EVALUATION OF THREE DISTINCT ADAPTIVE CONTROL STRATEGIES
FOR NJ STATE HIGHWAYS USING PARAMICS

Harsh Doshi, MS.C.
Schoor DePalma Inc.,
200 US Highway 9, Manalapan, NJ 07726
Tel: 732.577.9000
Fax: 732.577.9888
Email: hdoshi@schoordepalma.com

Kaan Ozbay, Ph.D.
Associate Professor
Associate Director, Center for Advanced Infrastructure and Transportation (CAIT)
Rutgers University, Civil and Environmental Engineering
632 Bowser Road, Piscataway, NJ 08854-8014
Tel: 732.445.2792
Fax: 732.445.0577
Email: kaan@rci.rutgers.edu

Jihong Chen
Graduate Research Assistant
Rutgers University, Civil and Environmental Engineering
632 Bowser Road, Piscataway, NJ 08854-8014
Tel: 732.445.2792
Fax: 732.445.0577
Email: jihongch@rci.rutgers.edu

Boile Maria, Ph.D.
Assistant Professor
Rutgers University, Civil and Environmental Engineering
632 Bowser Road, Piscataway, NJ 08854-8014
Tel: 732.445.7979
Fax: 732.445.0577
Email: boile@rci.rutgers.edu

Contact Person
Harsh Doshi

Word Count: 8121
Submitted to 85th TRB Annual for Possible Presentation and Publication
ABSTRACT

Theoretically, adaptive control strategies (ACS) have the ability to reduce a performance index (like delay, emission etc.) at an intersection when compared with pre-timed traffic signal control strategies. However, theoretical levels of benefit are not always achieved when these systems are implemented under real-world conditions. Errors due to detection and estimation procedures may have different effects on networks with different characteristics like geometry, traffic demand and number of phases in signaling scheme. Hence, it is beneficial to know how much improvement can be expected when these strategies are implemented on networks with different characteristics. This study compares the performance of three adaptive control strategies by using microscopic simulation tool PARAMICS. Prototypes for reactive (SCOOT-like), Case-based reactive (SCATS-like) and proactive / predictive (OPAC-like) algorithms, each using a different control logic to control signal timings, were developed in Paramics Programmer. These prototypes were tested on various well-calibrated NJ state highway intersections. It was observed that reduction in travel time achieved using these various adaptive control strategies peaked at different volume-to-capacity ratios for networks with different characteristics. The paper also identifies salient features of these distinct adaptive control strategies in terms of benefits and concludes that the benefits from these strategies tend to decrease with higher demand on cross streets at higher volume-to-capacity ratios. Jug-handle turns or closely spaced intersections also affect benefits that can be achieved using these algorithms. With higher number of phases and highly varying traffic demand, reactive algorithms are observed to be slow to respond, decreasing the level of benefits.
INTRODUCTION
This paper evaluates the performance of three distinct approaches to adaptive traffic signal control, using realistic and well-calibrated microscopic simulations. Algorithms that emulate the logic behind SCOOT, SCATS and OPAC control strategies have been developed to identify the control strategy that works best on networks with different characteristics such as traffic demand, network geometry, and number of phases in a signaling plan. Microscopic simulation model of portions of three major arterials in New Jersey, namely NJ Route 10, 18, and 23 have been developed in Paramics. In this paper, intersections along these three arterials will be used for the evaluation of three distinct adaptive signal strategies.

BACKGROUND
Adaptive traffic signal control has the potential to reduce a set of performance measures such as stopped and total delay, emissions, travel time, and average number of stops at an intersection by responding to changes in traffic demand in real-time. With the advances in communication and computational techniques many adaptive traffic signal control algorithms are now being deployed in various parts of the world. Each of these algorithms has a different concept of controlling signal timing at an intersection. It can be expected that due to inherently different ways of controlling signal timings, reduction in performance measures achieved using these systems would also vary. Some of these systems were previously tested using microscopic and macroscopic simulation tools and were also implemented at various locations across the world. However, traffic engineers are still looking to better understand the benefits from adaptive control strategies and have a difficulty in selecting a strategy that best suits the specific network conditions on which they are trying to implement them.

Under the Federal Highway Administration’s (FHWA) Real-Time Traffic Adaptive Control Systems (RT-TRACS) program, an effort was made to test promising ACS using both microscopic simulations and field implementation. This was a unique effort to test various ACS against a common baseline control (1). However this program evaluated the strategies with an aim to further develop most promising ACS amongst Real-time Hierarchical Optimized Distributed Effective System (RHODES), Optimized Policies for Adaptive Control (OPAC), University of Minnesota ARTS, ISAC:AFT, and University of Maryland RTACL (1). The strategies were tested on only arterial networks. Likewise, limited testing has been done on other types of networks like grid, and isolated intersections (2-11). Further effects of number of phases, varying traffic demand, and cross street demand is not clear from the available literature. Under such circumstances, it becomes difficult to estimate the performance of adaptive control strategies for networks with different characteristics without actually deploying them. Traffic agencies end up spending large amount of money for upgrading traffic signals before actually knowing the level of benefits that can be expected from such systems. Also these systems are sometimes difficult to maintain and there are instances when traffic control was switched back to actuated control strategy (2).
Several case studies aimed at evaluating the performance of different ACS like SCOOT, SCATS, OPAC, LADOT, and RHODES are available in literature (2-11). These systems were implemented on different types of networks ranging from grid, arterial and isolated intersections. The baseline control, against which their performance was compared, was different in each of these studies. Also traffic networks had different levels of congestion and the statistical methods used to evaluate the performance were also different. Hence with limited implementation results true benefits of adaptive control systems are difficult to estimate.

According to Shelby (12), the first step in evaluating the performance of adaptive control strategies is to check if they can outperform well-calibrated signal control at isolated intersections. If these strategies fail to generate “benefits for isolated intersections, then they might not perform well on network-wide levels”. As a first step in the evaluation procedure, this study focuses on the performance of adaptive control strategies on isolated intersections. The strategies developed in this study are based on available mathematical formulations of OPAC, SCOOT, and SCATS in literature. The study evaluates three different adaptive control strategies each using different concepts of controlling signal timing at an intersection and identifies salient features in these algorithms that works well on different types of networks.

The selected adaptive control strategies use different approaches to adaptive signal control. SCOOT is a reactive control strategy. It reacts to varying traffic by changing cycle length and phase splits in small increments. SCATS, a reactive/case-based strategy, adapts to varying traffic by selecting a timing plan from an offline-stored library of plans that suits best to current traffic demand. OPAC on the other hand is a proactive control strategy. It detects current traffic demand, and predicts future arrivals at an intersection to select a switching strategy that will reduce delay over short time intervals in the future. Thus the three selected strategies have different concepts of controlling signal timings at an intersection.

While there are many studies that test adaptive control strategies based on a macroscopic simulation model developed using a higher level programming language, this approach is not considered prudent for testing adaptive control strategies. This is because under macroscopic simulation models, it is assumed that the signal control will be based on true values of current as well as estimated traffic demand. However, in real-time situations it is never possible to acquire perfect information from detection and estimation procedures, and hence, the benefits of adaptive signal strategies can easily be over-estimated using macroscopic simulation models. Also one of the main reasons why adaptive control strategies are not tested using microscopic simulations is because of the computational difficulty involved in developing a software code which would run in conjunction with a microscopic simulation tool.

Paramics simulation suite was selected to perform microscopic simulations because it provides an Application Programming Interface (API), through which a user can add, new functionality or modify existing one. An object-oriented software code called EZParamics (15) was developed to decrease the level of computational efforts and allow
code reuse. With this approach, it is now possible to extend the research to simulate arterial and grid networks. The functioning as well as the capabilities of EZParamics is outside the scope of this paper and more information is available in the literature (15). For testing these algorithms, New Jersey State Highway arterials were first calibrated for average intersection delay (seconds per vehicle) and traffic volume (vehicle per hour per lane) \(^{13}\). The baseline control selected for this study was optimized pretimed signal control.

The paper discusses:

1. Comparison and evaluation of three most popular approaches to adaptive control strategies namely, reactive, reactive/case based, and proactive (predictive).
2. A comprehensive comparison of each strategy for the same networks/intersections to achieve an objective evaluation with the same baseline.
3. Conduct these comparisons for a number of various network conditions to study the impact of these network conditions on the performance of the control strategies. Use well calibrated microscopic simulation models of test intersections that reflect real-world conditions observed in the field.
4. Evaluation of these strategies using a well-accepted microscopic simulation software namely Paramics to get as close as possible to the real-life conditions where uncertainties due to individual car behavior and arrival patterns might reduce the performance of these control strategies. In fact the use of a microscopic simulation, which is in many ways a black box to the researchers, also ensures the objectivity of the conclusions.

The next section discusses the logic used to implement the three different adaptive control strategies.

**Proactive – Predictive Control Logic (OPAC-like logic)**

A basic version of a proactive – predictive OPAC-like algorithm was developed using mathematical formulations available in the literature (16). This control strategy makes decision to change or not to change the current signal status at short and fixed time intervals. This decision is based on the current status of the signal (green or red), queue length at each approach at the beginning of each interval as well as the number of vehicles, which will arrive on each approach during the current and future time intervals. For computational simplicity it was assumed that if there is a decision to change the signal status there is no movement on either intersection approaches for one time interval (which is similar amber and all red time).

Let,

\[ q_{ij} = \text{A state variable describing the length of the queue on approach } j, j=1,2. \]
\[ s_{ij} = \text{A state variable describing the status of the signal on approach } j, j=1,2. \text{ A value of 0 corresponds to green and 1 to red. Note that if } s_{1i} = 0, \text{ then } s_{2i} = 1 \text{ and vice versa.} \]
\[ a_{ij} = \text{A state variable describing the number of arrivals on approach } j \text{ during interval } i, i = 1,2,\ldots,N \]
\( d_i \) = The decision variable associated with interval \( i \). A value of 0 corresponds to a decision of no switch while a value of 1 corresponds to a decision to switch the signal.

\( Q_i = [q_{i1}, q_{i2}] \) = Vector containing length of the queue on approach \( j \) (\( j = 1, 2 \)) for interval \( i \).

\( A_i = [a_{i1}, a_{i2}] \) = Vector containing arrivals on each approach for interval \( i \).

\( S_i = [s_{i1}, s_{i2}] \) = Vector containing signal status at the beginning of interval \( i \).

\( C(Q_i, S_i, A_i, d_i) \) = Cost function incurred at stage \( i \), as a result of decision \( d_i \).

Then,

\[
C(Q_i, S_i, A_i, d_i) = \begin{cases} 
T^*(q_{i1} + q_{i2} + \frac{1}{2}(a_{i1} + a_{i2})) & d_i = 1 \\
T^*(q_{i1} + \alpha_{i2} + \frac{1}{2}(a_{i1} + \max(0, a_{i2} - \alpha_{i2}))) & d_i = 0, s_{i1} = 1 \\
T^*(q_{i2} + \alpha_{i1} + \frac{1}{2}(a_{i2} + \max(0, a_{i1} - \alpha_{i1}))) & d_i = 0, s_{i1} = 0 
\end{cases}
\]

Where, \( T = \) length of the discrete time interval.

\( \alpha_{ij} = \begin{cases} 
\left(2 \cdot \frac{T}{5}\right) - q_{ij} & \text{... If, } \left[\left(2 \cdot \frac{T}{5}\right) - q_{ij}\right] \geq 0 \\
0 & \text{……... Otherwise.}
\end{cases} \)

Queue length after each interval on each approach would depend upon the decision variable as well as the initial signal status. The queue length can be defined as:

\[
q_{i+1,j} = \max(0, q_{ij} + a_{ij} - 2[(1 - s_{ij})(1 - (s_{ij} + d_i)_{MOD2})])
\]

The signal status after each decision variable will be

\[
s_{i+1,j} = (s_{ij} + d_j)_{MOD2}
\]

Based on the dynamic programming principle of optimality, the best decision \( d_i \) at each stage \( i \) will be the one which minimizes the performance measure from stage \( i \) to stage \( i = N \).

\[
OP_i = \min_{d_i = 0, 1} C(Q_i, S_i, A_i, d_i) + OP_{i+1}
\]

(Where \( OP_i \) is the optimal value of performance measured from stage \( i \) onwards).

The above algorithm is relatively easy to implement using a macroscopic simulation model developed using any higher programming language because in a macroscopic simulation, the traffic flow as well as arrival and departure models will be well known and it will be relatively easy to tweak the above control strategy to produce results closer
to the expected ones. However, for microscopic simulation implementation many parameters discussed in the above formulation are not known a priori and they have to be estimated in a way that is very similar to the real-world implementation of these algorithms. For microscopic simulations, number of departures during each time interval was derived from saturation flow rate of each approach. Queue was estimated as the difference in vehicle count of stop-line detector and a detector placed at 5 seconds in the upstream direction from the stop line. Arrivals during the head period of the horizon were estimated using detectors located at 5, 10, and 15 seconds in the upstream direction and subtracting the count of upstream detector with the count at the next downstream detector. To make the algorithm more realistic, green time was restricted between a maximum and minimum green time. Minimum green time was used as the time necessary clear average queue obtained using optimized pre-timed signal control. Keeping the maximum cycle length of 150 seconds, maximum green time was obtained by dividing the splits based on traffic volume (13). The flowchart used to implement the above algorithm is shown in Figure 1.

**Reactive control logic (SCOOT-like Prototype)**

Reactive control logic, SCOOT optimizes phase timings and offsets in a network by using data collected from detectors located in upstream direction of each approach at an intersection. Signalized junctions in the area controlled by such control logic are grouped into “sub-areas”. Usually all intersections within a sub-area are operated on a common cycle length. In response to the change in demand, a cycle optimizer varies the cycle length of each sub-area by a small amount, usually a few seconds. The frequency of this optimization is 5 minutes, but not less than 2.5 minutes (18). The algorithm used under such control logic continuously measures the degree of saturation for each movement in the sub-area. The most heavily loaded intersection determines the change in cycle length in the sub-system. Since this study considers only isolated intersections, the cycle length is changed based on the Degree of Saturation ($DS$) of the entire intersection. This value is calculated as the average value of the degree of saturation of all approaches at an intersection.

The optimization process used in this study follows the criteria shown below (13):

1. If the $DS \leq 80\%$ for an entire intersection, the cycle-length optimizer will reduce the cycle length by 4, 8, 16 seconds. Decrement in cycle length was kept 4 seconds in this study.
2. If the $DS \geq 90\%$ for an entire intersection, the cycle optimizer will increment the cycle length by 4, 8, or 16 seconds to increase capacity. Again, the increase is considered to be 4 seconds in this study.
3. If the maximum degree of saturation is in the range of 0.8 to 0.9, the cycle length maintains the same value.
4. The cycle length is constrained between minimum and maximum values. These values are calculated using maximum and minimum values of phase split discussed below.
The split optimizer optimizes the green time split between different phases in the signaling scheme. This optimization is done every 5 seconds before the end of an ongoing cycle length. It estimates whether it is beneficial to make a change in green time earlier, as scheduled, or later according to the degree of saturation. The objective of this optimization is to minimize the maximum degree of saturation on all approaches. Along with maximum degree of saturation, queue value is also used in the decision process. Maximum and minimum allowable queue lengths are determined from network geometry. Any decision by the optimizer may alter a scheduled stage change time by a few seconds. Usually the split time is temporarily changed by –4, 0, +4 and permanently changed by +1, 0, -1 seconds. The duration of green time is constrained by the minimum green time, maximum green time and fixed green time lengths, which is equal to the optimized pre-timed signal control green. Maximum and minimum green times are calculated in similar fashion as described in OPAC-like algorithm. Offset optimization is not used since the scope of this study is limited to isolated intersections. The flowchart used to implement this algorithm is shown in Figure 2.

**Reactive/Case based control logic (SCATS-like Prototype)**

As mentioned earlier, a reactive/case based control logic such as the one used in SCATS is selected for this study. This strategy is termed as reactive/case based because it reacts to a different “cases” of traffic demand by selecting an appropriate signal plan. Using the data collected from detectors, it calculates the degree of saturation as the ratio of effectively used green time to the total available green time. Unlike SCOOT, detectors under SCATS control are placed immediately before the stop line.

Reactive/Case-based control logic usually calculates four or eight splits plans for each intersection. It nominates a phase serving highest traffic demand as a “stretch phase.” The SCATS algorithm computes phase splits based on required green time for each phase during the peak period. It also computes a stretch cycle length based on green times of the selected stretch phase. The algorithm to calculate phase splits is described in more details in the literature (13,19). SCATS perform a split plan vote to select a split plan at the end of each cycle. It calculates expected degree of saturation for each phase using the equation:

\[ DS_{ij} = DS_{ic} \cdot \frac{g_{ij}}{g_{ic}} \]

Where,
- \( DS_{ij} \) = degree of saturation on movement \( i \) under phase timing plan \( j \),
- \( DS_{ic} \) = degree of saturation on movement \( i \) under current phase timing plan,
- \( g_{ij} \) = green time on movement \( i \) under phase timing plan \( j \),
- \( g_{ic} \) = green time on movement \( i \) under current phase timing plan

It then compares the maximum \( DS \) under each split plan and assigns a positive vote to the plan that has a minimum \( DS \) value among all the maximum \( DS \). If a plan gets two votes in three consecutive cycles, it is selected, otherwise, the signal control is based on current split plan.
SCATS calculates the cycle length based on the intersection with highest traffic demand within a group of intersections. Since this study focuses on isolated intersection, SCATS-like prototype developed for this study uses maximum DS value amongst all approaches to calculate cycle length change as shown below:

\[ C'' = C + C' \]
\[ C' = 60 \times (Max(DS) - f(C)) \]
\[ f(C) = \frac{0.9 - 0.5(C - C_x)}{C_{max} - C_x} + 0.5 \]

Where,
\( C'' \) = New cycle length,
\( C \) = Previous cycle length,
\( C' \) = Changes of cycle length,
\( C_{max} \) = Maximum cycle length (seconds),
\( C_x \) = Medium cycle (seconds).

The cycle length is restricted between a maximum and minimum value. Similar to the stretch cycle length, SCATS-like algorithm selects a medium cycle length. It allocates the time change in cycle length to different phases using the following logic:

Let,
\( C_{max} \) = Maximum cycle (seconds)
\( C_{min} \) = Minimum cycle (seconds)
\( C_x \) = Stretch cycle length (seconds)
\( C_s \) = Medium cycle (seconds)

The allocation of the increment to different splits in a cycle follows the rules below:

1. \( C_{min} \leq C \leq C_x \), If the rate of flow measured by the nominated strategic detectors falls below a preset value, the cycle length can only operate on \( C_{min} \). If the split of each phase equals the \( C_{min} \), multiply the percentage of each phase in the selected phase split plan,

2. \( C_x \leq C \leq C_{max} \), Cycle length is allocated to each phase according to the percentage of each phase in the selected timing plan,

3. \( C_x \leq C \leq C_{max} \), \( C_x \) is allocated to each phase according to the percentage of each phase in the selected phase. The difference of \( C - C_x \) goes to stretched phase.

The optimized pre-time cycle length for an intersection was selected as the medium cycle length for this study. Maximum and minimum cycle lengths were selected using the same
concept described for OPAC-like prototype. Offset optimization is not used since the scope of this study is limited to isolated intersections. The flowchart used to implement a SCATS-like algorithm based on the above logic is shown in Figure 3.

PARAMICS IMPLEMENTATION OF ADAPTIVE CONTROL PROTOTYPES

As mentioned earlier, OPAC-like, SCOOT-like, and SCATS-like prototypes were developed using information available in open literature. These prototypes were then tested using microscopic simulation models developed in Paramics. The main difficulty in such an endeavor, beyond the development of the adaptive control algorithm, is in its integration with the microscopic simulation tool. For this integration, the Application Programming Interface (API) of Paramics was used to add new functionality or modify existing ones. Vissim and Corsim also provide such functionalities and their performance may vary from Paramics. One of the main drawbacks that have prevented widespread use of Paramics API is the steep learning curve involved. This can be traced to fact that development is performed in C language, a procedural programming language. In programming language community, it is well known that in a procedural programming language, data and program are interspersed. Operations that modify a data item may be spread over the entire code. This creates dependencies among several portions of the program leading to the following drawbacks:

- **Difficult to understand:** Understanding a function may require knowledge of all the dependent functions.

- **Difficult to modify:** Dependence between various functions in the code means that changes may not be localized to a certain part, instead being spread over multiple parts of the program.

- **Difficult to re-use:** Re-using a specific function in a different code is not straightforward because of the dependencies.

To overcome these difficulties in developing an API, a simpler programming environment called EZParamics (16) was developed using the Object Oriented (OO) programming concept. EZParamics extend Paramics in two significant ways:

*Simplifies the main body of the plugin:*

It moves the code from the main body to the various classes defined in the code, simplifying it considerably. The main body is now only involved in instantiating and initializing objects with appropriate parameters.

*Promote code re-use:*

The classes in EZParamics contain the entire logic. Classes in OO programming promote re-use of code. It allows building a library of classes that makes development easier and faster. EZParamics user needs to concentrate only on the specific class that needs to be
extended or modified. This considerably simplifies programming. Library of classes also leads to standardization of commonly used functionality.

The main class in EZParamics is called EZObject. It is the parent class for all other classes in EZParamics. It defines the default behavior of all the overload functions available in Paramics. A new class can optionally modify this default behavior. Defining EZObject has two advantages:
1. Provides a uniform interface to all the classes in EZParamics.
2. Frees the user from worrying about the functions not related to the specific application being implemented.

The structure of program using EZParamics is shown in Figure 4-b.

The figure shows a program that defines two new classes – Intersection and Vehicle. Both these classes extend EZObject class. The api_setup() function in the main body is used to create and initialize objects in EZParamics. Additionally it calls the initialize() function. This function simply calls the api_setup() member function for each of the object created in EZParamics. All overload functions in the main body have the same basic structure – they simply call the corresponding overload function for each of the object created in EZParamics. For example, the net_post_action() in the main body simply calls the net_post_action() member function of each of the object created in EZParamics.

Since Paramics simulations with different seed values give different results, it is essential to use a suitable output analysis method to determine the number of replications required for simulation of each intersection. The number of replications required, were determined using sequential method (20). This statistical procedure aims to obtain the mean \( \mu = E(X) \) of the selected performance measure \( X \), within a specified precision.

If we estimate \( \overline{X} \) such that \(|\overline{X} - \mu|/|\mu| = \gamma\), then \( \gamma \) is called the relative error of \( \overline{X} \). The specific objective of this procedure is to obtain an estimate of \( \mu \) with a relative error of \( \gamma \) and a confidence level of 100(1-\( \alpha \)) percent. If we denote the half-length of the confidence interval by \( \delta(n, \alpha) \) then the sequential procedure is as follows:

1. Make \( n_0 \) replications of the simulation and set \( n = n_0 \)
2. Compute \( \overline{X}(n) \) and \( \delta(n, \alpha) \) from \( X_1, X_2, \ldots, X_n \)
3. If \( \delta(n, \alpha)/|\overline{X}(n)| \leq \gamma' \), use \( \overline{X}(n) \) as the point estimate for \( \mu \) and stop. If not, replace \( n \) by \( n + 1 \), make an additional replication of the simulation and go to step 1

Where, \( \gamma' = \gamma/(1 - \gamma) \)
Simulation runs with different seed values were stopped once relative error was within 10%.

TESTING OF ADAPTIVE CONTROL ALGORITHMS ON NJ HIGHWAYS

Portions of three NJ state highways 10, 18, and 23 were used for this evaluation study. Networks for these routes were developed using Paramics Modeller. Simulated portions of NJ route 10 and 23 have 15 intersections while NJ route 18 has 5 intersections. However, the reliability of any simulation software model depends on its ability to produce results close to actual data. When the networks were created, base runs are taken to observe the model accuracy without any adjustments, and the quality of data inputs are provided. Initial runs provided incompatible results for flow, except for Route 18. Calibration was required to get the similar flow in the simulation output compared to the data input.

A widely used error measure that can provide a fairly good initial estimate of the degree of fit between the simulated and the actual traffic measurements is the Root Mean Squared Percent Error (RMSE). This error value was used to test the accuracy of microscopic simulations. This error gives an estimate of the total percentage error and is defined as

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left( \frac{x_i - y_i}{y_i} \right)^2}$$

where RMSE is the root mean squared percentage error

- $x_i$ is the simulated traffic measurement value of run $i$
- $y_i$ is the actual traffic measurement value of run $i$

To get the observed flow for NJ route 23, various mean headway and mean reaction time were tested until RMSE stopped decreasing. In Paramics, the network is not loaded with vehicles when the simulation begins, instead they are generated after the simulation starts. To avoid discrepancy in flow values due to this, simulation was run for two hours. It was observed that vehicles are evenly spread across the network after one hour, namely the warm-up period, and results for the second hours were considered for the analysis. When all these changes were made, results provided the correct flow and Paramics gave results comparable to real-world observations. Root mean square percentage error obtained was between 3 and 6\(^{(13)}\).

For, route 10, to balance the flow between two intersections, additional intermediate links were added to the base model. These additional links were not modeled in the base SYNCHRO model used to develop pre-timed signal timings because SYNCHRO is not sensitive to real-world network geometries. These additional links were needed to represent the entrance to and exit from shopping centers, parking lots or any minor streets. In Paramics animation, the congestion was found at junctions where additional
links joined the main route. To calibrate the flow in Paramics these links were repositioned to closely represent real-world geometries. RMSE was between 3 and 6. (13)

Individual intersections were then selected from the whole arterial keeping the values of mean headway and mean reaction time same as the ones in the calibrated arterials.

After testing and debugging the selected adaptive control strategy prototypes developed using EZParamics, they were tested on intersections from New Jersey State Highway 10, 18, and 23. These intersections were selected based on different network geometry, traffic demand and number of phases in the traffic signal plan. The volume to capacity ratio on the main street was varied from 0.1 to 1.0 to test the behavior of these algorithms under different traffic demand on the main street. Finally, plots showing percentage improvements for a pre-defined performance index were generated comparing the results of all three adaptive control algorithms.

To compare the Paramics results, it was necessary to define a performance index. Paramics generates an output file, which has a summary of statistics for travel time, stop time, etc. The file is named as “general” and is generated for every simulation run. The performance index used is given below:

\[ PI = 0.6 \times \text{travel\_time} + 0.4 \times \text{stopped\_time} \]

where,

\text{Travel Time} = \text{Mean travel time (seconds) in transit per vehicle for the network}

\text{Stop Time} = \text{Mean Stop time per vehicle in the network}

The PI for each adaptive control algorithm was compared with the PI of the optimized pre-timed control traffic.

SIMULATION RESULTS

Description of Case Studies:

\text{Intersection of Route 10 and Mount Pleasant Road}
The Mt Pleasant and Route 10 intersection has a jug-handle turn from eastbound route 10 for vehicles to turn onto Mt. Pleasant road. Cross street traffic from Mt. Pleasant road and the jug-handle is very low. The intersection signal runs on two phases.

\text{Intersection of Route 10 and Whippany Road}
The Whippany and Route 10 intersection too has a jug-handle turn from eastbound route 10 for vehicles to turn onto Whippany road. Cross street traffic from Mt. Pleasant Road and the jug-handle is high. The intersection signal runs on two phases.
Intersection of Route 10 and Troy Hills Road
This intersection has jug-handle turns from both directions on Route 10. The cross street demand is low compared with the main street. The signal on the intersection runs on three phases.

Intersection of Route 18 and Eggers/S. Woodland Street
Route 18 and Eggers/S. Woodland Street intersection has a low cross street demand. The signal on this intersection runs on two phases.

Intersection of Route 18 and Tices Lane
This intersection has a jug-handle turn from route 18 northbound sides to vehicles to go onto westbound Tices lane. The signal at the intersection runs on two phases. The cross street demand on this intersection is high.

Intersection of Route 23 and Oak Ridge Road
The intersection of Route 23 and Oak Ridge road has a jughandle turn from northbound Route 23. The cross street demand moving from Oak Ridge when making a right turn on Route 23 southbound is high. The signal runs on two phases.

Intersection of Route 23 and LaRue Road
The intersection of Route 23 and LaRue road is separated by a median, and is there is a group control signal. The cross street demand is low at this intersection.

ANALYSIS
It is difficult to see a consistent trend in the performance of prototype adaptive signal algorithms for different intersections. However as mentioned earlier, these intersections were selected such that they cover a broad range of intersection and network types. The results have effects of each parameter such as cross-street demand, network geometry, level of saturation etc. The results were also compared with the performance of adaptive control strategies reported in literature. It should be noted that for comparison the baseline control strategies, volume-to-capacity ratio, cross street demand, and network characteristics should be taken into account. For example, if there is a jug-handle turn at an intersection, it is difficult to install more detectors along the jug-handle and predict future arrivals during the head period. For (predictive) OPAC-like algorithm that uses future arrivals at an intersection, the shorter link length on jug handles does not allow more detector placements. In such a case, it is possible only to predict the queue on the intersection, and the future arrivals are estimated even for the head period of the horizon. Hence, there is less information needed for dynamic programming model and it can be expected that OPAC-like prototype will act differently for this type of intersections compared with intersections where there is enough space on the approaches to accommodate more detectors. For the reactive SCOOT-like algorithm, the detector would
be placed at the entrance of the jug handle turn. For cases where cross street demand is less, the detector at the entrance of the jug handle does not remain occupied for a longer duration. Hence reactive algorithm would give higher green time for the main street resulting in more stopped time for vehicles on the jug handle. This results in higher delay for SCOOT-like prototype. For reactive/case-based SCATS-like algorithm, the detector at the stop line would mostly remain occupied, however the algorithm would not identify if the queue on jug handle spills on to the main street. This affects the travel time on the main street and there will be reduction in improvement in the performance index. However proactive algorithm (OPAC-like), uses a good queue estimate algorithm, would get the best estimate of number of vehicles on the jug handle and can act to reduce the queue.

The percentage change in the performance index defined above varied for networks with jug handle turns from 2.87% to 33.95% for SCOOT-like (reactive) algorithm, 2.58% to 30.26% for SCATS-like (case-based) algorithm, and 4.47% to 60.31% for OPAC-like (proactive) algorithm. The average reduction the in performance index for higher volume to capacity ratio (0.6 to 1.0) was 17.36%, 7.09%, and 16.11% for SCOOT-like, SCATS-like and OPAC-like algorithms, respectively.

Similarly, the number of phases also has an impact on performance of adaptive signal control prototypes. For Route 10 and Troy Hills Road intersection, SCOOT-like and SCATS-like prototypes fail to generate benefits at higher volume-to-capacity ratio. However OPAC-like prototype gives a consistent improvement in PI, because it can respond to varying traffic demand, by changing the cycle length quickly. SCOOT-like and SCATS-like prototypes may be slower to react since change in cycle length is done in small increments at regular intervals of about 300 seconds. The percentage change in the performance index for networks with higher number of phases was 29.67% for SCOOT-like (reactive) algorithm, 30.26% for SCATS-like (case-based) algorithm, and 60.32% for OPAC-like (proactive) algorithm. However the average reduction the in performance index for higher volume to capacity ratio (0.6 to 1.0) was 3.46%, 11.28%, and 40.53% for SCOOT-like, SCATS-like and OPAC-like algorithms, respectively.

From the results of Route 18/Eggers, S. Woodland street intersection and Route 18/Tices lane, we see that with higher cross street demand, SCOOT-like, SCATS-like and OPAC-like prototypes fail to generate higher benefits at higher volume to capacity ratio. The main street demand in all the cases is similar and both intersections run on two phases. The percentage change in the performance index defined above for intersections with higher cross street demand were 9.82% for SCOOT-like (reactive) algorithm, 9.55% for SCATS-like (case-based) algorithm, and 26.18% for OPAC-like (proactive) algorithm. However the average reduction the in performance index for higher volume to capacity ratio (0.6 to 1.0) was –0.944%, 7.58%, and 14.93% for SCOOT-like, SCATS-like and OPAC-like algorithm. Compared to this, the overall reduction in performance index for intersections with lower cross street demand were 20.49%, 13.46%, and 28.44% for SCOOT-like, SCATS-like and OPAC-like prototypes respectively. For higher volume to capacity ratio the reduction was 18.82%, 13.14% and 35.96% for these three strategies,
respectively. Hence adaptive signal prototypes work well on networks with lower cross street demand.

CONCLUSIONS

This study compares the performance of three adaptive control strategies, each with a different concept of controlling signal timings, on intersections with different characteristics. With higher demand on cross streets it is observed that benefits achieved using adaptive control prototypes reduced at higher volume-to-capacity ratios. Jug-handle turns or closely spaced intersections affect the level of benefits that can be achieved using these algorithms. With higher number of phases and highly varying traffic demand, SCOOT-like and SCATS-like algorithms are slow to respond affecting the level of benefits. In this paper, the prototypes are compared with pre-timed control and it is expected that benefits would be even lesser when compared with actuated signal control. The results summarized in Figure 7(b) closely match with results available in literature. However, these results do not reflect real performance of these ACS systems and they are merely comparison of control concepts they are based on. The approach of horizontal comparison used in this study is a first step in trying to understand how different adaptive control systems would perform under similar conditions using identical implementation platform, in this case, Paramics. The performance of adaptive signal prototypes should not be generalized from the performance of seven intersections used in this paper. Further research is being done to analyze more intersections. However, a detailed cost-benefit analysis should be performed to analyze the true benefits of adaptive control strategies.

REFERENCES


7. William C. Taylor, Ahmed S Abdel-Rahim, “Analysis of Corridor Delay Under SCATS Control”, [www.itsdocs.fhwa.dot.gov/%5CPDOCS%5CREPTS_TE/75V01!.PDF](http://www.itsdocs.fhwa.dot.gov/%5CPDOCS%5CREPTS_TE/75V01!.PDF), accessed on 05/03


14. Ozbay, K., Goel, S., Doshi, H., EZPARAMICS, [www.rci.rutgers.edu/~hpdoshi/index1.html](http://www.rci.rutgers.edu/~hpdoshi/index1.html) accessed on 12/04


LIST OF FIGURES

Figure 1 Flowchart used to implement OPAC-like prototype

Figure 2 Flowchart used to implement SCOOT-like prototype

Figure 3 Flowchart used to implement SCATS-like prototype

Figure 4 EZParamics Program Structure

Figure 5 (a) Percentage Improvements in PI for Adaptive Control for the Intersection of Route 10 and Mt. Pleasant Road (b) Percentage Improvements in PI From Adaptive Control for the Intersection of Route 10 and Whippany Road (c) Percentage Improvements in PI from Adaptive Control for the Intersection of Route 10 and Troy Hills Road

Figure 6 (a) Percentage Improvements in PI from Adaptive Control for the Intersection of Route 18 and Eggers/S. Woodland Street (b) Percentage Improvements in PI from Adaptive Control for the Intersection of Route 18 and Tices Lane (c) Percentage Improvements in PI from Adaptive Control for the Intersection of Route 23 and Oak Ridge Road

Figure 7 (a) Percentage Improvements in PI from Adaptive Control for the Intersection for Route 23 and LaRue Road (b) Summary of Adaptive Signal Control Performance
Figure 1 Flowchart used to implement OPAC-like prototype

Start

Decide Horizon Length
(Roll Period + Projection Length)

Get Initial Signal status and Queue Length for
the first interval of the horizon

Get the number of arrivals for the roll period
from detectors

For the projection length, get the number of
arrivals from historic data

Is the minimum time
for current signal over?

Yes

Keep the current signal
status

No

Is the maximum green
time for current signal
over?

Yes

Change signal status

No

For the horizon length find the optimum switching
sequence considering all possible switching sequences as
per the algorithm described above

Apply the optimum sequence only to the roll
period

Move the horizon ahead by the time equal to the
roll period
Figure 2 Flowchart used to implement SCOOT-like prototype
Figure 3 Flowchart used to implement SCATS-like prototype
Figure 4 (a) EZParamics Working, (b) EZParamics Program Structure
Figure 5 (a) Percentage Improvements in PI for Adaptive Control for the Intersection of Route 10 and Mt. Pleasant Road (b) Percentage Improvements in PI From Adaptive Control for the Intersection of Route 10 and Whippany Road (c) Percentage Improvements in PI from Adaptive Control for the Intersection of Route 10 and Troy Hills Road
Improvement in PI with Adaptive Control

Figure 6 (a) Percentage Improvements in PI from Adaptive Control for Intersection of Route 18 and Eggers Street. (b) Percentage Improvements in PI from Adaptive Control for the Intersection of Route 18 and Tices Lane (c) Percentage Improvements in PI from Adaptive Control for the Intersection of Route 23 and Oak Ridge Road
Improvement in PI with Adaptive Control

![Graph showing improvement in PI with adaptive control](image)

<table>
<thead>
<tr>
<th>Type of Adaptive Control Strategy</th>
<th>Average Reduction in Performance Index</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>With Jug Handles</td>
</tr>
<tr>
<td></td>
<td>Average</td>
</tr>
<tr>
<td>Reactive (SCOOT-like)</td>
<td>17.41%</td>
</tr>
<tr>
<td>Reactive/Case Based (SCATS-like)</td>
<td>11.07%</td>
</tr>
<tr>
<td>Proactive/Predictive (OPAC-like)</td>
<td>21.21%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of Adaptive Control Strategy</th>
<th>Cross Street Demand (Low)</th>
<th>Cross Street Demand (High)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Peak Volume</td>
</tr>
<tr>
<td>Reactive (SCOOT-like)</td>
<td>20.49%</td>
<td>18.82%</td>
</tr>
<tr>
<td>Reactive/Case Based (SCATS-like)</td>
<td>13.46%</td>
<td>13.14%</td>
</tr>
<tr>
<td>Proactive/Predictive (OPAC-like)</td>
<td>28.44%</td>
<td>35.96%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of Adaptive Control Strategy</th>
<th>Number of Phases (&lt;2)</th>
<th>Number of Phases (&gt;2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Peak Volume</td>
</tr>
<tr>
<td>Reactive (SCOOT-like)</td>
<td>15.41%</td>
<td>14.76%</td>
</tr>
<tr>
<td>Reactive/Case Based (SCATS-like)</td>
<td>9.53%</td>
<td>11.58%</td>
</tr>
<tr>
<td>Proactive/Predictive (OPAC-like)</td>
<td>23.71%</td>
<td>21.68%</td>
</tr>
</tbody>
</table>

Comparison of above results with results available in literature

<table>
<thead>
<tr>
<th>Type of Adaptive Control Strategy</th>
<th>% Reduction</th>
<th>Performance Measure</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reactive (SCOOT-like)</td>
<td>-23% to 30%</td>
<td>Total Delay</td>
<td>(3)</td>
</tr>
<tr>
<td>Reactive/Case Based (SCATS-like)</td>
<td>15% to 31.8%</td>
<td>Average Travel Time</td>
<td>(7)</td>
</tr>
<tr>
<td>Proactive/Predictive (OPAC-like)</td>
<td>-17% to 72%</td>
<td>Intersection Delay</td>
<td>(22)</td>
</tr>
</tbody>
</table>

(Negative “% Reduction” refers to an increase in delay/travel time using adaptive control systems)

Figure 7 (a) Percentage Improvements in PI from Adaptive Control for the Intersection for Route 23 and LaRue Road (b) Summary of Adaptive Signal Control Strategy Performance