Scalable Coordinated Intelligent Traffic Light Controller for Heterogeneous Traffic Scenarios Using UPPAAL STRATEGO

Thamilselvam B, Subrahmanyam Kalyanasundaram, and M V Panduranga Rao
Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad
India 502285.
Email: {cs17resch11005,subruk,mvp}@iith.ac.in

***Abstract*—We propose a new approach for coordinating traffic flows in large cities that helps in reducing the travel time and carbon emissions from vehicles. We use the UPPAAL STRATEGO tool chain that leverages statistical model checking and machine learning for synthesizing optimal traffic coordination strategies. Our approach employs a hierarchical view of the city with two levels — individual traffic intersections and area controllers. While the choice of a phase at an intersection is decided locally, the phase threshold is decided at the level of an area consisting of several intersections. The algorithm and models that we report in this paper are a nontrivial generalization of previous approaches that used UPPAAL STRATEGO. This generalization allows scaling to large cities with several traffic intersections, with improved results.**

We compare our approach against other techniques including fixed-time and fully-actuated controllers. Experiments show that the it performs better in terms of waiting time and carbon emissions, especially in scenarios of changing traffic loads. Our approach also reduces overall and individual delays at intersections.

***Index Terms*—traffic light controller, model checking, hierarchical controller, UPPAAL STRATEGO**

I. Introduction

Traffic management in large cities is a universal concern. Good coordination between traffic intersections, resulting in better traffic management, has several benefits: reduced individual time, higher throughput and reduced emissions. For example, if most vehicles encounter a "green wave"–a sequence of green signals during their journey–so that they do not have to stop at any intersection, we would achieve many of these objectives. The problem is exacerbated in heterogeneous traffic scenarios. Accounting for vehicles with different acceleration, speed profiles and space requirements poses unique modeling challenges.

Due to the obvious benefits, this problem has attracted a lot of research attention in the past few decades. Several approaches have been suggested from various standpoints. While many earlier approaches modelled it as an optimization problem, recent ones have also used ideas in machine learning with varying degrees of success. Of the several approaches that have been reported in scientific literature, some have found their way to implementation and deployment–examples being the Split Cycle and Offset Optimisation Technique (SCOOT) [1] and the Sydney Coordinated Adaptive Traffic System (SCAT) [2].

The possibility of using formal methods for traffic management has been explored in the past, albeit for severely restricted scenarios. Eriksen et al. [3] used statistical model checking and reinforcement learning (through the UPPAAL STRATEGO tool) for reducing waiting times at a single intersection with homogeneous traffic. Thamilselvam et al. [4] extended it to coordination between two intersections with homogeneous traffic. The idea behind the coordination was a threshold based adjustment of the maximum phase duration and cycle length at the two intersections. The *phase threshold*, defined as the maximum time for which the same phase can continue at an intersection, was found by trial and error based on traffic volumes at the two intersections.

For a large city with several intersections, finding such thresholds by trial and error does not scale well. But if we make the model to support coordination (to find the phase thresholds for intersections in a neighborhood) and optimisation (to optimise the flow in an isolated intersection) at two levels, scaling to large cities is possible. The present work opens up the use of formal methods in traffic management beyond toy scenarios. While it remains to be seen how it compares with various optimization approaches (cf [5]–[7]), in this paper we show superiority of this approach over some simple yet widely used traffic controllers. Our proposed approach involves coordination (phase threshold finding) and optimization (optimizing the flow locally at every intersection) at the intersection level and the area level. In addition to this main methodological innovation, we also incorporate heterogeneous traffic scenarios where the model is not agnostic of the type of a vehicle. We also show that our method works well when the traffic fluctuates rapidly, since we adaptively change the phase threshold over an area.

As in the works mentioned above, we use the UPPAAL STRATEGO tool chain that combines statistical model check-

This work was supported by the project "M2Smart: Smart Cities for Emerging Countries based on Sensing, Network and Big Data Analysis of Multimodal Regional Transport System", JST /JICA SATREPS, Japan.

ing for analysis of stochastic systems and reinforcement learning for strategy optimization, in conjunction with the urban traffic simulator SUMO. However, the generalization necessitates a nontrivial algorithmic and modeling improvement. We give evidence that the approach yields a considerable reduction in waiting time, queue lengths, carbon emissions as well as the overall vehicle delay and delay at intersections.

This paper is arranged as follows. The next section briefly discusses previous work in the area of coordinated traffic controllers and some preliminaries needed for this paper. Section III describes our approach including the modeling, analysis and implementation details. Section IV discusses the experiment setup and results. We conclude with a brief summary of our work and future directions in Section V.

II. Related Work

We begin this section by introducing some terminology that is used in the rest of the paper.

- Phase of an intersection: A conflict-free assignment to lights in all directions at an intersection that allows flow in at least one direction.
- Phase duration: The time interval for which the phase of an intersection remains unchanged.
- Phase Threshold: The maximum time for which a phase can last at an intersection. In general, this should be periodically reassessed for an area.
- Cycle: comprises of a sequence of phases which repeats periodically at an intersection.
- Cycle time: the duration for one cycle. That is, sum of all phase durations.
- Offset: The time between the starting of green phase of two consecutive intersections with a reference point.

We now discuss briefly some existing literature addressing the problem of traffic management. Ye, Wu, and Mao [8] developed a novel method that handles large road networks by classifying the principal arterial roads based on their priority. The idea behind their coordination approach is to transform the problem of coordinating traffic lights into coordinating the principle major roads and isolated intersections. The paper dealt with only homogeneous traffic demand and did not include uncertainty in traffic flow.

One of the popular ideas in traffic management is to maintain green waves on arterial roads. Zhao-Meng, Xiao-Ming, and Wen-Xiang [9] proposed the integration of traffic-actuated control and variable cycle time of green waves. This accommodates dynamic fluctuation of traffic flow rate in a particular direction or phase. Green phase is calculated based on the vehicles arriving and waiting in each intersection. They used two types of algorithms called green early-start and green late-start to handle different traffic scenarios such as low or high traffic demand. They showed an improvement of expansion of green wave bandwidth and reduction in length of queue.

Recent studies show the advantage of using reinforcement learning to solve complex optimization problems like traffic phase calculations. Some distributed constraint optimization approaches have also been used in the past. Pham et al. [10] provide a comparison of these two techniques and suggest algorithms that were tested in real-world situations.

Van der Pol and Oliehoek [11] formulate the problem as a Markov Decision Process and use transfer learning and Max-Plus coordination algorithm of Kok and Vlassis [12] to scale to large road networks. Later, van der Pol [13] showed that a new reward function in the multi-agent deep reinforcement learning paradigm gives a considerable reduction in travel time in comparison to their previous work. Medina and Benekohal [14] model an isolated intersection as a single agent. The coordination mechanism of the Max-Plus algorithm is combined with reinforcement learning to take the traffic signal decision among neighbouring intersections.

Hierarchical control techniques have been used in the past to improve and decide the traffic signal timing by dividing geographic regions into sub-regions and improving the traffic flow characteristics, see for instance [15]. They divide the sub-regions by characterizing the traffic demands between intersections. Average travel time and cycle length can be optimized using model-based algorithms and Q-Learning. Wang et al. [15] show that the performance of algorithm works better in medium and heavy traffic flow but does not work well for low traffic flows.

We mentioned earlier about SCOOT and SCAT as examples of systems already having been deployed. SCOOT is used to optimise signal timing by adjusting three optimisation procedures such as Offset optimiser, Cycle time optimiser and Split optimiser based on the sensor data collected. It is shown that it reduces an average delay by 20% against fixed-time controller [16]. SCAT [2] supports two levels of hierarchy of traffic light control system, the *local* and *master* units. The local unit is situated at the roadside and collects detector data and processes the traffic information. The master unit is remotely located, and controls the local units at every cycle. It does not optimise cycle length. Instead it adjusts the signal timing based on changes in traffic flow by acting as a heuristic feedback system.

A. Preliminaries

This work uses tools from statistical model checking and traffic simulators, which we introduce briefly. The interested reader is encouraged to read the references provided.

Model checking is a verification paradigm to algorithmically check if a system described in a mathematically precise formalism, say a timed automaton [17], satisfies a property expressed in a precise language, say TCTL [18], [19]. Asarin et al. [20] and Maler et al. [21] proposed ideas to *synthesize* models that satisfy certain requirements, called controllers[1]. The synthesized controllers would typically be in terms of nondeterministic or stochastic choices of actions allowed at each state in the automaton. While these controllers would satisfy the given property, it remained to choose a controller

[1]This controller is different from the traffic controllers that we discuss later. The difference will be clear from the context.

Scenarios	Fixed-Time Controller						Actuated Controller					
	Green Time Direction				Yellow	Cycle Length	Green Time Direction				Yellow	
	N	E	S	W			N	E	S	W		
Low	25	20	25	20	4×5	110	30	25	30	25	4×5	
Med	35	25	35	25	4×5	120	40	30	40	30	4×5	
High	45	35	45	35	4×5	160	55	45	55	45	4×5	

TABLE I
GREEN TIME FOR FIXED-TIME AND ACTUATED CONTROLLERS

that satisfied additional requirements like those of performance. The UPPAAL STRATEGO tool chain implements these. While there exist several model checking tools available that are appropriate to different settings, we briefly discuss the UPPAAL tool suite, which we use in this paper.

UPPAAL STRATEGO: UPPAAL is a tool for model checking timed automata against properties expressed in TCTL [22]. UPPAAL-SMC was added to this tool-suite David et al. [23] for model-checking stochastic priced timed automata. However, these were restricted to checking if a property was satisfied by the system or not. UPPAAL-TIGA implements model synthesis ideas [24]–models that satisfy functional specifications can be synthesized. Using ideas from David et al. [25], [26], and the previously mentioned tools in the chain, David et al. [27] reported the addition UPPAAL STRATEGO to the suite, which (i) uses reinforcement learning to take non-deterministic strategies as input and give out a deterministic strategy that is optimized for some objective functions, and (ii) uses (statistical) model checking to check other properties.

This tool chain has been used to analyze several systems in the past. For example, it was used to learn strategies for controlling the floor heating in a home [28], which resulted in a significant reduction of a "comfort penalty", the distance between desired and simulated temperatures.

More relevantly to this work, UPPAAL STRATEGO was used to find the next phase information in controlling the traffic light at a single intersection based on traffic volume [3]. It was subsequently improved to two intersections with homogeneous traffic by Thamilselvam et al. [4].

SUMO: Simulation of Urban MObility (SUMO) [29] is an open source, microscopic road traffic simulation package. SUMO supports the modelling and measurement of various road network parameters like traffic demand, vehicle types, emission etc. We use TRAffic Control Interface (Traci) to control SUMO simulator. Traci [30] is a Python package which supports to interact and retrieve all objects involved in SUMO.

SUMO also supports measurement and monitoring of a large number of traffic parameters including pollutant emissions of vehicles, and details of each vehicle's journey. Additionally, it allows simulation of various detectors and detector outputs–the lane area detector and loop detectors. Indeed, we use area and loop detectors for collecting the arrival and waiting vehicles list to improve the controller as shown in Fig 1.

For our intersections, we consider four phases–each phase allows the left, right turn and straight of corresponding lane. We do not consider pedestrian phase in this model. Our

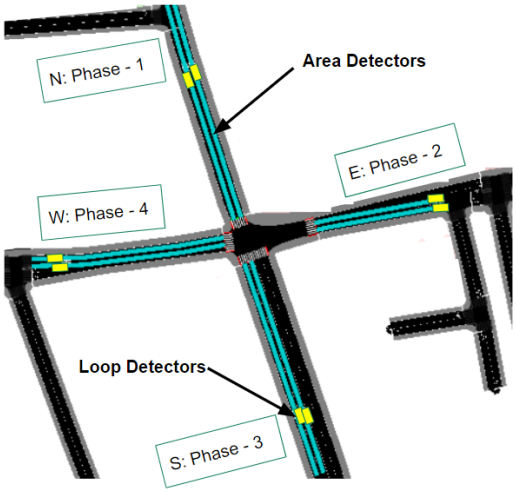

Fig. 1. Area and Loop Detectors at Intersection

techniques easily generalize to other polyphasic traffic models. There are two important devices that detect vehicles on the street: a) Loop detectors, which count the number of vehicles crossing the detector in unit time and b) Area detectors, which detect the waiting and arriving vehicles, along its length, which is typically 300 meters.

Using these devices, some straightforward techniques have been proposed for traffic light controllers in the past.

In this paper, we do a comparison of five types of controllers, including ours:

- **The Fixed-Time Controller**: Here, the controller has fixed cycle length and phase duration. SUMO supports xml file (named tll.xml) for encoding the phase information for different traffic loads. We consider Phase - 1 as N, Phase - 2 as E, Phase - 3 as S and Phase - 4 as W as shown in Figure 1. The phase and cycle length details for a given direction in table I.
- **The Actuated Controller** uses the induction loops and area detectors for detecting the vehicles. Loop detectors are fixed in the range of 70 meters to 80 meters from the intersections in all roads. Area detectors length of 10 meters are located 1 meter away from the intersection to detect the presence of vehicles nearby. The minimum green green time for the phase is 8 seconds. The controller works as follows in all intersections:
  - The controller normally goes green in N directions always if no vehicles present in all other directions.
  - If vehicles are detected from loop and area detectors in any directions, green phase gets extended for 4 seconds in relevant directions until it reaches its maximum green time. The maximum green time for all directions is in table I
  - The direction which triggers the current green time to extend will be served as next phase. If there are more than one directions trigger,then FCFS (First Come First Service) will be applied.

- **UPPAAL STRATEGO based Single Intersection Controller (USSIC) of Eriksen et al [3]**: We consider all vehicle types are equal and split phase into 4 phases without changing the model mentioned in their work. Area detector are fixed in all roads.
- We call the controller that we design in this paper, the **coordinated intelligent** controller. We discuss this in the next section.
- **UnCoordinated Intelligent Controller (UCI)**: Finally, we use our model without coordination among intersections, we call it as an UnCoordinated Intelligent Controller. Considering this variant allows us to see the advantage in having an additional level of area coordination.

III. Our Approach

We begin by giving an informal but intuitive description of our approach. The main insight behind our approach is as follows. Traffic light management boils down to chiefly to two activities:

- Setting the phase threshold for intersections and
- Deciding whether or not to increase a phase at an intersection.

Naturally, the phase threshold has to be adaptive to the changes in the traffic patterns in an area. Therefore, this threshold is updated periodically by UPPAAL STRATEGO using an *Area Level Timed Automata Network* (ALTAN), and data from a neighborhood of intersections. We will refer to the duration between two consecutive updates as an *epoch*.

For every intersection, a decision whether to continue with the current phase or change to another phase is taken by UPPAAL STRATEGO using the *Intersection Level Timed Automata Network* (ILTAN) model and data and parameters pertaining to that intersection. Continuation of the same phase cannot extend beyond the phase threshold in place for the current epoch.

We now describe the architecture of the solution, before moving on to the algorithm itself.

1) Traffic data (from area and loop detectors) is first acquired from SUMO through the Traci interface.
2) This data is updated in the timed game automata model.
3) UPPAAL STRATEGO synthesizes the strategy using the updated model and query file.
4) The current phase information from synthesized strategy is passed to Traci, which then updates the SUMO traffic light phases.

Indeed, this architecture is followed in earlier works [3], [4] as well.

The algorithm, listed in Algorithm 1, works by invoking the ILTAN and ALTAN. Since ILTAN is about tactical decision making as to whether or not to extend the current phase at an intersection, it is invoked more frequently (e.g every 10 seconds). On the other hand, ALTAN concerns a strategic decision regarding the phase threshold for an epoch, it is invoked less frequently. In both cases, the strategy suggested by UPPAAL STRATEGO is conveyed to SUMO through the Traci interface. If only ILTAN is used, and the phase threshold is set manually, we get the *UnCoordinated Intelligent Controller*, or the **UCI**.

Algorithm 1: High level algorithm for the Coordinated Intelligent controller

Input: Vehicle waiting count with type, queue length and phase from SUMO
Output: Next signal phase and updated *phase threshold* from UPPAAL STRATEGO

```
Simulation Step ← 0
for All intersections do
    ExtendGreen← FALSE
end for
while Simulation Step < Maximum Simulation Step do
    Read detector data from SUMO
    ILTAN: At every 5th Simulation Step
    for All intersections do
        if ExtendGreen=TRUE then
            Run Extend Green Model–To find whether to
            extend the current green phase or go to yellow
            phase (Figure 2)
            //Updates current phase and current green time
        else
            Run Yellow Model (Figure 3)–To find which
            directions green phase has to be given
            //Resets current green time
        end if
    end for
    ALTAN: At randomly chosen time steps in the
    simulation:
    for All Areas do
        Run the Area models in Figures 5 6, 7–Coordinate
        among all traffic lights within the area.
        //Updates phase threshold of phases in all directions
        for the next epoch
    end for
    Simulation step ++
end while
```

Now, we discuss each Timed Automata Network in detail.

A. The Timed Game Automata Model

1) the ILTAN: Each intersection is modeled by two automata:
   - the *Extend Green* automaton (Fig 2): This model is used to find whether the current phase at the intersection needs to be extended or changed. UPPAAL STRATEGO arrives at this decision through simulating this model and reinforcement learning. The function *Initialize()* updates the direction-wise waiting vehicle count (Bus, Car, 3-Wheeler, 2-Wheeler), queue length and all the phases in the model while moving to the *commit* location shown at the center. From that location, it has five lo-

cations to explore. While these choices are non-deterministic in the timed game automaton, UPPAAL STRATEGO combines statistical sampling and reinforcement learning to choose simulation paths. When it takes *greenAgain*, then it will continue flow of vehicles in the current phase (say northward), provided that the current phase green time has not exceeded the phase threshold of that epoch. Similarly, when it takes *goYellow_i*, for some $1 \leq i \leq 3$, then it will allow flows of vehicle in phase i. The function *flow(.,.)* takes as input two parameters, delay–how much time the traffic flow should be, and phase–which directions the traffic flow should move and calculates the flow of the traffic. Once it reaches End location, it outputs the best choice of phase.
The timer variable `MAX_GREEN` carries the phase threshold and is updated by ALTAN.

- The *Yellow* automaton (Fig 3): When the green phase in a direction is not extended, the intersection enters the yellow phase. At this point, a green phase has to be chosen for one among the four directions, including the one that just relinquished the green phase. UPPAAL STRATEGO simulates and learns the best strategy to decide this. Whenever a direction is chosen for green phase, the remaining directions will not be given green for a threshold horizon.

2) The ALTAN: As mentioned earlier, the area controller combines some "adjacent" intersections into an "area". Figure 4 shows four areas, for example. The Area Controller is modeled by three automata:

- The *Polling* automaton (Figure 5) polls all the intersections in the same area one by one and the next automaton (Figure 6) is invoked to run for a duration of a time horizon H. This is repeated until an *end horizon EH* is reached, at which time, the simulation ends.
- Each intersection in the area is modeled by an instance of the automaton shown in Figure 6 and is identified by a unique id. The function $Initialize()$ initializes all the vehicles count for this intersection. The automaton synchronizes with the automaton given in Figure 7, which in turn provides the maximum green times in a direction. At the central *committed* location (labelled by C in Fig 6) in the automaton, there are six choices (locations)–four of them (labeled by N, E, W and S respectively) indicate in which direction the green phase has to be activated for this intersection in the area. The fifth location is for updating the arrival list (by invoking the $updateArrivalList()$, discussed later), the end of the current horizon and the sixth is End location.
- Maximum Green Selection (Fig 7): This automaton is used to select the maximum green time. The main area controller calls it and updates maximum green time of all neighbour junctions. Every run has different maximum green time, Area main coordination model searches the best maximum green time by simulating several time and updates the maximum green time of local controller.

We now explain coordination among the neighbours.

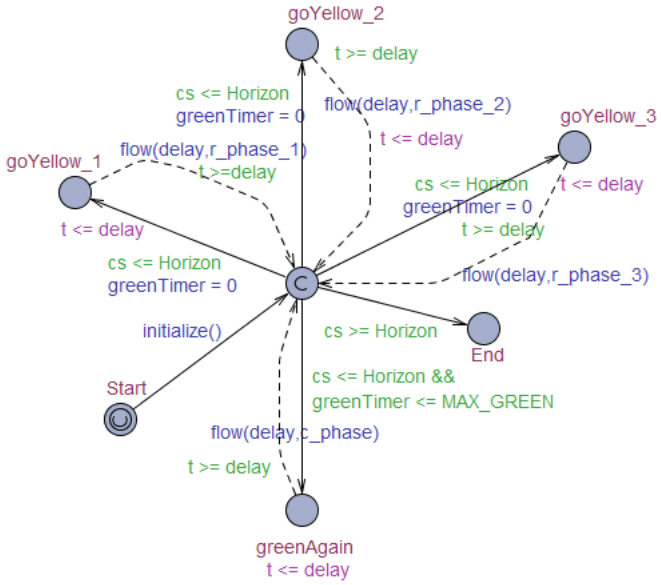

Fig. 2. Extend Green Model

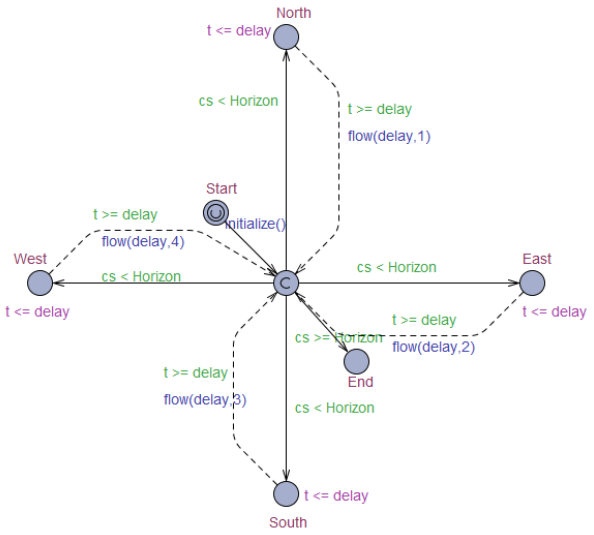

Fig. 3. Yellow Model

1) Coordination of Traffic Light in Area controller: We seek to coordinate and synchronize multiple signalised intersections to enhance throughput in one or more directions.

The update rules in the Area Controller Automata are implemented in the updateArrivalList() function. This function updates the list of vehicles arriving at a neighboring intersection, based on the current phase and the offset value to it.

We illustrate this using an example. Fig 8 shows two neighboring intersections J1 and J2. A sample run is shown

Fig. 4. City with 23 Intersections

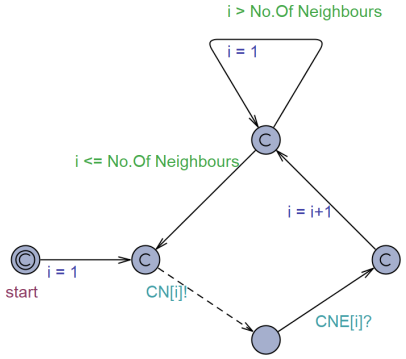

Fig. 5. The Area Polling Automaton

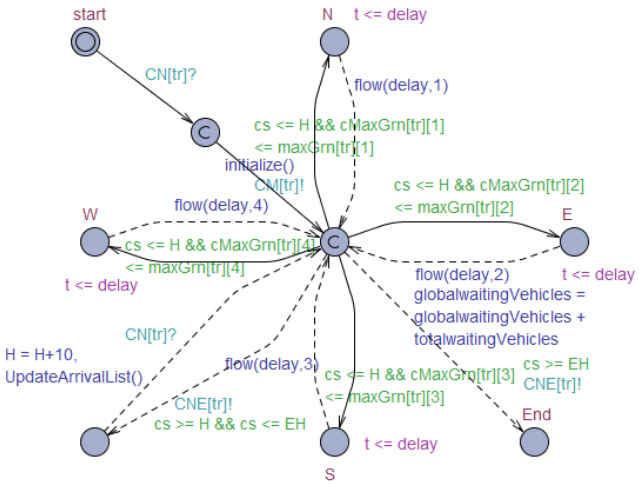

Fig. 6. Area Controller: Main Coordination Model

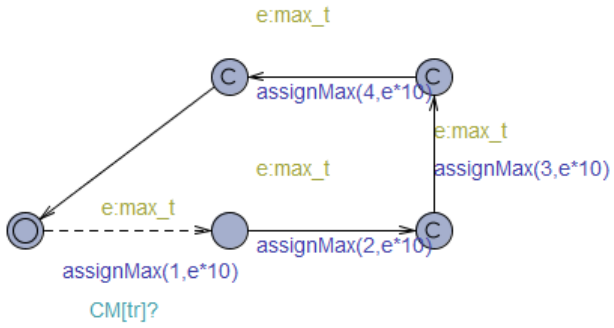

Fig. 7. Area Controller: Maximum Green Selection Model

in II. Time progresses from left to right in intervals of 5 seconds. The second and the seventh rows specify which phase is active at intersection J1 and J2 respectively. Rows 3–6 and 8-11 specify the number of vehicles waiting in the North, East, West and South directions at each junctions respectively. An increase in the number of vehicles at an intersection due to an inflow from the neighbour is showed in italics, whereas a decrease due to an outflow (when the green phase occurs in a particular direction) is shown in bold. A combination of inflow and outflow is shown in both italics and bold.

The *updateArrivalList()* function works as follows. An area controller has three main variables: the global variable - EndHorizon (the time at which an area controller automata ends), the local variables Horizon for each intersection and Current step (current time step of an intersection). An area controller polls an intersection one by one at the Horizon time interval in round robin method up to the Current step reaches the EndHorizon. The *updateArrivalList()* is called at every time when the intersection is switched over and it updates the arrival list of neighbours only if the offset(time to reach neighbour junction) value crossed the Current step.

The system will be simulated several times to choose the best combination of phase which gives the best performance, here minimum number of waiting vehicles. The best run is found by the reinforcement learning subroutine of UPPAAL STRATEGO.

UPPAAL STRATEGO finds the optimum strategy for this Area Controller, based on the traffic volume of the neighbor.

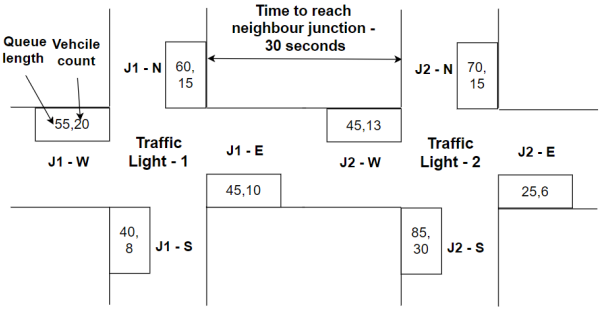

Fig. 8. Two Signalised Intersection with Detector Data

Time	0	5	10	15	20	25	30	35	40	Sum of Waiting Vehicles
J1 - Phase	S	E	S	W	W	S	W	W	W	
J1 - E	10	10	10	10	10	10	*12*	*13*	*15*	100
J1 - S	8	8	**5**	5	5	5	5	5	5	51
J1 - W	20	**17**	17	**14**	14	14	**11**	11	11	129
J1 - N	15	15	15	15	**12**	**9**	9	**6**	**3**	99
J2 - Phase	N	W	N	E	E	S	S	E	E	
J2 - E	6	**3**	3	**0**	0	0	0	0	0	12
J2 - S	30	30	30	30	**27**	**24**	24	24	**21**	240
J2 - W	13	13	13	13	13	13	***12***	***10***	*12*	112
J2 - N	15	15	**12**	12	12	12	12	12	12	114
Total Number of Waiting Vehicles										**857**

TABLE II
ONE SAMPLE RUN

Features	USSIC [3]	Coordinated UPPAAL STRATEGO [4]	Coordinated Intelligent This Paper
Heterogeneous Vehicle Type	No	No	Yes
Phases	2 - Phases	2 - Phases	4 - Phases
Number of Intersections	1	2	23
Coordination	No	Yes	Yes
Scalability	No	No	Yes
Input	Vehicle counts	Vehicle counts	Vehicle counts, queue length
Optimization Parameter	Delay	Delay	Delay, queue length
Spatial Priority	No	No	Yes

TABLE III
FEATURE COMPARISON WITH OTHER CONTROLLERS

For example, one can now ask the following query to obtain the optimum strategy. In the interest of space, we omit the syntax and semantics of the query language and refer the reader to David et al. [27].

```
strategy Opt = minE (globalWaitingVehicles)
[ ≤ NoOfNeighbors × EndHorizon ] :      <>
trafficLight.End.
```

The query seeks to minimize the expected global total waiting time of the vehicles over all strategies within the end horizon. Then we simulate the model to get the phase of traffic light using the strategy obtained by query.

We employ a coordinated intelligent traffic light controller, where all the local decisions of traffic lights are updated into common and shared variable. These variables can be accessed by neighboring lights in order to arrive at an optimized decision. Coordination among traffic lights ensures smooth traffic flow and reduces the delay and emission.

B. Comparison of features

We compare our model features with two different models designed in Eriksen et al. [3] and Thamilselvam et al. [4] in Table III.

IV. SIMULATION SETUP AND RESULT ANALYSIS

We use the SUMO simulator for real-time microscopic traffic simulation. From the OpenStreetMap (www.openstreetmap.org) database, we downloaded the Ahmedabad city road map. This is converted into the SUMO network file using the NETCONVERT application. The 23 main signalized intersections in the city of Ahmedabad, are taken into account for the experiment mentioned in Fig 4. All the other are set to be non-signalized intersections. The traffic flow values with the various origins and destinations and different vehicle type are taken from the Operations Document of Ahmedabad Traffic Management and Information Control Centre [31]. In all the SUMO simulations, we use seconds as the time unit. This can be scaled. We use this information to generate traffic demand. The discrete time interval is set to be 1 second and the whole simulation period is 1200 seconds. We measure the following parameters.

Delay: Cumulative waiting time of vehicles. If 10 vehicles are waiting for 10 seconds, then waiting time is 100 seconds.

Queue Length: Cumulative waiting vehicle length. If one 3 meter vehicle and one 5 meter vehicle are waiting in queue for 10 seconds then queue length is 80 meter seconds.

Throughput: Total number of vehicles crossed in all the 23 intersections per second.

CO and CO_2 emission: This is the cumulative value calculated from all individual vehicles at every time step and added together.

The source code used for simulation is available for download at the following URL: https://github.com/ThamilselvamB/Intelligent-Traffic-Light-Controller-using-Uppaal-Stratego.

	Acceleration (m/s^2)	Deceleration (m/s^2)	Max Speed (km/hr)	Width(m)	Length(m)	Space Headway (m)
Bus	0.5	3.5	50	2.45	10.1	1.0
Car	0.8	4.5	70	1.5	3.6	0.5
3-wheeler	0.6	3.6	55	1.25	2.6	0.3
2-wheeler	0.9	3.5	60	0.7	1.85	0.15

TABLE IV
VEHICLE PHYSICAL PROPERTIES

A. Traffic Demand Generation

We generate the traffic demand based on Poisson distribution mentioned in the following equation.

$$P(k \text{ cars in an hour}) = \frac{\lambda^k e^{-\lambda}}{k!},$$

where λ is the average number of vehicles per hour in the direction (as per data from [31]). In order to generate medium and low traffic demand we multiply λ by 60% and 30% respectively. For generating the corresponding SUMO route file, we sample by repeated Bernoulli trials.

B. Vehicle Parameters and Sublane

We consider standard values for the vehicle properties mentioned in the Table IV. We use the notion of sublanes to make the simulation more realistic. This models the situation where more than one vehicle in parallel can be situated in a single lane if its width fits. This also allows the vehicles overtake other vehicles on a single lane. We simulate the demand using the open source traffic simulator SUMO. Using this, we adjust the parameters to generate traffic demand for our scenarios.

Controllers	Delay (Hour)	Queue Length (Km)	CO (Kg)	CO2 (Kg)	Throughput (Veh / Sec)
Fixed-Time	891.2	10.1	267.7	4576.6	7.6
Actuated	835.1	9.5	242.6	4019.6	7.2
USSIC	722.0	9.2	223.3	3840.7	8.17
UCI	623.0	9.8	192.7	3408.7	8.24
Coordinated	612.3	9.6	189.7	3356.4	8.3

TABLE V
RESULTS FOR CONSTANT HIGH TRAFFIC

Controllers	Delay (Hour)	Queue Length (Km)	CO (Kg)	CO2 (Kg)	Throughput (Veh / Sec)
Fixed-Time	583.8	6.63	175.7	3054.8	6.11
Actuated	560.5	6.24	179.7	2982.2	6.1
USSIC	414.2	4.81	125.6	2288.9	7.12
UCI	358.5	5.43	113.6	1944.2	7.35
Coordinated	355.9	5.48	109.2	2034.1	7.45

TABLE VI
RESULTS FOR VARYING TRAFFIC DEMAND

C. Traffic Load Scenarios

Different traffic demands are generated based on origin and destination points of Ahmedabad from [31]. We consider three cases of load scenarios – High, Medium and Low traffic demands. High traffic has 100% of traffic demand mentioned in Ahmedabad city document, Medium has 60% and Low traffic has 30% of the traffic demand. We consider four types of vehicles, passenger bus, car, 3-wheeler and 2-wheeler. The distribution of these types are also according to the distribution occurring in Ahmedabad city, as provided in [31].

D. Results

We report results for two scenarios. In both scenarios, unsurprisingly, the uncoordinated and cooordinated approaches proposed in this paper outperform traditional Fixed Time and Actuated strategies. The first experiment concerns the high traffic scenario for 1200 seconds. The results are shown in Table V. The advantage of using a two level hierarchy for local and area-wise decision making is evident on all parameters. The waiting time for each intersection with the five different controllers is shown in Fig 9. The scatter plot for the vehicle trip delay is given in Fig 11. The cumulative delays for the five different controllers at every 2 seconds are plotted in Fig 10.

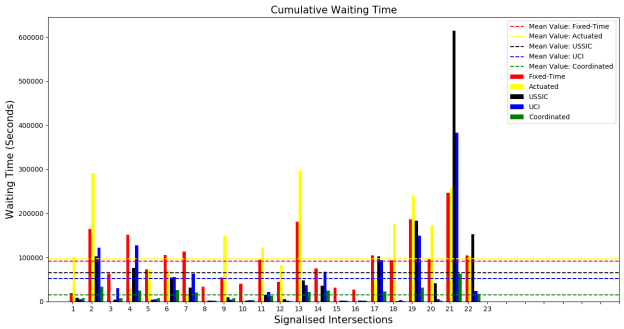

Fig. 9. Delay at the intersections with five different controllers and their mean delay.

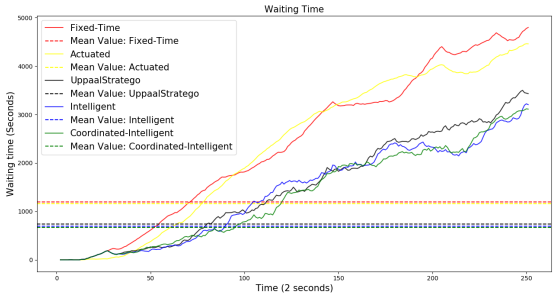

Fig. 10. Cumulative waiting time of vehicle with time.

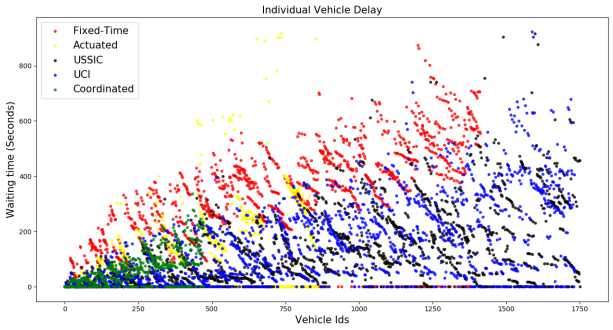

Fig. 11. Individual vehicle waiting time.

We see that in all the cases, there is a reduction in emissions, waiting times and queue lengths. Since the area controllers update local coordinated controllers' phase threshold dynamically, the decision taken by the the coordinated controller will ensure that the traffic flow is smooth. Therefore, we observe a reduction in emissions.

The second experiment is performed with a varying traffic load starting from low traffic, increasing through medium to high, and returning to low at intervals of 200 seconds. Results for this "triangle" shaped traffic load is shown in Table VI. It can be seen that there is an improvement over and above the UCI for the coordinated approach. This is in spite of carefully choosing a phase threshold for the UCI approach. The increase in the queue length in both UCI and coordinated approach can be explained by the heterogeneity of the vehicles considered in these two models.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a solution for the traffic management problem that scales to two levels of hierarchy. We leave the question open whether a multi-level generalization of our technique helps in achieving better accuracy or scalability in terms of the sizes of cities that can be handled. Further generalizations would incorporate pedestrian traffic and driver characteristics.

REFERENCES

[1] D. I. Robertson and R. D. Bretherton, "Optimizing networks of traffic signals in real time-the scoot method," *IEEE Transactions on vehicular technology*, vol. 40, no. 1, pp. 11–15, 1991.

[2] P. Lowrie, "Scats-a traffic responsive method of controlling urban traffic," *Sales information brochure published by Roads & Traffic Authority, Sydney, Australia*, 1990.

[3] A. B. Eriksen, C. Huang, J. Kildebogaard, H. Lahrmann, K. G. Larsen, M. Muniz, and J. H. Taankvist, "Uppaal stratego for intelligent traffic lights," in *12th ITS European Congress*, 2017.

[4] B. Thamilselvam, S. Kalyanasundaram, and M. P. Rao, "Coordinated intelligent traffic lights using uppaal stratego," in *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*. IEEE, 2019, pp. 789–794.

[5] T. Le, P. Kovács, N. Walton, H. L. Vu, L. L. Andrew, and S. S. Hoogendoorn, "Decentralized signal control for urban road networks," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 431 – 450, 2015, special Issue: Advanced Road Traffic Control.

[6] T. Le, H. L. Vu, N. Walton, S. P. Hoogendoorn, P. Kovács, and R. N. Queija, "Utility optimization framework for a distributed traffic control of urban road networks," *Transportation Research Part B: Methodological*, vol. 105, pp. 539 – 558, 2017.

[7] P. Kovács, G. Raina, and N. Walton, "Ramp and signal control: Where motorways and urban roads meet," in *2015 7th International Conference on Communication Systems and Networks (COMSNETS)*, 2015, pp. 1–6.

[8] B.-L. Ye, W. Wu, and W. Mao, "A method for signal coordination in large-scale urban road networks," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[9] C. Zhao-Meng, L. Xiao-Ming, and W. Wen-Xiang, "Optimization method of intersection signal coordinated control based on vehicle actuated model," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[10] T. T. Pham, T. Brys, M. E. Taylor, T. Brys, M. M. Drugan, P. Bosman, M.-D. Cock, C. Lazar, L. Demarchi, D. Steenhoff *et al.*, "Learning coordinated traffic light control," in *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS-13)*, vol. 10. IEEE, 2013, pp. 1196–1201.

[11] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*, 2016.

[12] J. R. Kok and N. Vlassis, "Using the max-plus algorithm for multiagent decision making in coordination graphs," in *Robot Soccer World Cup*. Springer, 2005, pp. 1–12.

[13] E. van der Pol, "Deep reinforcement learning for coordination in traffic light control," *Master's thesis, University of Amsterdam*, 2016.

[14] J. C. Medina and R. F. Benekohal, "Traffic signal control using reinforcement learning and the max-plus algorithm as a coordinating strategy," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2012, pp. 596–601.

[15] Y. Wang, X. Yang, Y. Liu, and H. Liang, "Evaluation and application of urban traffic signal optimizing control strategy based on reinforcement learning," *Journal of Advanced Transportation*, vol. 2018, 2018.

[16] S. A. Leaflet, "1: The" scoot" urban traffic control system," *available at scootutc. com/documents/1_SCOOT-UTC. pdf*, 1995.

[17] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[18] R. Alur, C. Courcoubetis, and D. Dill, "Model-checking for real-time systems," in *[1990] Proceedings. Fifth Annual IEEE Symposium on Logic in Computer Science*. IEEE, 1990, pp. 414–425.

[19] ——, "Model-checking in dense real-time," *Information and computation*, vol. 104, no. 1, pp. 2–34, 1993.

[20] E. Asarin, O. Maler, and A. Pnueli, "Symbolic controller synthesis for discrete and timed systems," in *Hybrid Systems II*, P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 1–20.

[21] O. Maler, A. Pnueli, and J. Sifakis, "On the synthesis of discrete controllers for timed systems," in *STACS 95*, E. W. Mayr and C. Puech, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 229–242.

[22] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on uppaal," in *Formal methods for the design of real-time systems*. Springer, 2004, pp. 200–236.

[23] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, "Uppaal smc tutorial," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 397–415, 2015.

[24] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime, "Uppaal tiga user-manual," *Aalborg University*, 2007.

[25] A. David, H. Fang, K. G. Larsen, and Z. Zhang, "Verification and performance evaluation of timed game strategies," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2014, pp. 100–114.

[26] A. David, P. G. Jensen, K. G. Larsen, A. Legay, D. Lime, M. G. Sørensen, and J. H. Taankvist, "On time with minimal expected cost!" in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2014, pp. 129–145.

[27] A. David, P. G. Jensen, K. G. Larsen, M. Mikučionis, and J. H. Taankvist, "Uppaal stratego," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2015, pp. 206–211.

[28] K. G. Larsen, M. Mikučionis, M. Muniz, J. Srba, and J. H. Taankvist, "Online and compositional learning of controllers with application to floor heating," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2016, pp. 244–259.

[29] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, November 2018.

[30] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "Traci: An interface for coupling road traffic and network simulators," in *Proceedings of the 11th Communications and Networking Simulation Symposium*, ser. CNS '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 155–163.

[31] "Ahmedabad Traffic Management and Information Control Centre operation document, ahmedabad tmicc," https://smartnet.niua.org/sites/default/files/resources/ahmedabad_tmicc.pdf, accessed: 2020-04-21.