2010 13th International IEEE
Annual Conference on Intelligent Transportation Systems
Madeira Island, Portugal, September 19-22, 2010

Soilse: A Decentralized Approach to Optimization of Fluctuating Urban Traffic Using Reinforcement Learning

As'ad Salkham and Vinny Cahill[†]
Distributed Systems Group - School of Computer Science and Statistics
Trinity College Dublin
[†]Member of Lero - The Irish Software Engineering Research Centre
{salkhama, vinny.cahill}@cs.tcd.ie

***Abstract*—Increasing traffic congestion is a major problem in urban areas, which incurs heavy economic and environmental costs in both developing and developed countries. Efficient urban traffic control (UTC) can help reduce traffic congestion. However, the increasing volume and the dynamic nature of urban traffic pose particular challenges to UTC. Reinforcement Learning (RL) has been shown to be a promising approach to efficient UTC. However, most existing work on RL-based UTC does not adequately address the fluctuating nature of urban traffic. This paper presents Soilse[1], a decentralized RL-based UTC optimization scheme that includes a nonparametric pattern change detection mechanism to identify local traffic pattern changes that adversely affect an RL agent's performance. Hence, Soilse is adaptive as agents learn to optimize for different traffic patterns and responsive as agents can detect genuine traffic pattern changes and trigger relearning. We compare the performance of Soilse to two baselines, a fixed-time approach and a saturation balancing algorithm that emulates SCATS, a well-known UTC system. The comparison was performed based on a simulation of traffic in Dublin's inner city centre. Results from using our scheme show an approximate $35\% - 43\%$ and $40\% - 54\%$ better performance in terms of average vehicle waiting time and average number of vehicle stops respectively against the best baseline performance in our simulation.**

I. Introduction

Urban traffic is an evolving problem closely related to population growth and economic factors. Many countries are seeing increases in vehicles per capita with each passing year. In the Organisation for Economic Co-operation and Development (OECD) countries, road motor vehicles per thousand inhabitants have increased over the period from 1990 until 2006 in all studied countries (except the United States) [1].

The negative impact of increasing vehicle numbers can essentially be summed up in one word "congestion". This causes worldwide economic, environmental and social problems. In the EU, congestion costs around 1% of the member countries' Gross Domestic Product (GDP) annually [2] and an estimated ~AU$20.4 billion by 2020 in Australia [3]. In 2007, congestion cost the United States ~US$87.2 billion in 439 urban areas calculated based on wasted time and fuel [4]. As far as the environment is concerned, congestion is a major cause of air and noise pollution. Urban mobility in the EU contributes 40% of the overall CO_2 emissions caused by road transportation while this percentage increases to 70% of all other pollutants [5]. These costs are due to increasing traffic growth and to the stop-go nature of driving in cities and despite advances in vehicle emission reduction technologies [2]. Furthermore, a survey by the Department of Transportation in the United States has shown that 47% of Americans agree that delay caused by traffic congestion is a top community concern [6].

While it is true that better and more efficient UTC systems cannot alone solve this increasing problem, they can surely help to reduce it [5], [7]. Consequently, the need has arisen for more sophisticated UTC systems to provide efficient traffic control strategies that reduces road congestion through minimizing vehicle delay, providing less-interrupted traffic flow, minimizing number of vehicle stops and increasing vehicle velocity. Besides widely deployed UTC systems such as the Sydney Coordinated Adaptive Traffic System (SCATS) [8], [9] and the Split Cycle Offset Optimization Technique (SCOOT) [10], numerous approaches to UTC have been proposed as computational problem-solving methodologies have evolved. Such approaches mainly use Dynamic Programming (DP) [11], [12], [13], [14], graph theory [15], Petri Nets [16] and game theory [17]. Others simply use heuristic models and rule-based/logical programming approaches [18], [19].

Among these, RL has emerged as a promising approach to highly-adaptive UTC optimization [20]. The essence of RL derives from the manner by which nature's intelligent elements can learn by interacting with their surrounding environment. RL is defined by [21] as "learning how to map situations to actions so as to maximise a numerical reward signal". RL agents explore their environment by sensing different situations and then executing some selected action(s) that result in feedback in the form of a reward. RL is an unsupervised learning approach in the sense that an agent does not rely on a knowledgeable master that might have specific domain knowledge. For UTC optimization, we apply an RL strategy that uses Q-Learning [22] given its applicability to online (re)learning that allows for the adaptiveness and responsiveness needed

This work was partly supported by Science Foundation Ireland under Investigator award 02/IN.1/I250 and grant 03/CE2/I303_1 as well as by the IHEA Programme for Research in Third Level Institutions (as the Networked Embedded Systems Centre).

[1]Soilse is the Irish for "lights".

978-1-4244-7659-6/10/$26.00 ©2010 IEEE

by UTC. Q-Learning is a well-established model-free off-policy (explained below) RL strategy based on the concept of discounted expected rewards. An RL agent that uses Q-Learning usually learns with a specific rate $\alpha : 0 \leq \alpha < 1$ and a certain discount rate $\gamma : 0 \leq \gamma < 1$ through a form of environmental representation, typically, a state-action space. It is model-free in the sense that it does not require some a priori likelihood model for the actions that could be executed on the environment. Q-Learning is considered an off-policy strategy as it learns and updates the agent's knowledge even while taking actions that could prove to be non-optimal in the future [20]. Being an off-policy learning strategy, as well as allowing for short period knowledge updating per action taken, Q-Learning is an ideal candidate for UTC optimization given the non-stationary nature of traffic [20].

In a previous work [23], we presented a decentralized RL-based scheme for UTC optimization in which each RL-based agent learns to control a given signalized junction in an adaptive round-robin (ARR) manner with possible collaboration with upstream and downstream neighbours. That scheme did not support pattern change detection and agents did not deal with fluctuating traffic patterns. In this paper we extend our previous work to allow agents to detect local changes in the traffic pattern and consequently respond in an adaptive manner. Agents adapt to local traffic conditions by learning a sequence of traffic light phases[2] to be used. They respond to fluctuating traffic patterns or unsatisfactory performance by relearning based on a local non-parametric traffic-pattern change-detection mechanism. We refer to our approach as Soilse. Soilse can be described as an online RL-based decentralized UTC optimization scheme that deals with fluctuating traffic in an adaptive and responsive manner without a priori knowledge of traffic models. We assessed Soilse using a microscopic urban traffic simulator, which models individual vehicles' behaviours in a detailed map of a real city.

The remainder of the paper is organized as follows. Section II presents related work. Soilse is presented in Section III where the non-parametric pattern change detection mechanism and its collaboration mechanism are detailed. Section IV describes our experimental setup and evaluation results. We conclude in Section V.

II. Related Work

Despite the existence of some widely-deployed adaptive UTC systems, such as SCATS and SCOOT, UTC remains an active research area. Both systems follow an optimization methodology that uses proprietary mathematical models to tune specific settings of a traffic controller for each phase, namely, the offset[3], the split[4] and the cycle time[5]. The performance of both SCATS and SCOOT is however poor under saturated traffic conditions [24], [25].

[2]A phase is characterized by the set of traffic directions allowed to proceed at a given signalized junction from certain approaches at a given time. Only one phase can be active at a time.

[3]The time between signalling adjacent traffic controllers.

[4]The proportioned green time allocated per phase in a cycle.

[5]The time needed to complete a sequence of phases including offsets.

Alternative UTC optimization approaches vary. [15] presents a traffic-responsive urban control approach that is based on a store-and-forward model of signalized junctions represented as a directed graph. [16] uses hybrid Petri Nets to model the traffic network and a supervisor to coordinate all signalized junctions. Several DP approaches to UTC optimization also exist [26], [11], [12], [14], [13], [27]. A reservation-based approach for UTC is presented in [28]. A distributed game theory-based approach for coordination between traffic light controller agents is presented in [17]. An approach for traffic control using decentralized logic programming is presented in [18].

We concentrate on RL-based UTC approaches. A number of these approaches use hybrid modelling techniques such as the combination of genetic programming or fuzzy neural networks along with RL while others are purely RL-based. [20] describes the use of Q-Learning for UTC optimization and presents promising results against pre-timed traffic controllers on a small-scale scenario. [29] also showed showed that Q-Learning outperformed random and best-effort policies. More complex RL techniques were used in [30] where they exploited the Natural Actor-Critic (NAC) [31] algorithm. NAC outperformed a SCATS inspired technique (termed, SAT) in a 10×10 junction grid simulation while optimizing for vehicle average travel time. However, NAC needed approximately three days of learning in order to be on par with SAT. [32] addresses UTC optimization for multiple vehicle types using an algorithm referred to as Distributed W-learning that is based on a combination of Q-learning and W-learning [33]. Significant improvements in performance were achieved for all vehicle types, however, this approach was applied only to stationary traffic conditions.

In order to provide "intelligent" cooperation schemes among RL-based traffic control agents, different RL schemes have been coupled with centrally executed genetic algorithms for parameters tuning as in [34]. A combination of fuzzy neural networks and a form of RL is used to build the hierarchical real-time traffic control architecture presented in [35]. In a vehicle-centric approach, [36] researched the benefits of using multi-agent model-based RL for traffic control. In [37], a similar approach is presented. It is noticeable, that these two approaches place some serious assumptions on the type of information, (e.g., probability estimates of waiting time per vehicle's destination, and its place at every traffic light controller including the state of that traffic light (green or red)) that is needed and might not be possible to acquire realistically, especially, if traffic patterns are changing and drivers' nature is taken into account.

To our knowledge, only two approaches to UTC have attempted to address the fluctuating nature of traffic, one using an RL-like technique in an offline manner [38] and the other using model-based RL in an online manner [39]. The so-called Organic Traffic Control (OTC) approach [38] uses a combination of evolutionary genetic programing and a Learning Classifier System (LCS) which learns similarly to RL in terms of receiving rewards from the environment but models the problem as a set of rules each represented as a triplet of {condition, action, value}. Several rules' conditions

can however match a single environmental situation. The OTC architecture per junction is composed of three layers. The top layer uses an evolutionary algorithm in an offline manner that interacts with a traffic simulator in order to provide new classifiers (for different traffic conditions). The middle layer comprises an LCS while the bottom layer is a tunable traffic light controller. They have simulated two signalized junctions of different sizes with a flow of traffic that has one peak on three different days. Their results show $10-12\%$ improvement in average delay distributed among the three days for the bigger junction and $6-8\%$ for the smaller junction against a fixed-time controller. As an extension of the OTC work, coordination among OTC controllers was added [40], however on a small-scale Manhattan-like grid simulation, no significant improvement on their previous results was obtained. In [39], an RL approach to optimizing UTC while responding to traffic volume change and driver behaviour (Nagel-Schreckenberg model [41]) is presented. The main design is divided into two stages, firstly, learning for a given traffic pattern and, secondly, detecting changes in traffic patterns. For learning, they assume that every stationary situation can be defined by a so called "partial model" that comprises a transition function that estimates the transition probabilities and a reward function for reward estimation. Any typical model-based RL can be used to locally optimize for a given partial model, such as Prioritized Sweeping (PS) [21]. The detection of changes in traffic is based on how well a given partial model can represent the current traffic condition. The experimental setup in [39] is based on a 3×3 Manhattan road network of varying link speed limits $(54,\ 36,\ 18\,km/h)$ where different types of traffic patterns are inserted. They have experimented with scenarios of varying deceleration probabilities; $[zero,\ 0.1,\ 0.2,\ 0.3]$ where the number of stopped vehicles throughout the experiment duration was used as a metric. As comparison baselines, they have used fixed-signal plans, greedy controllers, and Q-Learning and PS RL methods. Only in one single scenario, (i.e., where the deceleration probability was set to 0.1) did their approach outperform the baselines. In the scenario where the deceleration probability was set to $zero$, their approach performed on a par with most of the baselines. In the two scenarios with deceleration probability 0.2 and 0.3, their approach failed against the greedy baseline.

A common critique for most of the RL-based, whether hybrid or not, approaches is that they do not show significantly better performance in realistic simulations and do not support online UTC optimization under fluctuating urban traffic conditions without using model-based[6] approaches.

III. SOILSE

Soilse models individual traffic light controllers as adaptive RL agents capable of responding to changing traffic patterns that might adversely impact their performance. These agents (built using a generic C++ RL framework described in [23].) can make use of collaboration with their neighbours to improve performance. In contrast to previous RL-based work in UTC, Soilse provides a flexible RL agent design that supports optimization for different traffic patterns using a pattern change detection (PCD) mechanism that causes an agent to relearn based on the degree of pattern change detected. The PCD mechanism we use is a nonparametric one that does not depend on a presumed traffic data distribution.

In a non-collaborative setting, each signalized junction is controlled by a dedicated Soilse agent that operates independently. However, in a collaborative setting each signalized junction is controlled by a dedicated agent, which we refer to as a SoilseC agent (presented in Subsection III-C), that operates in collaboration with neighbouring SoilseC agents. Both Soilse and SoilseC agents make use of a local PCD mechanism to provide for responsiveness in the face of fluctuating traffic patterns.

A. Pattern Change Detection

Our PCD mechanism is an online nonparametric change detection mechanism that quantifies the degree of traffic pattern change affecting on agent's performance. Such a mechanism is nonparametric in the sense that it does not rely on a specific distribution for incoming traffic on a given signalized junction. Our approach was inspired by work on anomaly detection for attacks or intrusions of a certain type in computer networks [43]. The technique used is based on sequential analysis of data series using a cumulative sum of squares (CUSUM) [44] that can identify change points in data variance (σ^2). CUSUM is a nonparametric change detection technique that has been shown to be accurate in helping to detect flooding attacks in a short duration [43], [45], [46].

Our PCD mechanism is used to detect variance change points on each lanes' incoming traffic using CUSUM and also incorporates knowledge of the agent's performance at each signalized junction. We choose lanes as the level of granularity at which to identify changes since lanes represent not only the traffic load but also its directionality. Therefore, this approach captures not only changes in traffic load but also changes in traffic direction. The performance of an RL agent controller can be naturally assessed based on its recent rewards as they represent the goodness of its local behaviour. Hence, recent reward history is used to provide the metric for the overall near-past agent performance.

The resulting design is based on multistage lane-centric filtering, see Figure (1). The incoming traffic count per lane on a given junction is sampled and filtered using a moving average filter in order to produce a smoother input (since we assume a fine-grained initial input as frequent as a reading per second over a minute) for the second stage. The output is then passed to the CUSUM filter that identifies changes in traffic variance on a given lane, see Equation (1).

$$CUSUM_{k,n}(Lane_a) = \mid \frac{\sum_{i=1}^{k} X_{a\,i}^2}{\sum_{j=1}^{n} X_{a\,j}^2} - \frac{k}{n} \mid;\ for\ 1 \leq k < n \quad (1)$$

[6]Model-based UTC approaches that use RL do not take into account the fact that using a priori traffic models given the uncertain behaviour of urban traffic is a strong assumption. This aligns with what [42] argues in relation to providing a traffic control scheme based on models of traffic flow: "which, given the highly nonlinear and uncertain aspects of human behaviour, is a virtually hopeless task in complex multiple-intersection networks."

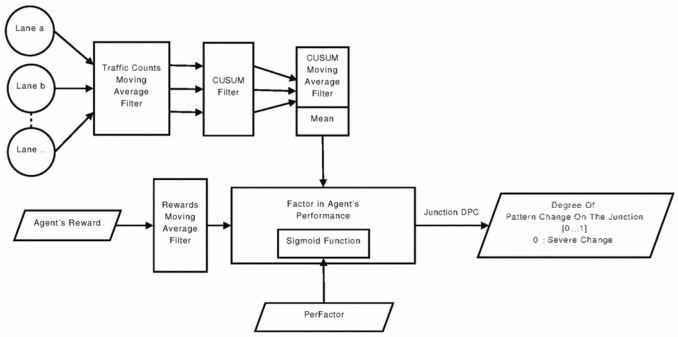

Figure 1. Junction PCD High-Level Scheme

The time series $\{X_{a\,1}, X_{a\,2}, ..., X_{a\,n}\}$ is formed from the outputs of the first filtering stage, i.e., a series of size n of smoothed traffic counts for lane a. The value of k represents the lagging sample size. Essentially, CUSUM works by comparing the sum of squares of a portion of a given sample against the sum of squares of the whole sample. This allows it to determine what is the proportional relevance of the smaller sample of size k to the whole sample of size n. Another moving average filter is applied on the CUSUM filter outputs (per lane) and the mean of these is then used as the final representation of the degree of pattern change (DPC) in incoming traffic for the junction.

Since we are interested in changes that affect the agent's performance, we need to incorporate the controller agent's performance in the DPC, which can then be used in the reparameterization process of the agent for relearning. A natural metric for the agent performance is the moving average of rewards (MAR) over a given time window. As rewards are an intrinsic indicator of RL agents' performance in the first place, we benefit from their availability without introducing an extra artificial metric of performance. Thus, the product $(MAR \times DPC)$ is used. Furthermore, in order to confine (squash) that value to a known range we apply a sigmoid function, see Equation (2), that has a known range of $[-1, 1]$.

$$DPC = tanh((MAR \times DPC)/\,PerFactor) \qquad (2)$$

$PerFactor$ is used to scale down the sigmoid function ($tanh()$) input data in order for the function to give sensible output in the range $[-1, 1]$. Determining $PerFactor$ is dependant on the range of performance to be measured for the controller agent. We are only concerned when the final DPC is negative, i.e., the agent is not performing well while the local traffic pattern is changing. This is because an agent should not relearn unless its performance is adversely affected by the possible traffic pattern change. At this stage, DPC will have a negative value only if the incorporated MAR was negative as the result of CUSUM is always positive. Hence, we chose the final DPC value to be in the range of $[0, 1]$ where $DPC \leftarrow 1 - abs(DPC_{[-1,0]})$ similar to the notion of $0 \leq P\_value \leq 1$ in nonparametric statistical tests. Moreover, the closer DPC is to 0, the more severe the negative change is. At a later stage, we detect a so-called genuine change as a situation where a sample of DPCs are persistently crossing a given junction change threshold (JCT) fixed for all signalized junctions. The sampling of DPC starts when a single DPC value crosses the JCT and continues until the so-called *persistence sample* is ready, (i.e., its size is met). That sample's mean is then compared against the JCT and a genuine pattern change is detected if this sample mean crosses the JCT, however, this is dependent on the thresholding technique used.

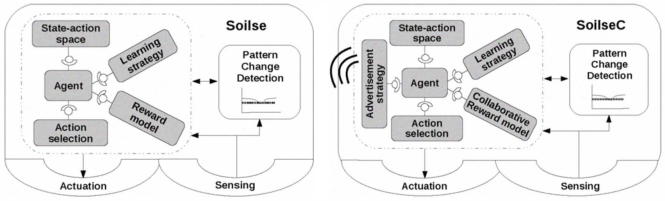

Figure 2. Soilse And SoilseC Agent Structures

B. Soilse Agent

A Soilse agent, see Figure (2), is composed of an RL agent and a PCD module. The RL agent comprises a representation of the environment, i.e., a state-action space, strategies for action selection and learning, and a reward model. Actuation and sensing are provided through generic interfaces.

The PCD module interacts with the RL agent by enquiring about the agent's performance, (i.e., rewards) as well as triggering relearning when required. It periodically polls the sensing interface for traffic counts per lane on the given junction in order to carry out the PCD process. If a genuine traffic pattern change is detected, the PCD mechanism passes the resulting DPC value to the RL agent. The DPC is used by the RL agent to calculate new learning parameters including both the learning and action selection parameters.

Soilse agents use Q-Learning as their learning strategy and ϵ-greedy as their action selection strategy. The state-action space representation is based on the ARR design of [23]. A given signalized junction's state-action space is modelled based on the available phases and their status, (i.e., busy/not busy). A given phase's status depends on all the incoming approaches for that phase. A pair of a phase and its status is considered a state (e.g., $S_0 \implies (Phase_x\, is\, busy)$) in the model. A given phase's status is determined by comparing the total number of vehicles within queueing range on its incoming approaches against a specific threshold value. A Soilse agent provides a number of actions, (i.e., candidate phase durations including a zero-second duration action) that could possibly be chosen in a given state. Given that we follow a round-robin style over n phases, after any action we take in any state of phase P_i, the next action will be in a state of phase $P_{(i+1)\,mod\,n}$ depending on local traffic conditions. The availability of a zero-second duration action allows the Soilse agent to skip unnecessary phases while exploring for policy optimization.

$$R = (\#vehicles_{\,crossed} - \#vehicles_{\,waiting\,on\,the\,junction}) \qquad (3)$$

Algorithm 1 Soilse Agent Process

/*Initialization*/
$Action\ Selection\ (AS) \leftarrow \epsilon$-greedy
While $(Soilse\ is\ running)$
  $Execute\ a_t$; $Receive\ s_{t+1}$
  $r_{t+1} \leftarrow R$
  $Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) +$
    $\alpha\left[r_{t+1} + \gamma\, max_a Q(s_{t+1}, a) - Q_t(s_t, a_t)\right]$
  Select $a_{t+1} \in A_{t+1} : \{all\ actions\ for\ s_{t+1}\}$ using AS
  $s_t \leftarrow s_{t+1}$, $a_t \leftarrow a_{t+1}$
  If $(relearn)$ /* Relearn status is updated by the PCD where it passes the DPC value to be used in reparameterization */
    Reparameterize the agent by passing (AS, DPC) to Algorithm (2)
  EndIf
  If $(Soilse.exploration == true)$
    Decay α and ϵ using $eq(5)$
  EndIf
EndWhile

Algorithm 2 Reparameterize Per Action Selection Strategy

Switch(AS)
case(ϵ-greedy):{
$\alpha \leftarrow \epsilon \leftarrow eq(4)$
$\alpha_{decay\ rate} \leftarrow \epsilon_{decay\ rate} \leftarrow eq(6)$
}break
EndSwitch

The reward model used (see Equation 3) aims at capturing the traffic that has crossed the junction during a given phase duration and the remaining waiting traffic on all approaches on the junction. The reward will result in a negative reinforcement if a given action (timing) on a given phase results in more traffic waiting on the junction as a whole compared to the traffic that has crossed.

During execution (see Algorithm (1)), the agent executes an action, receives its next state and calculates its local reward. The agent then uses its local reward for its policy update using Q-Learning and selects the next action and updates its state. Furthermore, the agent checks whether it was asked to relearn by its PCD, if so, it reparameterizes itself using the DPC value passed by the PCD. Naturally, the agent decays its learning and action-selection related parameters as long as it is exploring. The Soilse agent will continue exploration as long as its learning parameters have not reached their preset minimum values ($\alpha \approx \epsilon \approx 0$) where exploitation then begins.

When necessary, a new learning and action selection strategy parameters are calculated based on that DPC value. Given that the lower the DPC value, the more severe is the traffic pattern change that is adversely affecting the agent performance, a higher learning rate (α) is needed in these cases in order to cope with the severe change, see Equation 4. Similar to the new learning rate calculation, higher exploration in case of a lower DPC value is needed. The epsilon used in ϵ-greedy needs to be higher for the action selection to be more exploratory, see Equation 4.

$$\alpha_{new} = \epsilon_{new} = (1 - DPC) \tag{4}$$

In order to determine the duration of the relearning, a decay rate needs to be calculated per relearning parameter. The decay should be exponential using a generic Equation (5) as we need the relearning parameters to decay in a manner that is proportional to their value but that reduces exploration gradually.

$$value_{new} = (e^{-(value_{decay\ rate}) \times time\ step}) \times value_{initial} \tag{5}$$

A natural logarithmic function is then used to calculate decay rates that are proportional to the DPC but inversely proportional to the $PolicyModelSize$ (explained below) and the $ExpFactor$, (i.e., the larger the $PolicyModelSize$ and the $ExpFactor$ the slower the relearning/exploration should finish). The calculation is carried out as in Equation (6).

$$\alpha_{decay\ rate} = \epsilon_{decay\ rate} = \frac{log_e(1/(1 - DPC))}{PolicyModelSize \times ExpFactor} \tag{6}$$

For example, if a state-action space has two states where each has two actions, the $PolicyModelSize$ for that state-action space would be $(1 \times 2) + (1 \times 2) = 4$. The proportional relation can also be controlled using a so-called exploration factor ($ExpFactor$). The higher the $ExpFactor$, the more weight is given to the policy model size relative to the DPC value.

C. SoilseC Agent

A collaborative (SoilseC) agent has the same design as a Soilse agent with the addition of an advertisement strategy, see Figure (2). In addition, a collaborative reward model is used as opposed to the local reward model in Soilse agents. That collaborative reward model includes a local reward model similar to the Soilse agent design but also allows for the incorporation of exchanged reward information.

Exchanged information includes a series of rewards ordered by age. The older the reward value is, the less important it is. The exchanged rewards are discounted using a Net Present Value (NPV) [47] inspired Equation (7) that is a well-known method used in economics for discounting a series of values based on age. The NPV equation diminishes the significance of older rewards based on a given $disc\_rate$ value.

$$NPV(r_t) = \frac{r_t}{(1 + disc\_rate)^{(rv\_size - (t+1))}} \tag{7}$$

The collaborative reward is formed from the combination of a local reward and the NPV method applied to the advertisements received from neighbours in order to discount the received rewards by age. A normalization procedure follows per number of senders and the size of reward vectors for each.

$$CollStatus = \frac{\sum_{\forall n \in sending\ neighbours} \frac{\sum_{t=0}^{t=rv\_size_n - 1} NPV_n(r_t)}{rv\_size_n}}{number\ of\ sending\ neighbours} \tag{8}$$

$$r_{coll} \leftarrow (r_{local} + CollStatus) \tag{9}$$

Consequently, a single value ($CollStatus$, see Equation (8)) denoting the overall recent performance of all sending SoilseC agents is calculated and added to the current local reward value, see Equation (9).

The only difference to the Soilse agent algorithm is that after the SoilseC agent calculates its local reward, it is added

to its local reward history which is used for its performance information exchange. If it is time for collaboration (depending on a predefined frequency) and the local reward history is not empty, the agent advertises its local reward history to a given set of neighbours predefined by the advertisement strategy, (i.e., depending on the collaboration mode set). The agent then calculates its collaborative reward, which is used for its policy update using Q-Learning.

IV. Evaluation

The evaluation of our approach is based on a simulation of traffic in Dublin inner city centre. The UTC simulator [48] that we use follows a microscopic model. Its input is a set of XML files describing the road network to be simulated and the valid phases for each signalized junction. This includes the number of lanes per road, the maximum allowed speed on a given road, and the distances between connected junctions. Moreover, traffic can be generated between specific junctions or among user-defined zones where the source/destination junctions are selected randomly within the source/destination zones. In the UTC simulator, vehicles exhibit different behaviours such as car-following, acceleration, deceleration and lane-switching. They also abide by the speed limits on the different roads they travel on. The UTC simulator was also used in our earlier work [23] and in [32] for evaluating multi-policy optimization schemes in decentralized autonomic systems.

A. Experimental Setup and Results

The experimental scenario is based on a map of Dublin's inner city centre. The map (see Figure (3)) represents the real road network of a considerable portion of Dublin city centre. In this scenario, there are 62 signalized junctions out of an overall total of 270 junctions. The overall traffic duration is $\sim$ 19 hours comprising four different patterns: A uniform low pattern (ULP) where traffic is generated over $\sim$ 4 hours following a uniform pattern of low traffic load consisting of 2000 vehicles. The morning peak pattern (MPP) where traffic is generated over $\sim$ 4 hours following a pattern that reflects high traffic loads on main roads consisting of $20,000$ vehicles. The uniform high pattern (UHP) where traffic is generated over $\sim$ 7 hours following a uniform pattern of high traffic load consisting of $21,000$ vehicles. The evening peak pattern (EPP) where traffic generated over $\sim$ 4 hours following a pattern that reflects high traffic loads on main roads consisting of $20,000$ vehicles but generally opposite in direction to the morning peak pattern.

For the ULP and UHP traffic patterns, traffic is generated in both directions from all opposing edges of the map uniformly. To clarify, traffic incoming from different sources in $zones\,\{A, B, C, D\}$ is destined for different exits on $zones\,\{I, K, L, J\}$ and vice versa. Also, traffic incoming from different sources in $zones\,\{A, E, G, I\}$ is destined for different exits on $zones\,\{D, F, H, , J\}$ and vice versa. The difference between ULP and UHP is only in the traffic load and in the pattern duration. For the MPP traffic pattern, two types of traffic are generated, the first is heavy traffic destined

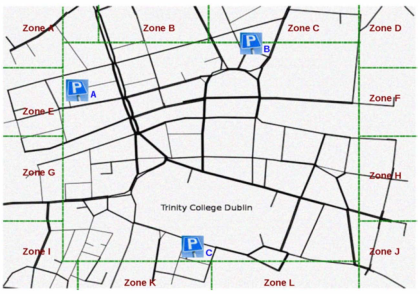

Figure 3. Dublin Inner City Centre Map

for parking spaces in the city centre and the second is light uniform traffic similar to the ULP. Incoming heavy traffic in that case arrives to a given parking space from all remote zones, for example, $parking\ space\ A$ would receive heavy traffic from all zones except nearby $zone\ E$ (as vehicles will arrive almost immediately) and so on. Such heavy traffic amounts to nearly 6324 vehicles per parking space area over the MPP duration. The opposite happens in the EPP where heavy traffic leaves from the parking spaces to all remote zones. Traffic loads leaving the parking spaces are similar in load to those in the MPP. Light uniform traffic similar to the ULP is generated simultaneously. The scenario would run for the joint series of its patterns, i.e., $ULP \rightarrow MPP \rightarrow UHP \rightarrow EPP$ for a duration slightly more than 19 hours to allow for the most recent traffic to clear the map.

The Soilse and SoilseC agents have no a priori knowledge of the traffic patterns discussed above. The agents share the following common learning strategy (LS), action selection (AS), and PCD specifics:

- LS: Q-learning is used as the learning strategy with α (learning rate) initially set to a high value of 0.99 and gradually decreasing based on an initial $\alpha_{decay\,rate} = 0.03$. This allows α to reach the minimum value of 0.001 after $\sim$ 115 minutes. The discount factor (γ) is fixed to 0.3.
- AS: $\epsilon-$greedy is used. The initial ϵ value is set to a high 0.99 and gradually decreases based on an initial $\epsilon_{decay\,rate} = 0.03$. Analogous to Q-learning's α, ϵ reaches the minimum value of 0.001 after $\sim$ 115 minutes.
- PCD: For the sample size needed for the CUSUM of squares on a lane, $n = 30$ and $k = 15$ are used. The moving average filter on each lane's traffic has a moving sample size of 60 traffic counts collected every second. For the smoothing moving average filter on the CUSUM of squares output, a sample size of 10 is used. The reward moving average filter uses a moving sample of 10 rewards. A fixed thresholding technique is used with a junction change threshold set to 0.85. A $PerFactor$ of 10 is used to squash the resulting DPC while the persistence sample has a size of 10.

For both Soilse and SoilseC agents, $ExpFactor$ is set to 2 while every SoilseC agent follows an advertisement strategy with discount rate 0.1 and collaboration frequency of 240

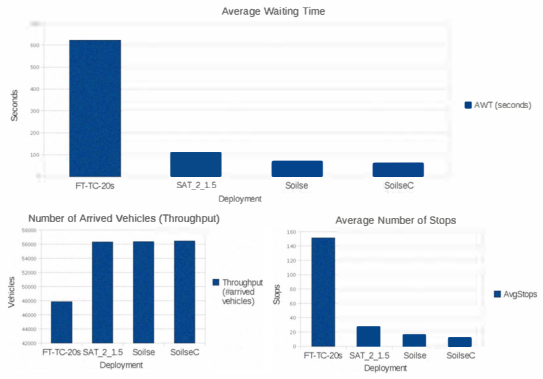

Figure 4. Dublin Inner City Centre Vehicle's AWT, AvgStops and Throughput

seconds and a collaboration mode where a SoilseC agent receives information only from upstream neighbours and send information only to the downstream ones. The threshold used to determine a phase's busy status is set to 1. The action set used is composed of 0, 20 and 30 second available for each state in the Soilse or SoilseC state-action space.

Our baselines for comparison are a SAT-like [49] algorithm that roughly emulates SCATS' behaviour of saturation balancing and to a Fixed Time Traffic Controller (FT-TC) deployment. The FT-TC cycles through a given junction's phases giving a fixed phase time duration for each. We chose to compare against FT-TC with 20 seconds phase time assuming that 20 seconds is a reasonable average phase time and that the minimum phase time in Soilse and SoilseC agents is 20 seconds (excluding the zero-second action used for skipping phases). The SAT deployment tries to achieve a 90% saturation level and uses a 20 second minimum phase time and a maximum cycle length of $[min\_phase\_time \times factor \times number\,of\,phases]$. The SAT controllers try to adapt according to the saturation level by incrementing or decrementing phase durations at the beginning of each cycle depending on information from the previous cycle. The decrement or increment amount (DIM) is a fixed number. The best performing setting for SAT, referred to as SAT_2_1.5, was under $\{DIM = 2\,and\,factor = 1.5\}$ where the number of phases is relative to each controlled signalized junction.

We rely on two metrics to assess the performance of Soilse and SoilseC against the baselines. The first metric is the average waiting time (AWT) for a vehicle which represents the average amount of time that a vehicle is motionless throughout the journey to its destination. The second metric is the average number of vehicle stops (AvgStops) which represents the average number of times that a vehicle's velocity reaches $zero$ throughout the journey to its destination. We also present the number of arrived vehicles as a measure of overall throughput. The results presented here are based on the average of three simulation runs. In Figure (4), AWT and AvgStops results for Soilse and SoilseC against the baselines are presented in addition to the throughput. Soilse and SoilseC achieved the lowest AWT compared to both baselines, i.e. $\sim 71.92\,sec$ and $\sim 63.47\,sec$ respectively. In terms of AvgStops Soilse and SoilseC also performed best resulting in

Performance%	Soilse		
Against (FT-TC-20s, SAT_2_1.5)	∼AWT%	∼#Arrived Vehicles%	∼AvgStops%
ϵ-greedy	(-88.45%, -35.72%)	(+15.05%, +0.08%)	(-88.82%, -40.01%)

Performance%	SoilseC		
Against (FT-TC-20s, SAT_2_1.5)	∼AWT%	∼#Arrived Vehicles%	∼AvgStops%
ϵ-greedy	(-89.80%, -43.27%)	(+15.18%, +0.23%)	(-91.53%, -54.56%)

Table I
DUBLIN INNER CITY CENTRE - SOILSE AND SOILSEC AGAINST THE BASELINES

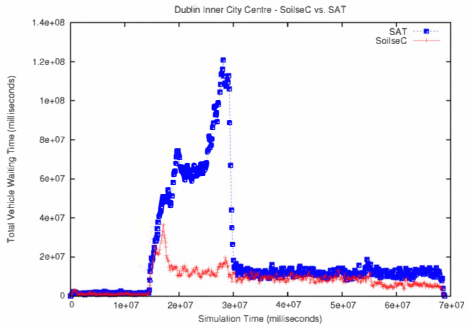

Figure 5. Dublin Inner City Centre - SoilseC vs. SAT (SAT_2_1.5) - Total Vehicle Waiting Time Throughout The Simulation Time

~ 17 and ~ 13 AvgStops respectively. FT-TC-20s on the other hand was the highest in terms of AWT ($\sim 622.74\,sec$) and AvgStops (~ 151) and the lowest in terms of throughput, i.e., 47895 vehicles. Furthermore, SAT_2_1.5, Soilse and SoilseC achieved similar throughput results; 56337, 56385 and 56470 vehicles respectively.

A performance comparison of Soilse and SoilseC against the baselines is presented in Table (I). Soilse and SoilseC has outperformed FT-TC-20s in terms of all metrics. Soilse provided $\sim 88.45\%$ lower AWT and SoilseC $\sim 89.80\%$ lower AWT against FT-TC-20s. Soilse and SoilseC also provided a better performance against FT-TC-20s in terms of AvgStops, i.e., $\sim 88.82\%$ and $\sim 91.53\%$ lower respectively. In terms of the number of arrived vehicles, Soilse and SoilseC allowed more vehicles to arrive to their destinations in comparison to FT-TC-20s. Specifically, Soilse and SoilseC allowed $\sim 15.05\%$ and $\sim 15.18\%$ more vehicles to arrive to their destinations respectively when compared to FT-TC-20s.

The SAT deployment, i.e., SAT_2_1.5 provided a more competitive baseline than FT-TC-20s. However, Soilse and SoilseC also outperformed SAT_2_1.5 in terms of both AWT and AvgStops but differences in the number of arrived vehicles were generally marginal against SAT_2_1.5. Soilse provided $\sim 35.72\%$ lower AWT than SAT_2_1.5. On the other hand, SoilseC achieved $\sim 43.27\%$ lower AWT as opposed to SAT_2_1.5. Furthermore, Soilse outperformed SAT_2_1.5 by $\sim 40.01\%$ in terms of AvgStops. SoilseC achieved $\sim 54.56\%$ less AvgStops than SAT_2_1.5. Moreover, a plot showing the difference between SoilseC and SAT_2_1.5 throughout the simulation time in terms of total vehicle waiting time is shown

in Figure (5). It is noticeable that SAT_2_1.5 could not respond to the change in traffic patterns in an adequately adaptive manner leaving vehicles to wait more on the way to their destinations.

V. Conclusions

This paper described and evaluated a decentralized scheme for online RL-based UTC optimization in a responsive and adaptive manner. RL has been shown to be a promising path for modelling and optimizing UTC when paired with a PCD mechanism. The collaborative version of Soilse, i.e., SoilseC shows that collaboration among agents controlling adjacent signalized junctions is beneficial in terms of vehicle AWT and AvgStops. In an evaluation based on a simulation of a real map of Dublin inner city centre (62 signalized junctions), Soilse and SoilseC outperformed both baselines in terms of vehicle AWT and AvgStops.

References

[1] OECD, *OECD Factbook 2008: Economic, Environmental and Social Statistics*, 2008.

[2] European Commission, "Memo - towards a new culture for urban mobility," 2007.

[3] D. Cosgrove and D. Gargett, "Estimating urban traffic and congestion cost trends for australian cities," 2007.

[4] D. Schrank and T. Lomax, "2009 annual urban mobility report," 2009.

[5] European Commission, "Green paper - towards a new culture for urban mobility sec(2007) 1209," vol. COM/2007/0551 final, 2007.

[6] United States FHWA, "Managing our congested streets and highways," Washington, DC, 2001.

[7] United States DOT, "Intelligent transportation systems for traffic signal control - deployment benefits and lessons learned," 2007.

[8] P. R. Lowrie, "Scats: The sydney co-ordinated adaptive traffic system-principles, methodology, algorithms," in *Proceedings of the IEE International Conference on Road Traffic Signalling*, 1982, pp. 67–70.

[9] A. Sims and K. Dobinson, "The sydney coordinated adaptive traffic (scat) system philosophy and benefits," *Vehicular Technology, IEEE Transactions on*, vol. 29, no. 2, pp. 130–137, May 1980.

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

[11] P. Mirchandani and F.-Y. Wang, "Rhodes to intelligent transportation systems," *Intelligent Systems, IEEE*, vol. 20, no. 1, pp. 10 – 15, jan.-feb. 2005.

[12] V. Mauro and C. Di Taranto, "Utopia," *Control Computers Communications in Transportation*, pp. 245–252, 1990, pergamon Press, Oxford.

[13] I. Porche and S. Lafortune, "Dynamic traffic control: Decentralized and coordinated methods," in *Proceedings of the IEEE Conference on ITS*, 1997.

[14] J. L. Farges, J. J. Henry, and J. Tuffal, "The prodyn real-time traffic algorithm," in *Proceedings of the IEE International Conference on Road Traffic Signalling*, 1983, pp. 307–312.

[15] V. Dinopoulou, C. Diakaki, and M. Papageorgiou, "Applications of the urban traffic control strategy tuc." *European Journal of Operational Research*, vol. 175, no. 3, pp. 1652–1665, 2006.

[16] A. Di Febbraro, D. Giglio, and N. Sacco, "Urban traffic control structure based on hybrid petri nets," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 224–237, Dec. 2004.

[17] A. L. C. Bazzan, "A distributed approach for coordination of traffic signal agents," *AAMAS*, vol. 10, no. 1, pp. 131–164, 2004.

[18] G. Felici, G. Rinaldi, A. Sforza, and K. Truemper, "A logic programming based approach for on-line traffic control," *Transportation Research Part C: Emerging Technologies*, vol. 14, no. 3, pp. 175 – 189, 2006.

[19] B. De Schutter, "Optimal traffic light control for a single intersection," in *Proc. American Control Conference the 1999*, vol. 3, 1999, pp. 2195–2199 vol.3.

[20] B. Abdulhai, P. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," in *ASCE Journal of Transportation Engineering*, vol. 129(3), 2003, pp. 278–284.

[21] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[22] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[23] A. Salkham, R. Cunningham, A. Garg, and V. Cahill, "A collaborative reinforcement learning approach to urban traffic control optimization," *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, vol. 2, pp. 560–566, 2008.

[24] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialosa, and Y. Wang, "Review of road traffic control strategies," in *Proceedings of the IEEE*, vol. 91, 2003, pp. 2043–2067.

[25] United States FHWA, "Signal timing under saturated conditions," *FHWA-HOP-09-008*, November 2008.

[26] P. Mirchandani and L. Head, "A real-time traffic signal control system: architecture, algorithms, and analysis," *Transportation Research Part C: Emerging Technologies*, vol. 9, pp. 415–432, December 2001.

[27] N. Gartner, "Opac: A demand-responsive strategy for traffic signal control," *U.S. Dept. Transportation*, Transp. Res. Record 906, 1983.

[28] K. Dresner and P. Stone, "Multiagent traffic management: A reservation-based intersection control mechanism," in *AAMAS '04*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 530–537.

[29] E. Camponogara and K. J. Werner, "Distributed learning agents in urban traffic control." in *LNCS*, vol. 2902. Springer, 2003, pp. 324–335.

[30] S. Richter, D. Aberdeen, and J. Yu, "Natural actor-critic for road traffic optimisation," in *Advances in Neural Information Processing Systems*, vol. 19. The MIT Press, Cambridge, MA, 2007.

[31] J. Peters, S. Vijayakumar, and S. Schaal, "Natural actor-critic." in *ECML*, 2005, pp. 280–291.

[32] I. Dusparic and V. Cahill, "Distributed W-Learning: Multi-policy optimization in self-organizing systems," in *Third IEEE SASO*, 2009.

[33] M. Humphrys, "Action selection methods using reinforcement learning," Ph.D. dissertation, University of Cambridge, 1996.

[34] Z.-s. Yang, X. Chen, Y.-s. Tang, and J.-p. Sun, "Intelligent cooperation control of urban traffic networks," in *Proceedings of 2005 ICMLC*, vol. 3, 2005, pp. 1482–1486.

[35] D. Srinivasan, M. C. Choy, and R. Cheu, "Neural networks for real-time traffic signal control," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 7, no. 3, pp. 261–272, Sept. 2006.

[36] M. Wiering, "Multi-agent reinforcement learning for traffic light control," in *Proceedings of the 17th ICML*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 1151–1158.

[37] M. Steingröver, R. Schouten, S. Peelen, E. Nijhuis, and B. Bakker, "Reinforcement learning of traffic light controllers adapting to traffic congestion." in *BNAIC*, 2005, pp. 216–223.

[38] H. Prothmann, F. Rochner, S. Tomforde, J. Branke, C. Müller-Schloer, and H. Schmeck, "Organic control of traffic lights," pp. 219–233, 2008.

[39] D. d. Oliveira, A. L. C. Bazzan, B. C. d. Silva, E. W. Basso, and L. Nunes, "Reinforcement learning based control of traffic lights in non-stationary environments: A case study in a microscopic simulator," in *CEUR Workshop Proceedings*, vol. 223. CEUR-WS.org, 2006.

[40] S. Tomforde, H. Prothmann, F. Rochner, J. Branke, J. Hahner, C. Muller-Schloer, and H. Schmeck, "Decentralised progressive signal systems for organic traffic control," in *Proc. Second IEEE SASO '08*, 20–24 Oct. 2008, pp. 413–422.

[41] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," *Journal de Physique I*, no. 115, p. 2221, 1992.

[42] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Hoboken, NJ: Wiley,, 2003.

[43] M. Thottan and C. Ji, "Anomaly detection in ip networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2191–2204, 2003.

[44] K. J. Oh, M. S. Moon, and T. Y. Kim, "Variance change point detection via artificial neural networks for data separation," *Neurocomputing*, vol. 68, pp. 239–250, 2005.

[45] H. Wang, D. Zhang, and K. G. Shin, "Detecting syn flooding attacks," in *In Proceedings of the IEEE Infocom*. IEEE, 2002, pp. 1530–1539.

[46] V. A. Siris and F. Papagalou, "Application of anomaly detection algorithms for detecting syn flooding attacks," *Computer Communications*, vol. 29, no. 9, pp. 1433 – 1442, 2006, iCON 2004 - 12th IEEE International Conference on Network 2004.

[47] G. C. I. Lin and S. V. Nagalingam, *CIM justification and optimisation*. London, UK: Taylor & Francis, 2000.

[48] V. Reynolds, V. Cahill, and A. Senart, "Requirements for an ubiquitous computing simulation and emulation environment," in *InterSense'06*. New York, NY, USA: ACM, 2006.

[49] S. Richter, "Learning traffic control - towards practical traffic control using policy gradients - diplomarbeit." Albert-Ludwigs-Universität Freiburg, 2006.