Available online at www.sciencedirect.com

ELSEVIER

Procedia - Social and Behavioral Sciences 00 (2013) 000–000

Procedia
Social and Behavioral Sciences

www.elsevier.com/locate/procedia

Conference on Agent-Based Modeling in Transportation Planning and Operations

A Simple, Naive Agent-based Model for the Optimization of a System of Traffic Lights: Insights from an Exploratory Experiment

Tong Pham*, Aly Tawfik[a], Matthew E. Taylor[b]

California State University, Fresno, CA 93740, USA
Washington State University School of Electrical Engineering and Computer Science, WA 99163, USA

Abstract

In this paper, two basic Distributed Coordination of Exploration and Exploitation algorithms, Static Estimation Optimistic K1 and Static Estimation Optimistic K2, were implemented and tested against standard isolated traffic actuated signals through simulation on a microscopic traffic simulator. The simple pilot exploratory experiment demonstrates how Naïve, Optimistic agent-based models were able to improve the performance of a system of traffic lights over time. Results also show the importance of a few variables on the system performance; namely, signal operation scheme, cool down time, and evaluation interval.

Keywords: agent, distributed, traffic signals, optimization, DCEE

1. Introduction

Climate change, the peaking of oil and limited funds are among the biggest challenges of the twenty first century. It is clear that traffic management contributes to all three of these challenges. Accordingly, this work attempts to improve the performance of a system of traffic lights using an agent-based methodology, the Distributed Coordination of Exploration and Exploitation (DCEE). Attempts to optimize a system of traffic lights using agent-based models are not new (McKenny & White, 2013), (Abdoos, Mozayani, & Bazzan, 2013), (Y., Quek, & Loh, 2009). However, none of these attempts is based on the DCEE algorithm, which is the contribution of this work. In the following sections, the authors present the objectives of the paper, followed by a short

* Corresponding author. Tel.: +1-714-234-8563.
E-mail address: me@tongpham.com

description of the traffic network setup, DCEE framework, and the experimental setup. Then, the authors present a summary of the results and conclusions.

2. Acknowledgements

We would like to thank the team at University of Texas at Austin's Learning Agents Research Group for their assistance with the modification of the AIM traffic simulator.

3. Literature Review

3.1. Existing adaptive signal control systems

The 1970s saw the development of SCOOT (Split, Cycle and Offset Optimization Technique) in the UK (Robertson & Bretherton, 1991). This system features a central computer system to monitor a series of intersections, attempting to minimize the sum of the average queues and number of vehicle stops. Notably, SCOOT makes use of a model of the traffic flow based on Cyclic Flow Profiles (average one-way flow of vehicles past a fixed point on the road) measured real-time using sensors. While the system has been observed to register a 12 percent improvement over fixed-time systems, it is not readily amendable to scaling due to the need for centralized control.

The SCAT system, which stands for ``Sydney Coordinate Adaptive Traffic'', was introduced in the 1980s to combat congestion in downtown Sydney (Sims & Dobinson, 1980). The system comprises one central supervisory computer at the control center, 11 remote region computers, and over 1000 microcomputer traffic signal controllers over the Sydney metropolitan area. In particular, the intersection computer is responsible for tactical decision on signal operation and detection of hardware failure. The regional computer each controls up to 200 sets of signals, implementing real-time operation of said signals by analysis of the detector information provided by the intersection computers. Finally, the supervisory computer allows monitoring of all subsystems below it and manual override. Interestingly, the control hierarchy can go beyond hardware limitation and be organic: subsystems start out as small as individual intersections, but combine with adjacent subsystems to form larger subsystems/one large system. The data for each subsystem specify minimum, maximum, and optimum cycle length. Four background plans are stored in the database for each subsystem; a number of detectors at key intersections are identified as strategic detectors. Various system factors are calculated from the strategic detector data to decide whether the current cycle and plan should change.

There also exist adaptive systems such as RHODES (Mirchandani & Wang, 2005), which features more involved sensor systems, models, allowing them to do away with explicit cycle length definitions. Basically, the system carries out two main processes. The first is estimation and prediction, which takes the sensor data and estimates the actual flow profiles in the network and the flows' subsequent propagation. The second process involves the decision system, which selects the phase timing to optimize a given objective function, such as minimizing average travel delay. This decision system is hierarchical. At the highest level is a *dynamic network loading* model that captures traffic characteristics that vary slowly over time: travel demand, availability of roads in the networks, etc. On the basis of these characteristics, the system can estimate the load on each particular road segment, in terms of vehicles per hour. These estimates provide RHODES with prior allocations of green times (times when the traffic signals are green) for each different demand pattern and each phase. At the middle level, called network flow control, the system updates the green-time decisions. At this level, the system measures traffic flow characteristics in terms of platoons of vehicles and their speeds. Given the approximate green times, the intersection control at the lowest level selects the appropriate times for phase changes. It does this on the basis of observed and predicted arrivals of individual vehicles at each intersection. Despite this sophistication, the

system still requires heavy use of models, which can take time to perfect, as well as requiring the setup of the hierarchy.

3.2. Related computational solutions

In (Junges & Bazzan, 2008), the authors used three complete Distributed Constraint Optimization (DCOP) algorithms to study their performance in improving traffic signals. The interest here is to optimize the restrictions, not just satisfying them. ADOPT, one of the algorithms used, performs a distributed search using cost communication as a guide for the agents to choose the optimum values for its variable. OptAPO uses direct constraint communication as a form of partially centralize the problem, through a mediator, which uses a centralized optimization in order to find an optimum solution to its portion of the problem. DPOP is based on dynamic programming, propagating utility in a tree-like network of agents. It is outside the scope of this paper to delve into the intricate details of these algorithms, and how they were implemented in this study.

For this particular problem domain, the authors have formulated it as a tuple $< A, D, F >$ where:

- $A = \{a_1, ..., a_n\}$ is the set of variables/agents, where n is the number of intersections;
- $D = \{d_1, ..., d_n\}$ is the domain of the variables, representing the possible intersections;
- $F = \{f_{1,1}, ..., f_{n,n}\}$ is the set of constraints among the variables, where f_i denotes the associated cost; each constraint has a cost for a given pair of values of the two variables.

Note that these constraints are binary among the agents/traffic lights.

The cost function associated with a constraint between two neighboring intersections i and j is $f_{i,j} = \beta_{i,j} \times \tau \times \gamma$ (Junges & Bazzan, 2008), where:

- $\rho_{i,j}$ is the density of vehicles in the lane $i \rightarrow j$
- $\beta_{i,j}$ represents the fraction of traffic at intersection j coming from direction i : $\beta_{i,j} = \frac{\rho_{i,j}}{\sum_z \rho_{z,i}}$
- γ expresses the degree two consecutive agents agree (are synchronized), which can be 1.5 when synchronized and 2 otherwise
- τ expresses the degree agents are coping with the volume of vehicles

Table 1 Synchronization parameter in DCOP approach

Plan run by agent i	**Plan run by agent** j	τ
+	+	0
-	+	1
+	-	1.5
-	-	2

In **Table 1**, + means that plan is synchronized and agrees with the direction of higher traffic volume; – means that plan is synchronized in a direction other than that of higher traffic volume.

Whilst the authors reported positive results, it is certainly intriguing as to how β can be computed in practice. Note that $\beta_{i,j}$ requires knowing the load of vehicles across all z , which varies greatly. The DCOP framework assumes that $f_{i,j}$ is known beforehand, and thus it is unclear how the authors sample the function space to allow this to hold.

4. The DCEE framework

4.1. Motivation

The main motivation to use the DCEE for this problem domain of optimizing traffic light signal is twofold. One, there has been no application of said framework to this domain before, and there is interest in contributing to the growing set of solutions that tackle this domain. Two, the DCEE framework possesses several relevant characteristics appropriate to this problem domain (Taylor, Jain, Tandon, Yokoo, & Tambe, 2011):

- Agents do not know their reward functions. In the context of optimizing traffic lights, one can let agents be the intersection controllers. As traffic patterns not only vary throughout the day, but also by season, special occasions, accidents... it is certainly the case that the desired outcome (traffic flow efficiency) cannot be determined by static functions mapping the traffic light configurations to real-time traffic outcome.
- Agents can execute a fixed number of actions per trial, for a finite number of trials, to maximize the on-line reward. Traffic signals optimization process can be done continually every day, but traffic light signaling must take into account the fact that human drivers need a minimum duration of green/yellow/red to react. This means that any changes that hope to improve the traffic condition cannot happen too rapidly, but rather can be thought of as episodic or at least with some sizeable delay between change. Also, the goal is not to achieve a theoretical upper bound of traffic performance, but rather to improve it as much as possible. Thus, the framework will realistically get the agents to maximize the expected reward over time.
- Agents cannot explore the entire cross-product of possible actions among themselves. The agents in a traffic signal network cannot hope to explore all possible coordination and control among them, owning to the fact that the environment is continuous.

Among previous works making use of this framework is one optimizing wireless signal transmission among physical robots forming an ad-hoc network (Taylor, Jain, Tandon, Yokoo, & Tambe, 2011). Thus there is practical promise for the framework, and it is only logical that we proceed to investigate its effectiveness on a new problem domain.

4.2. Formal Definition

Formally, a DCEE problem (Taylor, Jain, Tandon, Yokoo, & Tambe, 2011) consists of:

- a set of variables, $V = \{x_1, x_2, ..., x_n\}$, where $x_{i,l} \in D_i$;
- a set of agents, each controlling a variable from V (in the general case, one agent could control multiple variables);
- an (initially unknown) reward function $f_{ij} : D_i \times D_j \rightarrow R$, which gives the cost of a binary constraint $(x_i \leftarrow d_i, x_j \leftarrow d_j)$, where $d_i \in D_i, d_j \in D_j$;
- a set time horizon $T \in \bullet$;
- a set of assignments of values to variables A_0, ..., A_T to be processed sequentially by the agents. Each assignment A_t is a tuple $\left(x_{1t} \leftarrow d_{1t}, x_{2t} \leftarrow d_{2t}, ..., x_{nt} \leftarrow d_{nt}\right)$

The goal is to maximize the total reward during the time horizon:

$$R = \sum_{t=0}^{T} \sum_{x_i, x_j \in V} f_{i,j}\left(d_{i,t}, d_{j,t}\right)$$

One often formulates DCEE problems in terms of binary constraints between pairs of agents. Please refer to Figure 1 for an example of a network of three agents an two binary constraints. In particular, agent $i, i \in \{1..3\}$ controls variable x_i , where $D\_1 = [0..k]$, $D\_2 = [0..m]$, $D\_3 = [0..n]$ are the respective domains of the variables. $R_{1,2}$ and $R_{2,3}$ are the reward matrices that map from the corresponding pair of agents' variable values to a reward, which the agents typically attempt to maximize. The question marks in the reward matrices represent the fact that the reward function $f_{i,j}$ is initially unknown for any pair of variables x_i, x_j .

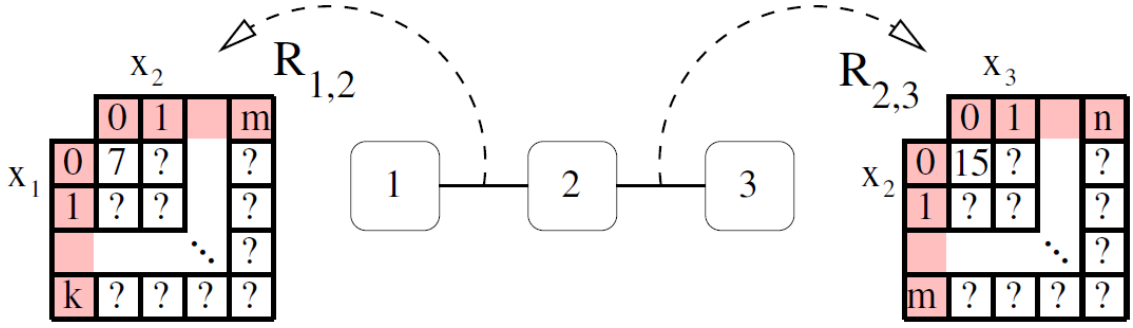

Figure 1 A sample chain constraint graph of three DCEE agents (Taylor, Jain, Tandon, Yokoo, & Tambe, 2011)

Therefore, $f_{i,j}$ must be empirically estimated by agents i and j trying out different variable assignments $(x_i \leftarrow d_i, x_j \leftarrow d_j)$. At the same time, an agent must be able to make use of the known portion of the reward matrix, so that it can maximize the real time reward over the given time horizon T . This need for balancing between *exploration* and *exploitation* of new and old pair-wise cross product tuples among the agents give rise to the name of the framework. Indeed, the agents are *distributed*, meaning they are each autonomous. However, they are aware of their neighbors, and carry out *coordination* delegation (when does each agent do what) in order to optimize the constraint values (Taylor, Jain, Tandon, Yokoo, & Tambe, 2011).

Now that the name of the framework has been broken down, one should also be sure to understand the implication of the algorithms which this work shall explore: the Static Estimation - Optimistic K1, and Static Estimation - Optimistic K2 algorithms. *Static Estimation* means when exploring the unknown tuples (x_i, x_j), the agent will assume that $f_{i,j}$ yields a constant value v . The agent is also *optimistic* in the sense that v is set to be an unrealistically high value. This symbolizes the agent's belief that any unexplored tuple of variables is equally worthy of a visit, and more important than the known ones. Finally, an important factor that shapes coordination among neighboring agents is *k-movement*, where at most k agents can change their variables simultaneously in a neighborhood every round. Naturally, an agent's *neighborhood* is the set of its neighbors. Larger k values allow for more joint moves, but sometimes this can decrease total team performance (Taylor, Jain, Tandon, Yokoo, & Tambe, 2011). This is referred to as the team *uncertainty penalty*, where lower k values allow agents to perform better in a sparse constraint graph. Thus, *K1* means only one agent may change its variable per round, whereas *K2* allows one or two agents to do so. Additionally, it is important to note that the agents are *rational*: each agent will attempt to maximize its own reward, defined as R_c , which is the total constraint value over all constraints with its neighbors:

$$R_c = \sum_{x_j \text{ controlled by } n \in N} f_{i,j}(d_i, d_j)$$

The result of the agent being optimistic in hoping to gain an impossible amount of reward for every unknown variable tuple is that 1) every agent wishes to change configurations on every round, 2) the algorithm will always be exploring the environment (practically speaking, since the premise is that there are too many configurations to exhaustively cover during the given time frame), and 3) agents with the worst performance per neighborhood will

be allowed to explore. On every *round*, every agent will measure the reward between itself and all of its neighbors.

4.3. Algorithms

The Static Estimation-Optimistic K1 (SE-Opt K1) algorithm allows a single agent to change variable(s) per neighborhood. For instance, in Figure 1, if agent 2 changes its variable setting, agents 1 and 3 must remain fixed. Alternatively, if agent 1 changes its variable, agent 2 must remain fixed, but agent 3 could choose to change.

The variable assignment will always result in be an unexplored variable tuple for each neighbor, because the agent is optimistic that such an unexplored variable tuple will have a very high potential reward v. Thus, the total potential gain will be, assuming the agent has L neighbors, $g = L \times v - R_c$. The agent then *bids* this potential gain as a promise of how much it will improve the current situation. After exchanging *bids* with the neighbors, an agent only moves if it sees that its *bid* is the greatest, which means that it is the worst performing member of the neighborhood.

Figure 2 details a walkthrough of the algorithm on sample network of agents. In each round of coordination in a DCEE problem, four SE-Opt K1 agents first exchange their current variable assignment $x_i \leftarrow d_i$. Then, they each empirically measures its pair wise constraint rewards (the innermost integers), updating their reward matrices with these measurements using the latest x_i values provided by their neighbors. Assuming $v = 100$, each agent gathers its total estimated gain g as shown in its dialog bubble. Next, the agents exchange values of g as their *bid*s, and an agent only moves if it has the highest *bid* value. Thus, agent 1 gets to change x_1 in this round, with highest *bid* value of 120.

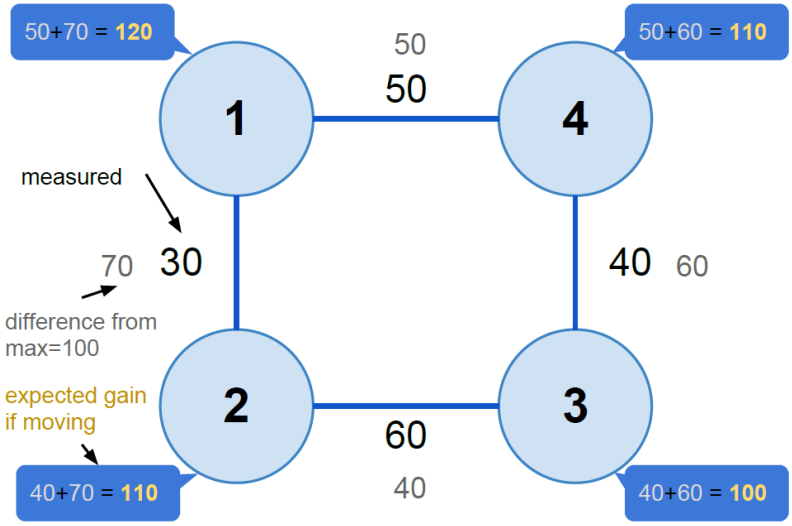

Figure 2 SE-Optimistic K1 agents collaborating during one round.

For the *SE-Optimistic K2* algorithm, each neighborhood can allow up to two agents performing joint movement to change configurations. Communication is more involved as the agents must first look around their neighborhood to see with whom they will likely make the strongest bidding pair (by assuming that the neighbor's neighbors would not change while evaluating the combined rewards). Each agent then sends an *OfferPair* message to their prospective partner p. Should p reply with an *OfferPair* message back as well, then they both will confirm to work with each other. The value g of their joint *bid*, then, is no longer their own gain, but rather the joint gain, which they will now use to compete with other agents. Should p not reply to accept the proposed

partnership, though, the agent will have to evaluate its potential reward as would an agent in the $k=1$ scenario, and then bid using that value.

Please refer to **Figure 3** for an example of SE-Opt K2 algorithm in action. In each round, four SE-Opt K2 agents first exchange their current variable assignment $x_i \leftarrow d_i$. Then, they each empirically measures its pair wise constraint rewards (the innermost integers), noting these rewards into their reward matrices by using the updated x_i values from their neighbors. Assuming $v = 100$, each agent i gathers its total estimated joint gain $g_{i,j}$ for each neighbor j (for example, $g_{1,2} = 160$ and $g_{1,4} = 180$. Note that the shared constraint between i and j is counted only once in the calculation. Since $g_{1,4} > g_{1,2}$, agent 1 identifies its best potential partner $p = 4$ and offers to pair. Fortunately, agent 4 returns the offer, as $g_{1,4} > g_{3,4} = 150$. The pair of agents (1, 4) sends out their joint bid of 180, dominating over the remaining neighbors. Thus the pair proceeds to change x_1 and x_4 for this round.)

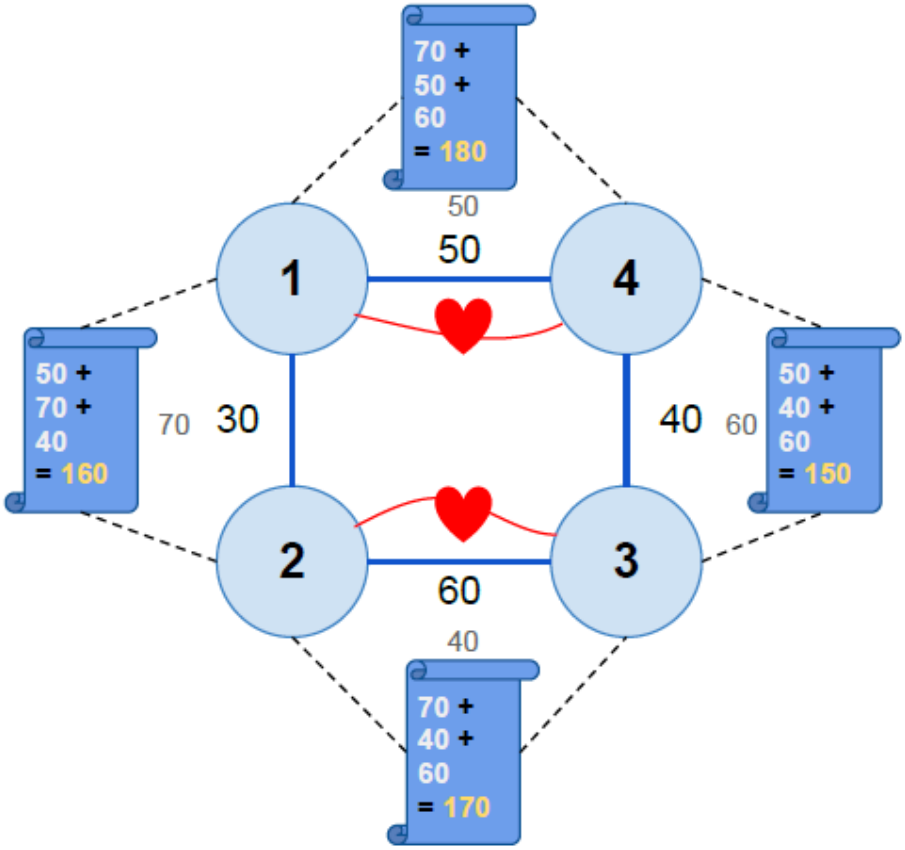

Figure 3 SE-Optimistic K2 agents in action during one round.

5. Implementation

5.1. The traffic simulator

The Autonomous Intersection Management (AIM) microscopic simulator is developed by the Learning Agents Research Group at the University of Texas at Austin, and supports a Manhattan topology of North-South and East-West multi-lane roads joined by a number of intersections. **Figure 4** shows a screenshot of a 2×2 intersection setup in the AIM simulator.

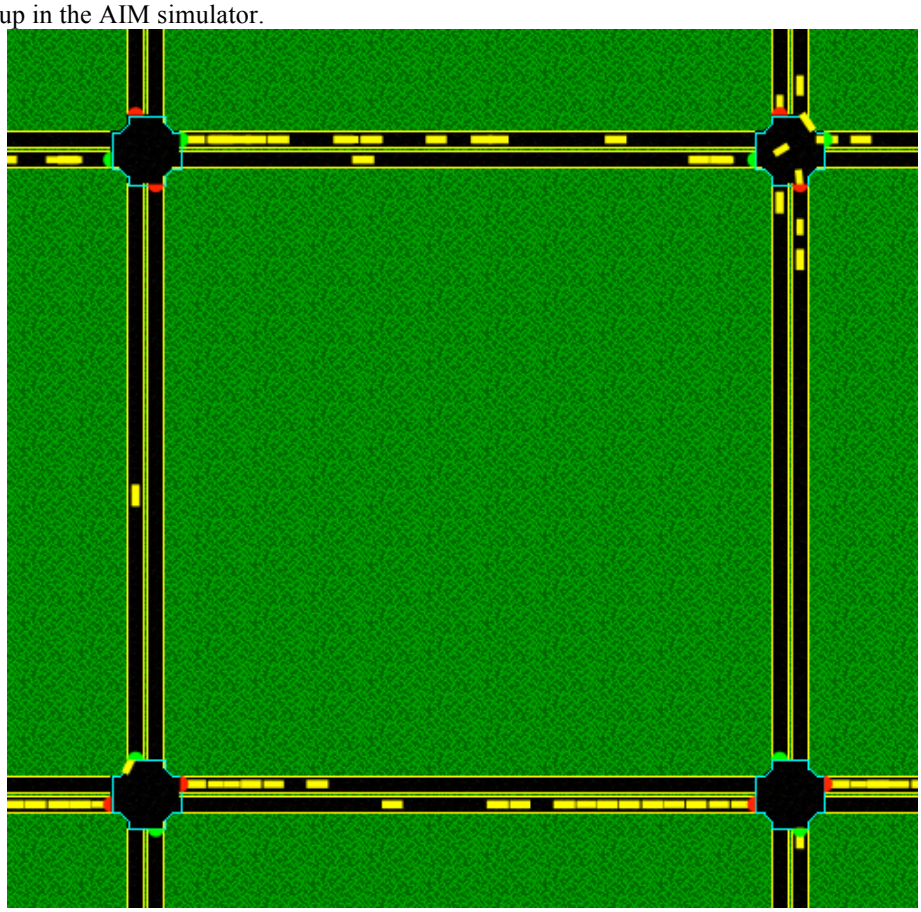

Figure 4 A screenshot of a four-intersection layout in AIM with two-way single-lane traffic.

Each road has two spawn points where new vehicles can enter the system. Each spawn point uses a Poisson process to determine the spawn time for the next vehicles; all spawn points share the same rate parameter λ. Because each direction has one lane, a newly spawned vehicle will not pass other cars, nor does it perform U-turns. Upon creation, each vehicle is assigned a destination, uniformly chosen among the seven possible exit points of the system; the vehicle then follows the shortest path to reach its designated exit point.

5.2. Inter-agent constraint: average travel time

Agents evaluate the rewards of binary constraints with a neighbor by measuring the average travel time of a fixed number of cars traveling on the stretch of road connecting two agents. The total team reward at each round is then the weighted average of this average travel time, scaled by the volume of traffic on each stretch of road linking a pair of agents. For our static estimation agents, we set the unexplored reward to be 0 seconds, an unattainable value for travel time. To allow numerically higher reward value to be more desirable, we negate the actual measurements. Average travel time serves as a reasonable metric to optimize, as it correlates with other commonly used metrics that gauge traffic light performance, such as queue length and throughput. Thus, by letting the agents optimize for higher rewards in this context, we are letting the intersections work toward lower average travel time along the lanes between them, thus improving traffic light performance.

5.3. Performance measures

Two performance measures were singled out for collection:

- *Average link delay (seconds/vehicle)*: take each vehicle currently traveling on a road linking two gents, and measure the time delay that the vehicle experiences as it travels on the current road. This is computed by noting the start time t_{start} for each vehicle as it enters a road. When evaluation occurs at time t, the vehicle has traveled a distance d from the beginning of the current road segment. If there is no obstruction along the way, the vehicle should have traveled that distance under a time duration of $t_{optimal} = \frac{d}{v_{max}}$, where $v_{\max}$ is the maximum legal speed for said vehicle on the road. However, the actual travel time, $t_\delta = t - t_{start}$ may be greater than $t_{optimal}$, and thus one obtains $t_{delay} = t_\delta - t_{optimal}$.
- *Average throughput (vehicles/second)*: this is the average number of vehicles being spawned across all spawnpoints in the simulator. Note that as queue grows longer from the nearest intersection back toward the spawnpoint, there comes a point where no new vehicles can be spawned when they are supposed to due to lack of space. This in turn impacts the spawn rate, and thus is indicative of congestion problem.

5.4. Signal schemes as action space

For this problem domain, the agent each controls precisely one traffic intersection, thus put in charge of the intersection's traffic signal. The space of possible traffic signals is created by 1) restricting the agents in an experiment to a particular signal scheme, and 2) discretizing the space of said signal scheme, and assigning each of the element in the discretized set of signals an ID. This ID, or *signal scheme index*, is the variable controlled by the agent. In the setup for the DCEE experiments, there are two straightforward traffic light signal schemes that were facilitated by the underlying engine: the approximately N-phase signal scheme henceforth known as *AIM*, relies on the simulator's definition of an *active phase*; and the "canonical" signal scheme, so called because it conforms to the design of the simple two-phase pretimed signal scheme.

5.4.1. AIM traffic light operation scheme

Two parameters specify precisely such an active phase: the *green_offset* and *green_time*, both measured in seconds (see **Figure 5**.) For simplicity, the majority of our experiments will have the active phase length fixed at 60 seconds. We then associate each intersection with a DCEE agent, letting each agent control exactly one variable --- its *signal scheme index*. This index enumerates all possible traffic signal configurations, running from 0 to 65 (see **Figure 6**.) Thus, index 0 maps to the tuple (0, 5), which is an active phase with no leading red, and five seconds of green time (followed by two seconds of yellow and 53 seconds of red). To translate the next index, we attempt to increase *green_offset* by a five-second interval, while keeping the total active phase length. Should this not be possible, we instead increase the offset by five seconds, and reset green time to five seconds. This results in a triangular translation table, stopping at (55,5), for a total of 66 possible combinations for the AIM signal format.

Once the *active phase* has been determined, the entire signal layout for each direction (North-South, East-West) will be specified as shown in **Figure 7**. Note that at any moment, only one direction is considered to be active (thus running the active phase signal scheme); the other direction is inactive, running the complementary signal scheme.

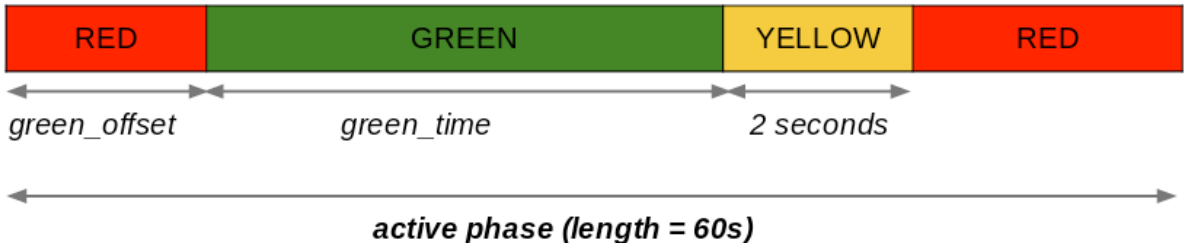

Figure 5 Traffic light configuration that makes up an *"active phase"* in the simulator

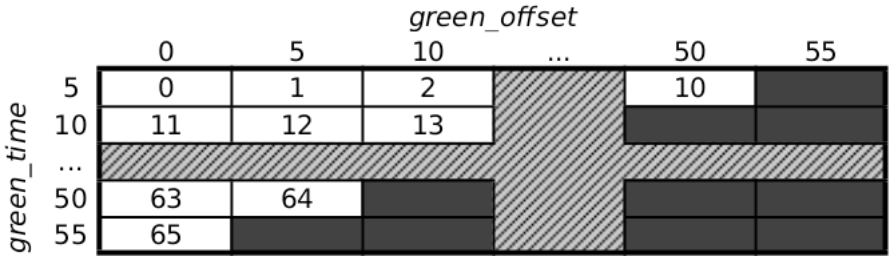

Figure 6 The AIM signal scheme index, and its corresponding ($green\_offset$, $green\_time$) valueseconds.

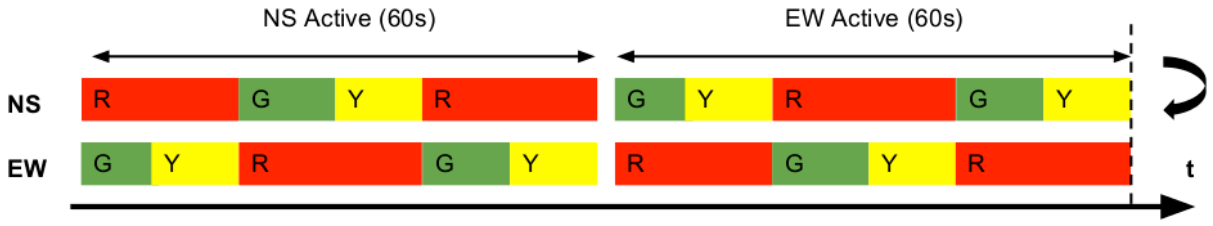

Figure 7 AIM signal scheme schematic, repeating after 120 seconds.

5.4.2. Canonical traffic light operation scheme

This signal scheme operates with two parameters: *green1*, the duration of green split for phase 1, and *C*, the cycle length (**Figure 8**.) Aside from this, the translation process from the *signal scheme index* to actual Canonical signal scheme follows the same process as that for AIM, with a jump step of five seconds for both *green1* and *C*. There are 54 available signal schemes for Canonical (see **Figure 9**).

Figure 8 Canonical signal scheme schematic

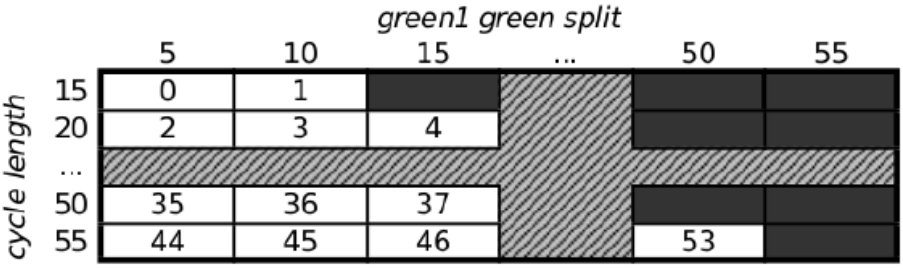

Figure 9 The Canonical signal scheme index

5.5. Incoming traffic flows

Using the aforementioned random generator, the *Simulator* will be responsible for spawning vehicles at spawnpoints, which are situated at the beginning of each lane heading into the network of intersections. The time interval between successive vehicles spawned follows an exponential distribution so that the arrival rate of vehicles at each spawn points obeys the Poisson process. Once created, a vehicle is assigned a destination lane through which it will exit the network. There is no U-turn allowed. The vehicle uses A*-search to determines the shortest path to get from the lane containing its spawnpoint to the exit lane. To simulate the random nature of traffic flows, each vehicle by default will pick a destination in a uniformly random manner.

6. Experimental Setup

6.1. DCEE algorithms

For our experiments, we set up a 2x2 Manhattan grid, in which each road is two-way, each direction having one single lane. Thus, there are four traffic intersections, each controlled by a DCEE agent. Each experiment took 90000 seconds.

The experiments all had a one-hour *warm-up* period, during which the agents will assign random signal schemes to the traffic lights every 300 seconds. As soon as the traffic simulation begins, independent Evaluator threads will start collecting average delay and throughput metrics every 60 seconds. After the *warm-up* period, successive rounds of agents communicating and changing signal schemes took place. There was a *cool down* period of 600 seconds to allow the effect of the new traffic signal to take place. We had the vehicle spawn

rate $\lambda \in \{1800, 300\}$, corresponding to oversaturateed and undersaturated traffic conditions, respectively. Each experiment for a given traffic level and algorithm is repeated 40 times. Data series were processed using a running average window of size 101.

6.2. Isolated actuated traffic signal benchmark

At the beginning, the same *warm-up* period is observed, after which the traffic lights are hardcoded to follow the procedures below:

- Remembers the current active direction (with green time) {North-South/East-West}. For our discussion, let's assume the current active direction is North-South.
- Scans the North and South incoming lane for vehicles within a distance of 30m, every 0.3 seconds. If there are no vehicles detected for North direction, then North *gaps out*. The same applies for the South direction. If and when both North and South direction *gaps out*, then change the active direction to East-West (going through the sequence of turning green-yellow-red clearance for North-South before turning green for East-West.)
- Keeps track of how long North-South has stayed in green. If and when the duration has exceeded max-green_{NS}, then also change the active direction to East-West in the same manner outlined above.
- Repeats the above logic for East-West.

6.3. Determining saturation flow rates

One important variable required for calculating max-green for isolated traffic actuated controllers are the saturation flow rates. A quick experiment was set-up to obtain these statistics. In the experiment, all spawnpoints other than spawnpoint #1 (spawning for the only lane of 1st Street East) were disabled, thus adding no vehicles to the system. Spawnpoint #1 was altered to ignore the Poisson process used to determine the time for the next vehicle to spawn, and instead would try to spawn new vehicles at every time-step. This represents the maximum incoming traffic rate scenario. All spawned vehicles from spawnpoint #1 were then set to always traverse the first encountered intersection in a turn direction $\omega \in \{STRAIGHT, RIGHT, LEFT\}$. Each direction was measured using a separate experiment, and in each experiment, an integer counter tracked the number of vehicles successfully traversing the intersection in the corresponding ω.

Since each ω involves a different curvature through the intersection, one would expect variation in average travel time, and hence the number of vehicles successfully traversing in a fixed amount of time. The result of the three experiments is outlined in **Table 2**.

Table 2 Saturation flow rate experiments, N=40, runtime = 3000 seconds.

ω	**# vehicles**
LEFT	2721.12 ± 0.913
THROUGH	2722.73 ± 1.004
RIGHT	2721.94 ± 1.118

6.4. Determining traffic flow profiles

For each intersection, the traffic flow profile details the rate of incoming traffic from each direction, and for each turn direction, ω.

For simplicity, let us refer to the spawnpoint adjacent to an exit point by the same index. Denote the volume (in vehicles/hour/lane) of vehicles spawning at spawnpoint a and heading ultimately for exit point b to be V_{ab}.

Also, denote $I_{m-\alpha-\omega}$ as the volume of incoming traffic for intersection $m \in \{1,2,3,4\}$ (counter-clockwise from top left), coming from $\alpha = \{EASTBOUND, SOUTHBOUND, NORTHBOUND, WESTBOUND\}$ of the intersection, and turning in direction ω, as defined previously.

Since the traffic simulator has been programmed to simulate the unpredictable nature of traffic, each vehicle spawned, in the uniform traffic volume case, will have automatically decided a random destination $d \in \{1,...,8\} - \{s\}$, where $s \in \{1,...,8\}$ is the spawn location (thus, no U-turn). This means that each vehicle in the uniform traffic volume case has a one in seven chance to go to any of its valid seven destinations. This information, and symmetry, can help us estimate the traffic flow profile for each intersection.

Now, consider $I_{1-EASTBOUND-RIGHT}$ (**Figure 10**). The assumption that vehicles will try to take the shortest route to get to a given destination means that all vehicles from 1 wanting to reach 2 will *all* turn right to get there. However, a vehicle coming from 1 and wanting to get to 4 shall have two possible options for each case (**Figure 10**): turn right then left and right (dotted line), or go straight and then right and straight (dot-dash line). Since one cannot assume to know the driver's behavior or preferences when planning the signal controller, it is reasonable to assume an equal probability of vehicles turning right and going straight at intersection 1, since either decision lets the driver to traverse an equal distance to get to 4.

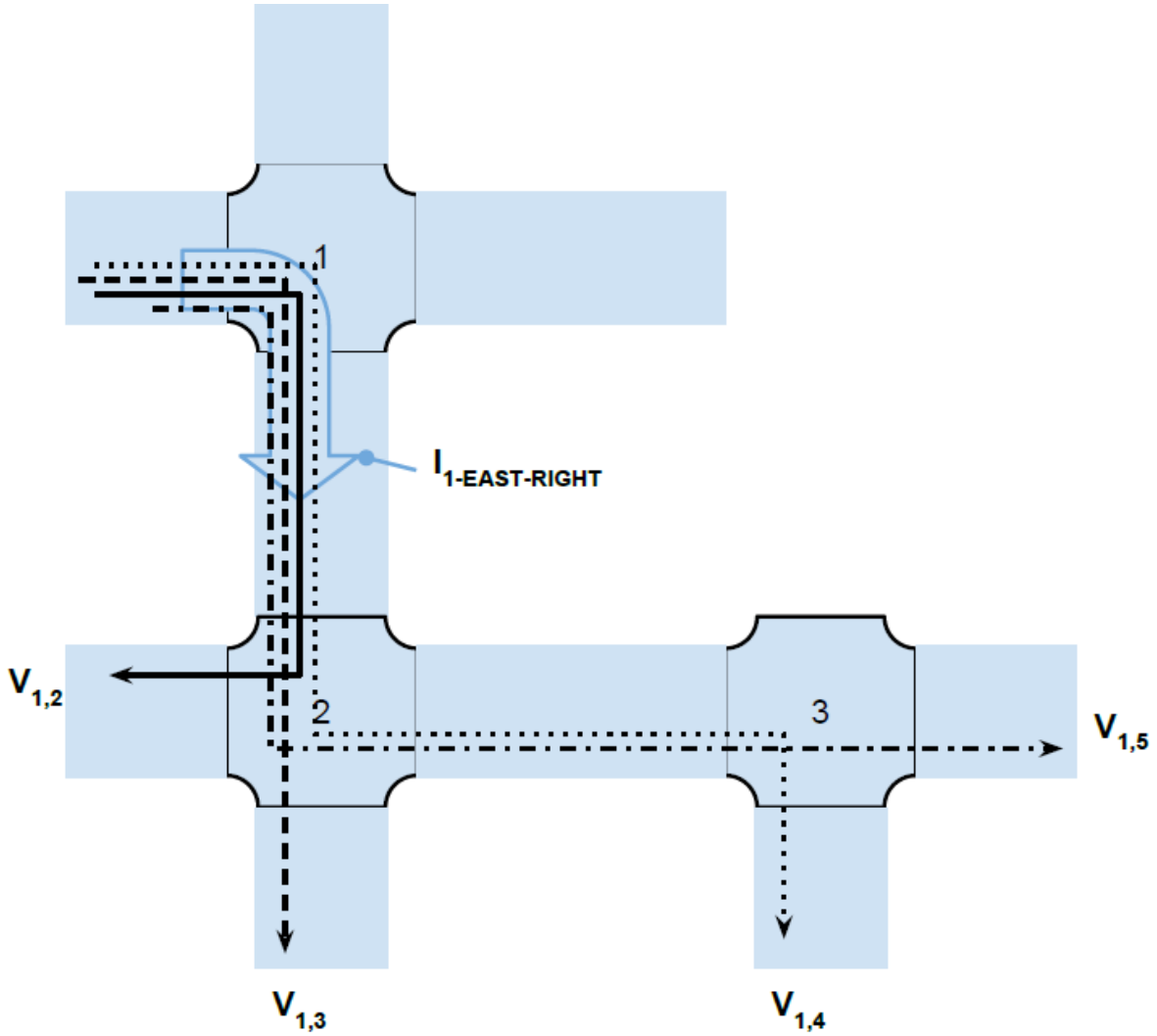

Figure 10 Flow volume for $I_{1-EAST-RIGHT}$. Each dashed arrow is a possible path that the vehicles, having turn so, will take.

Thus, denoting the uniform traffic incoming rate of λ vehicles/hour/lane (at each spawnpoint), the following equation holds:

$$\begin{aligned} I_{1-EAST-RIGHT} &= V_{12} + V_{13} + \frac{1}{2}\left(V_{14} + V_{15}\right) \\ &= \frac{\lambda}{7} + \frac{\lambda}{7} + \frac{1}{2}\left(\frac{\lambda}{7} + \frac{\lambda}{7}\right) \\ &= \frac{3\lambda}{7} \end{aligned}$$

This way, the values for the remaining volume of incoming traffic for I_1 , and then, through symmetry, I_3 .

With the above saturation flow rates (**Table 2**) and traffic volume profile for each and every intersection, the $max-green_{NS}$ and $max-green_{EW}$ parameters is determined through a software package (PASSER (TM) V-09,

2011) for optimizing isolated traffic signal. Refer to **Table 3** for settings regarding high traffic level (1800 vehicles/hour/lane). For low traffic level (300 vehicles/hour/lane), both values are 15 seconds.

Table 3 max_green setting for isolated traffic actuated signal controller agents, $\lambda = 1800$ vehicle/hour/lane

Agent ID	$max\_green_{NS}(seconds)$	$max\_green_{EW}(seconds)$
1	63	57
2	57	63
3	63	57
4	57	63

7. Results

While viewing the figures of the results, the reader should recognize that **PURE-WARMUP** experiments represent the case where the agents kept executing their *warm-up* sequence until the Evaluator threads died. In a sense, this case represents a benchmark of the worst possible performance of the system. Also, the **ISOACTUATED** experiments are scenarios where each traffic intersection is an isolated, traffic-actuated signal.

7.1. AIM traffic light operation scheme

Figure 11 and **Figure 12** show the performance of the DCEE algorithms in high and low traffic conditions, matched against the state-of-practice benchmark of actuated isolated traffic signals.

It is apparent that the DCEE algorithms have shown improvement in average link delay per vehicle over the course of the experiments for both over-saturated and under-saturated traffic levels. In particular, both **K1** and **K2** algorithms performed better than **PURE-WARMUP** over time, indicating performance increase over random action.

This holds also in terms of average throughput for over-saturated traffic condition; the identical throughput performance of all experiments for under-saturated traffic condition was due to the fact that there was no backlogging of vehicles under all circumstances.

Additionally, **K1** outperforms **K2** in terms of average link delay for each traffic condition. Last, there are instances where the DCEE algorithms outperform the state-of-practice of actuated isolated traffic lights.

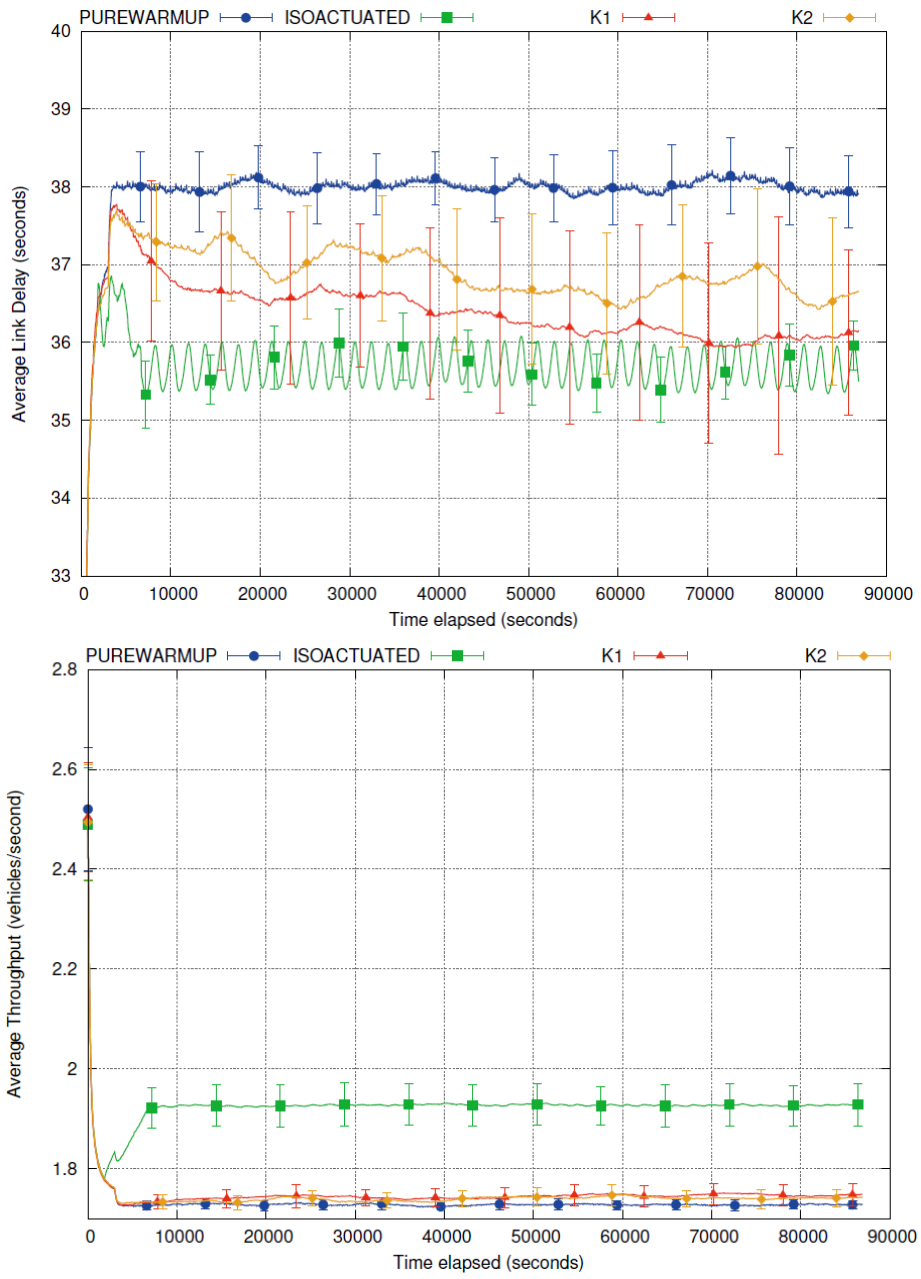

Figure 11 K1 and K2 performance using AIM signal scheme, high traffic

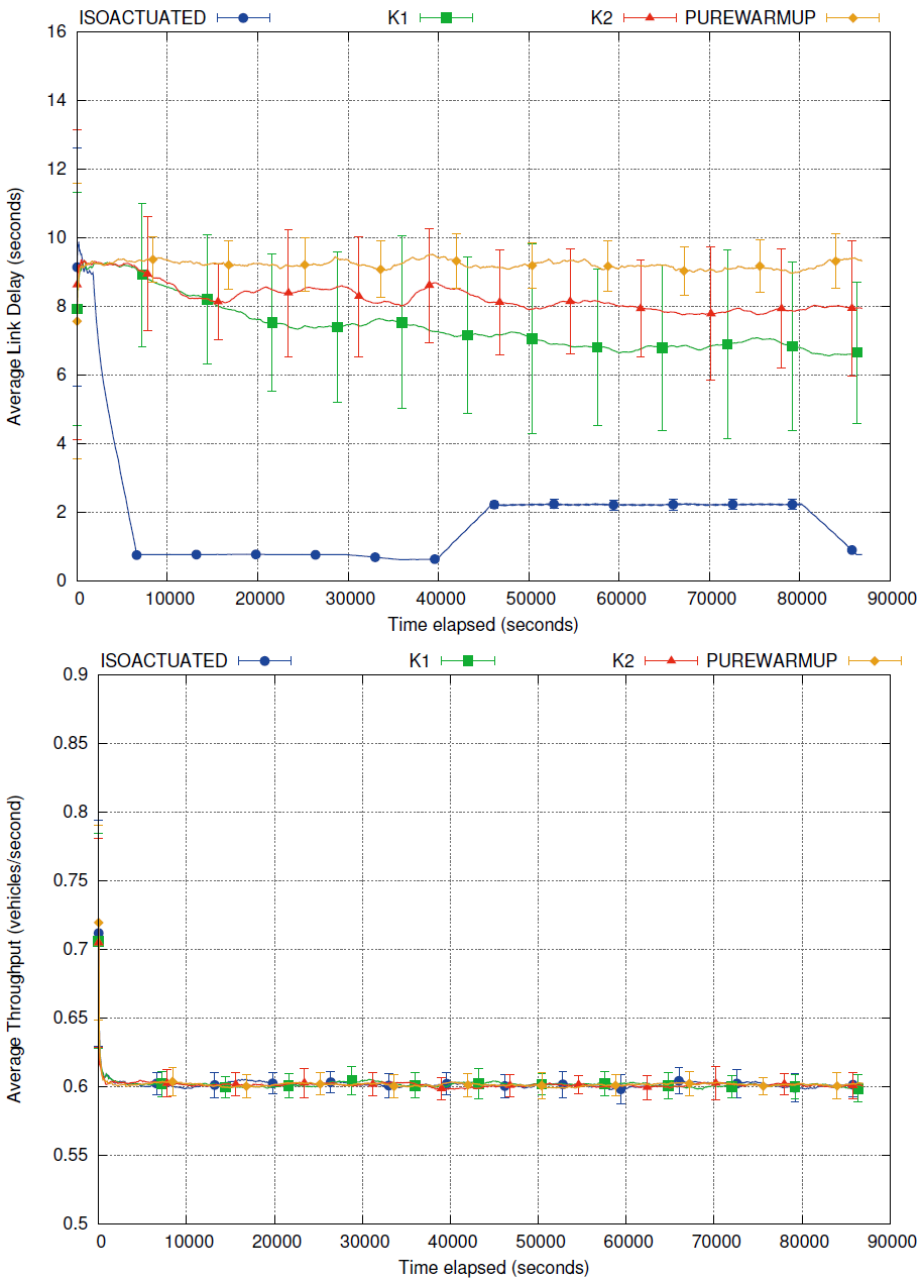

Figure 12 K1 and K2 performance using AIM signal scheme, low traffic

7.2. Canonical traffic light operation scheme

The overall performance of the DCEE algorithms under this signal scheme are shown at high traffic level (**Figure 13**) as well as low (**Figure 14**.) Note that unlike the case with AIM signal scheme, **K1** and **K2** algorithms' performance actually decreased over time for both average throughput and average link delay. This holds for both over-saturated and under-saturated traffic conditions.

However, **K1** outperforms **K2** in both performance metrics and for each traffic condition. Additionally, **K1** managed to maintain a very stable performance level over time relative to **K2**, characterized by very stable standard deviation of average link delay.

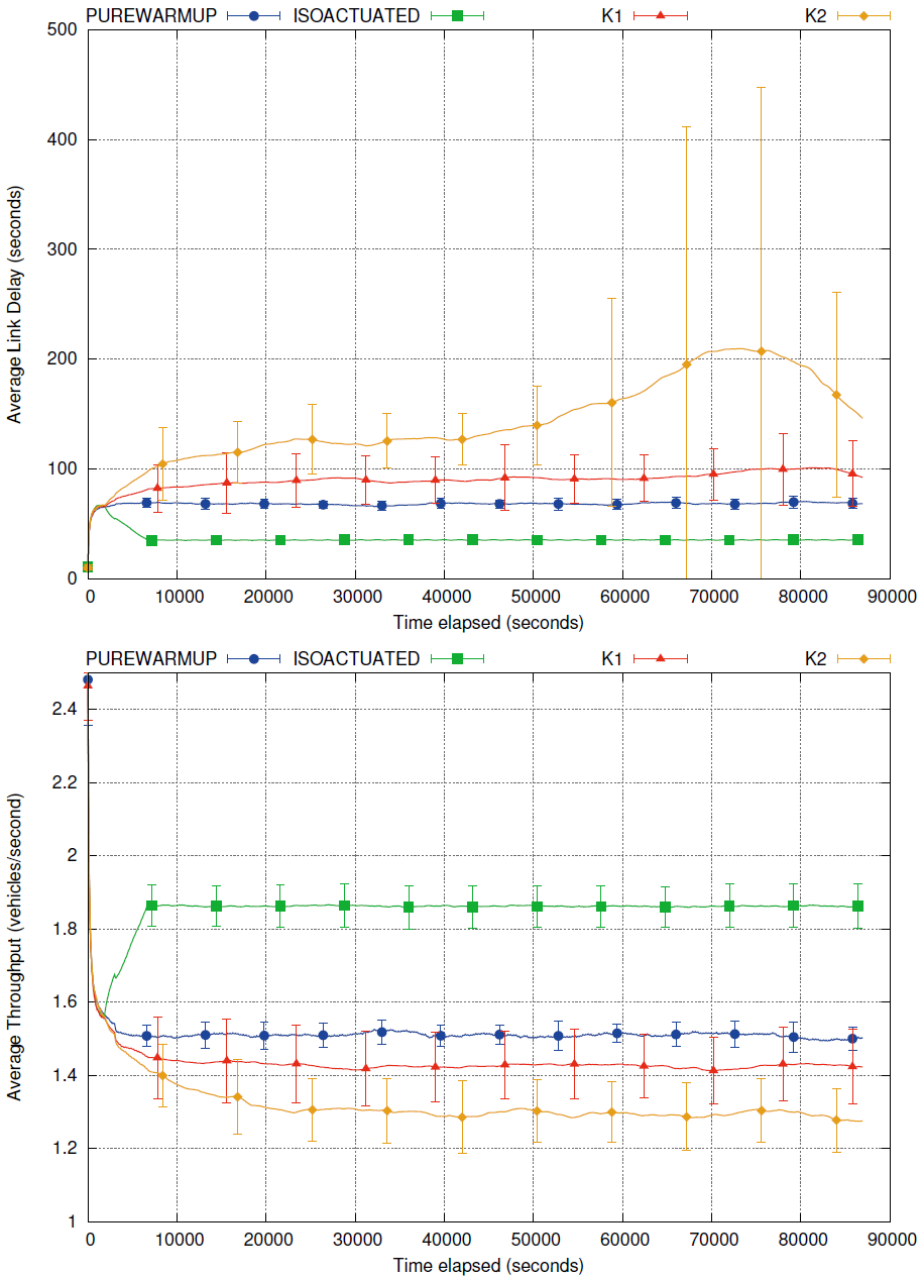

Figure 13 K1 and K2 performance using Canonical signal scheme, high traffic

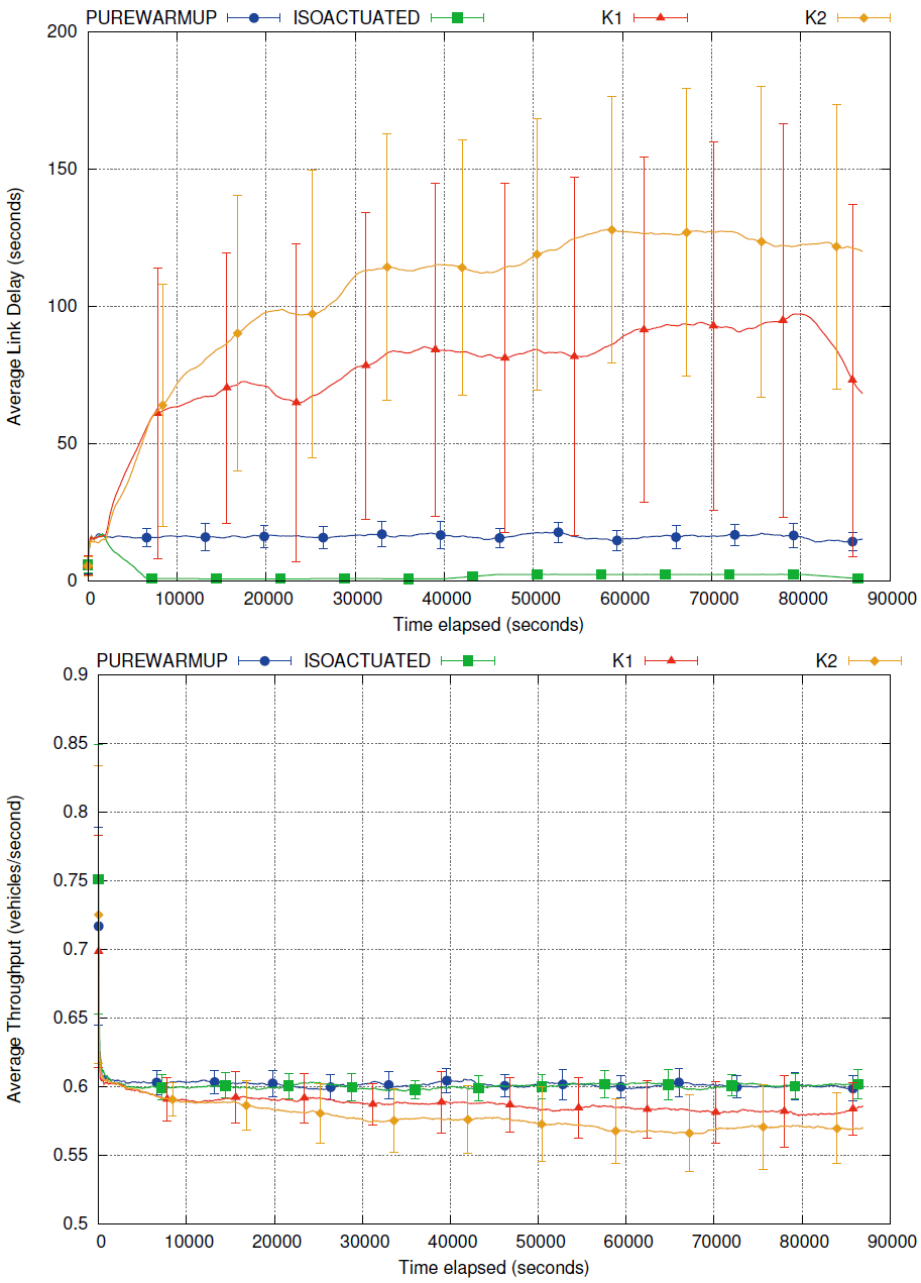

Figure 14 K1 and K2 performance using Canonical signal scheme, low traffic

8. Conclusion

The aforementioned results show that:

- With appropriate limitation of the action space in terms of signal schemes, the DCEE algorithms can enable the agent-based system of traffic lights to improve the traffic performance over time, albeit gradually.
- Performance of these naive algorithms is very sensitive to the signal scheme space employed. In particular, switching from the symmetric signal scheme type (AIM) to asymmetric signal scheme type (Canonical) actually decreased the agents' performance. Also, the frequency at which signal schemes are changed seem to be another important factor, evident by the fact that **PURE-WARMUP** can dominate over the DCEE algorithms under Traditional traffic light signal operation scheme. Finally, the number of agents able to change their signal schemes per round of coordination is also a differentiating factor, as **K1** algorithm outperforms or performs as least as well as its **K2** counterpart. This observation corroborates the *team uncertainty penalty* phenomenon which has been observed in other problem domain applications of the DCEE framework (Taylor, Jain, Tandon, Yokoo, & Tambe, 2011).

9. References

1. Abdoos, M., Mozayani, N., & Bazzan, A. (2013). Holonic multi-agent system for traffic signals control. *Engineering Applications of Artificial Intelligence* , 1575--1587.
2. Junges, R., & Bazzan, A. (2008). Evaluating the performance of DCOP algorithms in a real world, dynamic problem. *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems* (pp. 599-606). International Foundation for Autonomous Agents and Multiagent Systems.
3. McKenny, D., & White, T. (2013, 01). Distributed and adaptive traffic signal control within a realistic traffic simulation. *Engineering Applications of Artificial Intelligence* , 574--583.
4. Mirchandani, P., & Wang, F.-Y. (2005). RHODES to intelligent transportation systems. *IEEE Intelligent Systems , 20*, 10-15.
5. *PASSER (TM) V-09*. (2011). (Texas Transportation Institute) Retrieved from http://ttisoftware.tamu.edu/
6. Robertson, D., & Bretherton, R. (1991, Feburary). Optimizing networks of traffic signals in real time-the SCOOT method. *Vehicular Technology, IEEE Transactions on* , 11-15.
7. Sims, A. G., & Dobinson, K. W. (1980). The Sydney coordinated adaptive traffic (SCAT) system philosophy and benefits. *IEEE Transactions on Vehicular Technology* , 130-137.
8. Taylor, M. E., Jain, M., Tandon, P., Yokoo, M., & Tambe, M. (2011). Distributed on-line multi-agent optimization under uncertainty: Balancing exploration and exploitation. *Advances in Complex Systems* , 471--528.
9. Y., C., Quek, C., & Loh, P. (2009). A novel neuro-cognitive approach to modeling traffic control and flow based on fuzzy neural techniques. *Expert System Applications* , 4788–4803.