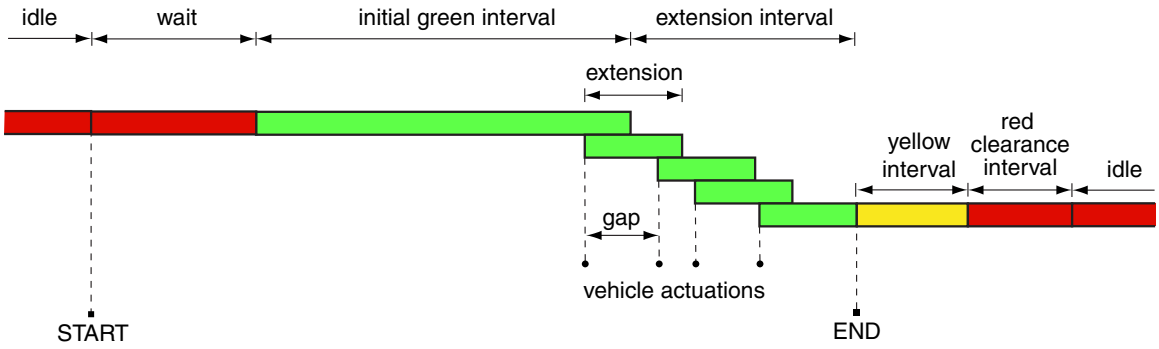

Figure 5: Interval timing

During the green interval, vehicles detected by an approach loop are given `extension` seconds of green time to move through the intersection, without exceeding the maximum green duration of `maxgreen`. The green interval ends when the maximum green time is reached (max out), or when the time gap between consecutive vehicle actuations exceeds the largest permitted gap (gap out). The permitted gap is a function of time, as plotted in Figure 6. It starts at `maxgap` and begins to decrease after a vehicle is detected on a conflicting phase. The size and frequency of the gap reduction is determined by `reducegapby` and `reduceevery`, and the minimum value is `mingap`. The yellow and red clearance intervals have durations `yellowtime` and `redcleartime`.

The parameters involved in the ASC algorithm are entered by the user in `param_asc.txt`. A sample input file is shown in Figure 7. The `recall` token is used to set the recall status of the phases to *on* (1) or *off* (0). Entries in the `lagleft` line indicate whether left turns are lagging (0) or leading (1). The remaining parameters are defined in seconds.

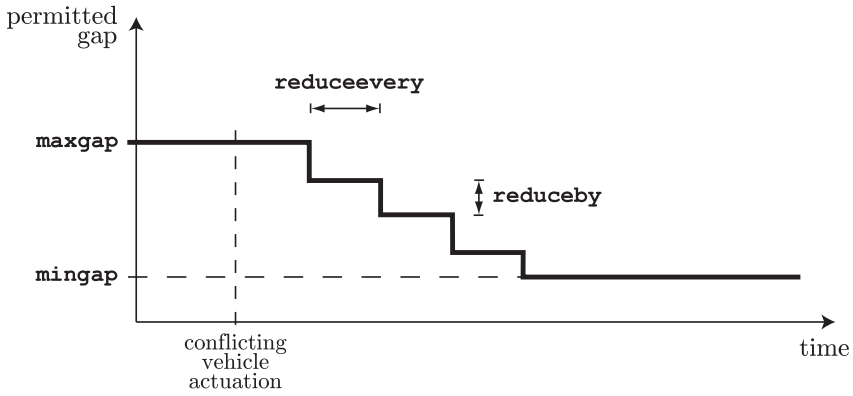

Figure 6: Permitted gap function

```
node 10
% phase#        1     2     3     4     5     6     7     8
recall          0     1     0     0     0     1     0     0
lagleft         1     0     0     0     1     0     0     0
mingreen        10    20    10    40    10    20    10    40
addpervehicle   2.1   2.1   2.1   2.1   2.1   2.1   2.1   2.1
maxinitial      20    30    20    50    20    30    20    50
maxgreen        30    40    30    60    30    40    30    60
extension       2.5   3.5   2.5   3.5   2.5   3.5   2.5   3.5
maxgap          2.5   5.0   2.5   2.5   2.5   5.0   2.5   2.5
mingap          2.5   2.5   2.5   2.5   2.5   2.5   2.5   2.5
reducegapby     0.1   0.1   0.1   0.1   0.1   0.1   0.1   0.1
reduceevery     2.0   2.0   2.0   2.0   2.0   2.0   2.0   2.0
yellowtime      3.0   3.5   3.0   3.5   3.0   3.5   3.0   3.5
redcleartime    1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0

node 11
% phase#        1     2     3     4     5     6     7     8
recall          0     1     0     0     0     1     0     0
lagleft         0     0     0     0     0     0     0     0
mingreen        10    20    0     40    10    20    0     40
addpervehicle   2.1   2.1   0     2.1   2.1   2.1   0     2.1
maxinitial      20    30    0     50    20    30    0     50
maxgreen        30    40    0     60    30    40    0     60
extension       2.5   3.5   0     3.5   2.5   3.5   0     3.5
maxgap          2.5   5.0   0     2.5   2.5   5.0   0     2.5
mingap          2.5   2.5   0     2.5   2.5   2.5   0     2.5
reducegapby     0.1   0.1   0     0.1   0.1   0.1   0     0.1
reduceevery     2.0   2.0   0     2.0   2.0   2.0   0     2.0
yellowtime      3.0   3.5   0     3.5   3.0   3.5   0     3.5
redcleartime    1.0   1.0   0     1.0   1.0   1.0   0     1.0
```

Figure 7: Input file for actuated signal control (`param_asc.txt`)

5 Traffic responsive control (TRSP)

The two coordinated modes included in the first generation UTCS are time-of-day and traffic responsive control. TOD is often sufficient for predictable systems, however the more adaptive TRSP option is preferred if there are significant day-to-day fluctuations in demand. This is because TRSP can automatically respond to variations by selecting an appropriate signal timing plan for the measured traffic condition.

Under TRSP control, a centralized system monitors the traffic measurements from *system loops* at several intersections. The system loops are usually the approach loops on the main arterial. The flow and occupancy measurements from these loops are used to calculate a characteristic 'vpko' (volume plus K occupancy) value for each loop l :

$$\text{vpko}(l) \;=\; \text{vol}^s(l) \;+\; \text{K} \times \text{occ}^s(l)$$

Here, $\text{vol}^s(l)$ and $\text{occ}^s(l)$ are the smoothed volume and occupancy measurements for system loop l. K is a system constant. The current state of the network is represented by the array of all $\text{vpko}(l)$ values.

The controller also stores a number of signal timing plans with associated vpko signatures. The signature for the p-th plan consists of vpko values for each of the system loops $\overline{\text{vpko}}(p, l)$. The controller selects a timing plan from its library by comparing the stored signatures to the measured traffic signature, and finding the best match in terms of a weighted 1-norm:

$$\Delta(p) \;=\; \sum_l \text{W}(l) \,\big|\, \text{vpko}(l) - \overline{\text{vpko}}(p, l) \,\big| \qquad \text{for each plan } p$$

$\text{W}(l)$ are loop-specific weighting factors. The plan with the smallest $\Delta(p)$ value is considered the best option for the current traffic condition. A transition to this plan will be initiated if the $\Delta(p)$ for the currently active plan is larger than a prescribed threshold. As with TOD control, the transition process is approximated with a user-defined delay interval.

Figure 8 shows the input file for TRSP control. The traffic responsive calculation is performed every `updatetime` seconds. `kweight` is the value of K. The system loops and their respective weighting factors $\text{W}(l)$ are listed with the `det` token. The timing plans are

```
updatetime   900          % 15 minutes
minchange    1.0
kweight      20

% System loops ............................
%---  Loop name    W
det    n10_2_A_1   3.0
det    n10_2_A_2   3.0
det    n10_6_A_1   3.0
det    n10_6_A_2   3.0
det    n11_2_A_1   3.0
det    n11_2_A_2   3.0
det    n11_6_A_1   3.0
det    n11_6_A_2   3.0

% Signatures ...............................
plan 1
%---  Loop name   Flow   Occ
sig    n10_2_A_1   500    0.7
sig    n10_2_A_2   500    0.7
sig    n10_6_A_1   100    0.1
sig    n10_6_A_2   100    0.1
sig    n11_2_A_1   500    0.7
sig    n11_2_A_2   500    0.7
sig    n11_6_A_1   100    0.1
sig    n11_6_A_2   100    0.1

plan 2
%---  Loop name   Flow   Occ
sig    n10_2_A_1   100    0.1
sig    n10_2_A_2   100    0.1
sig    n10_6_A_1   500    0.7
sig    n10_6_A_2   500    0.7
sig    n11_2_A_1   100    0.1
sig    n11_2_A_2   100    0.1
sig    n11_6_A_1   500    0.7
sig    n11_6_A_2   500    0.7
```

Figure 8: Input file for traffic responsive control (`param_trsp.txt`)

described in `param_tod.txt`. The characteristic signatures for each of these plans are defined using `plan` and `sig`. The syntax for the `sig` line is:

```
sig [loop name] [signature flow [vph]] [signature occupancy ∈(0,1)]
```

6 Critical Intersection Control (CIC)

Critical intersection control is a feature of many UTCS softwares which complements the TRSP strategy. This feature enables the controller to respond more quickly to short-term variations in demand, while preserving coordination on the major street, by automatically adjusting the green times of *critical intersections* based on local measurements. CIC requires that all of the approaches to the critical intersections be equipped with approach loops.

```
updatecycles  1
gdfunction    ATSAC
gdcoef        7.5 0.5 0.33 1.0
critical      10 11

%-------  node  1    2    3    4    5    6    7    8
mingreen  10    0    12   0    7    0    12   0    7
```

Figure 9: Input file for critical intersection control (`param_cic.txt`)

CIC periodically calculates the total *green demand* for all phases in a critical intersection using volume and occupancy measurements from the approach loops. This amount is reduced by the minimum green duration to obtain the excess green demand for each stage. The actual green times are then calculated by distributing the available excess green time among all the stages, in proportion to their excess green demand.

The plugin provides two options for computing the green demands. The first is used by LADOT's ATSAC system (Skabardonis 2000):

$$\mathrm{gd}(l) \;=\; A\left(\mathrm{vol}^s(l)\right)^B \;+\; C\left(\mathrm{occ}^s(l)\right)^D$$

$\mathrm{gd}(l)$ is the green demand measured by approach loop l. A, B, C, and D are user-defined coefficients. $\mathrm{vol}^s(l)$ and $\mathrm{occ}^s(l)$ are smoothed volumes and occupancies. The second is the standard UTCS formula appearing in the Traffic Control Systems Handbook (FHWA 1996):

$$\mathrm{gd}(l) \;=\; K_1\,\mathrm{occ}^s(l) \;+\; K_2\,\mathrm{vol}^s(l) \;+\; K_3\,\mathrm{vol}^s(l)\,\mathrm{occ}^s(l)$$

K_1, K_2, and K_3 are user-defined coefficients. The excess green demand for the stage $\mathrm{gd}_e(s)$ is calculated as the largest of the green demands for the associated approach loops, reduced by the minimum green interval, and limited below by zero:

$$\mathrm{gd}_e(s) \;=\; \max\Big\{\; \max_l \big\{\, \mathrm{gd}(l) \,\big\} - \texttt{mingreen}(s) \;;\; 0 \;\Big\}$$

The actual green times $\mathrm{g}(s)$ are found by apportioning the available excess green time G_e to the stages:

$$\mathrm{g}(s) \;=\; \frac{\mathrm{gd}_e(s)}{\sum \mathrm{gd}_e(s)} \;\times\; \mathrm{G}_e \;+\; \texttt{mingreen}(s)$$

$$\text{with} \qquad \mathrm{G}_e \;=\; \texttt{cyclelength} - \sum_s \Big(\texttt{mingreen}(s) + \texttt{yellowtime}(s) + \texttt{redcleartime}(s) \Big)$$

The parameters used by CIC are defined in `param_cic.txt`. A sample input file is shown in Figure 9. `updatecycles` defines the CIC update period as an integer multiple of the cycle length. The green demand formula is selected by setting `gdfunction` to `ATSAC` or `UTCS`. The coefficients in the green demand function are defined with `gdcoef` using the following format:

`gdcoef` [A] [B] [C] [D]	if `gdfunction` = `ATSAC`
`gdcoef` [K_1] [K_2] [K_3]	if `gdfunction` = `UTCS`

`critical` lists the node numbers for the critical intersections. `mingreen` is an optional input, which can be used to redefine the minimum green time used by CIC. If set to zero, the minimum green time will be that of `param_tod.txt`. Otherwise `mingreen`(s) will be set to the largest of the entries for the phases combined in stage s. The `cyclelength`, `offset`, `yellowtime`, and `redcleartime` are taken from `param_tod.txt`.

7 Conclusion

This document has described a new plugin for the Paramics simulation software which includes implementations of time-of-day, traffic responsive, critical intersection control, and actuated signal control. We are currently conducting a thorough test of the plugin, and plan in the future to do a comparative study of traditional controllers, such as UTCS and actuated, and adaptive controllers such as ACS Lite, RHODES, and TUC. The plugin can be obtained from the authors by request (gomes@path.berkeley.edu, dromeas@berkeley.edu).

This work is funded under Task Order 5322 of the California PATH Program.

References

Chu, L., M. McNally, and W. Recker. 2004. Development of the capability-enhanced PARAMICS simulation environment (Final report for PATH T.O. 4304).

Federal Highway Administration (FHWA). 1996. Traffic control systems handbook.

Henry, R. D. , R. A. Ferlis, and J. L. Kay. 1976. Evaluation of UTCS conrol strategies. Technical Report FHWA-RD-76-150, JHK & Associates / Peat, Marwick, Mitchell & Co.

Honeywell, Inc. 1979. UTCS-functional description-enhanced first generation software. Technical Report FHWA-TS-79-228, Honeywell, Inc.

Kay, J. L., J. C. Allen, and J.M. Bruggeman. 1975. Evaluation if the first generation UTCS/BPS control strategy - Final report. Technical report, JHK & Associates / Peat, Marwick, Mitchell & Co.

Skabardonis, A., B. Gallaggher, and K. Patel. 2000. Determining the capacity benefitsd of real-time signal control at an intersection. California PATH Working Paper UCB-ITS-PWP-2000-25.