A Collaborative Reinforcement Learning Approach to Urban Traffic Control Optimization

As'ad Salkham, Raymond Cunningham, Anurag Garg, and Vinny Cahill
Distributed Systems Group
School of Computer Science and Statistics
Trinity College Dublin, Ireland
{salkhama, raymond.cunningham, anurag.garg, vinny.cahill}@cs.tcd.ie

Abstract

The high growth rate of vehicles per capita now poses a real challenge to efficient Urban Traffic Control (UTC). An efficient solution to UTC must be adaptive in order to deal with the highly-dynamic nature of urban traffic. In the near future, global positioning systems and vehicle-to-vehicle/infrastructure communication may provide a more detailed local view of the traffic situation that could be employed for better global UTC optimization. In this paper we describe the design of a next-generation UTC system that exploits such local knowledge about a junction's traffic in order to optimize traffic control. Global UTC optimization is achieved using a local Adaptive Round Robin (ARR) phase switching model optimized using Collaborative Reinforcement Learning (CRL). The design employs an ARR-CRL-based agent controller for each signalized junction that collaborates with neighbouring agents in order to learn appropriate phase timing based on the traffic pattern. We compare our approach to non-adaptive fixed-time UTC system and to a saturation balancing algorithm in a large-scale simulation of traffic in Dublin's inner city centre. We show that the ARR-CRL approach can provide significant improvement resulting in up to ~57% lower average waiting time per vehicle compared to the saturation balancing algorithm.

1. Introduction

Managing traffic in urban areas is a continuously evolving problem. Increasing population size requires more efficient transportation systems and hence better traffic control. Even developed countries are suffering high costs because of increasing road congestion levels. In the European Union (EU) alone, congestion costs 0.5% of the member countries' Gross Domestic Product (GDP), and this is expected to increase to roughly 1% of the EU's GDP by 2010 if the problem is not dealt with properly [13]. In 2002, the number of vehicles per thousand persons had reached 460 which is nearly double the number in 1975. In addition, vehicles are now travelling triple the overall distance that they travelled 30 years ago [14]. Congestion and non-optimal driving in the EU accounts for up to 50% of fuel consumption on road networks resulting in toxic emissions that could otherwise be diminished [14]. Urban transport contributes 40% of carbon dioxide emissions from road traffic in the EU thus resulting in serious health and safety problems [13]. In order to avoid the high costs predicted by such threats, next-generation UTC has to provide efficient solutions to the problem of traffic management. Fortunately, the increasing adoption of global positioning and vehicle-to-vehicle/infrastructure communication systems may provide more detailed local view of the traffic situation that could be exploited to achieve efficient global UTC optimization.

Minimizing vehicle travel time, reducing traffic delay, increasing vehicle velocity, and prioritising emergency traffic are goals that an efficient UTC system may realize. Designing and implementing such a UTC system is not straightforward. Issues like the unpredictability of traffic flow, the heterogeneity of vehicles and the communication and fusion of traffic data, must all be taken into consideration. In order to provide a scheme that deals with all these challenges, an adaptive UTC solution must be deployed [12, 26]. Reinforcement Learning (RL) [36] is considered to be one of the approaches that provides adaptive optimization solutions to control problems. Classical RL is a centralized approach while a number of collaborative and decentralized RL methods [11, 3, 37] have been proposed for solving decentralized optimization problems.

We aim to use Collaborative Reinforcement Learning (CRL) to provide adaptive and efficient UTC by exploiting vehicle location data. Each signalized junction, (i.e., a junction controlled by a traffic signal) runs a CRL-based traffic controller/agent that follows an adaptive phase cycle, i.e., Adaptive Round Robin (ARR), and observes the local traffic pattern from local vehicle location data. Adaptiveness is provided through a set of actions that provide different service times per phase. In this paper we describe a large-scale UTC, (i.e., Dublin's inner city centre) optimization scheme using CRL-based ARR controllers (ARR-CRL). We also experiment with a decentralized ARR-RL design in order to assess the effect of collaboration on UTC optimization. We show that both the ARR-RL and ARR-CRL schemes outperform the non-adaptive fixed-time UTC and the saturation balancing algorithm baselines. Moreover, we show that collaboration can significantly improve performance in terms of average vehicle waiting time. Our ARR-CRL and ARR-RL schemes were developed using our generic CRL framework that provides the necessary components for agent-based application development

The urban traffic simulator we use is microscopic and models individual vehicles' behaviour in a detailed map of a real city. In addition, we rely on vehicle location data to provide the information needed by each agent, (e.g., vehicle count on a certain approach). In effect, UTC solutions or policies achieved through a given optimization scheme are near-optimal given the challenging, complex and dynamic problem nature. This is in contrary to the static chess problem, for example, where the possible system states are finite and each action's effect is predictable.

The remainder of the paper is organized as follows. Section 2 presents related work. Section 3 describes our generic CRL framework. Section 4 presents the decentralized ARR-RL-based and the ARR-CRL-based UTC simulation of Dublin's inner city centre traffic network and provides experimental results. In Section 5 we discuss future work and conclude in Section 6.

2. Related Work

Traffic control has been a widely-known problem for a long time. Many systems and methodologies to address it have been proposed over the past four decades. Most existing UTC systems are based on complex mathematical models to optimize specific settings of a traffic controller for each phase, namely, the time between signalling adjacent traffic controllers (offset), the green time on each approach (split), and the time given to all approaches on a specific junction (cycle time). The well-known SCATS [22] and SCOOT [18] systems follow such a methodology. These systems have improved traffic conditions in many countries [19] but the question remains open when it comes to their absolute success, due for instance, to the overhead of complex personnel training and their acceptability [19]. These systems are generally centralized and mainly suffer from inefficient handling of saturated traffic conditions due to inadequate real-time adaptiveness. They also need a complex software implementation that impairs their portability and incurs high maintenance cost [26, 33, 5]. In addition, undesirable human intervention in many unexpected situations such as accidents is almost inevitable. Other approaches like OPAC [16], PRODYN [15], and RHODES [24] calculate switching times by solving dynamic optimization problems in a real-time manner. Such systems suffer exponential complexities [10] that diminish their chances of being deployed on a large scale [26]. Another dimension being pursued in UTC is based on store-and-forward modelling as in TUC [9]. TUC models the traffic network as a connected graph of junctions and regulates a set of control rules (involving green time) while the system is online. The approach has low complexity and promising simulation and empirical results. Other methods are single intersection centric and use several forms of Dynamic Programming (DP) [36] as the means to minimise total delay time, for instance, COP [33], ADPAS [20], and ALLONS-D [29]. Nevertheless, problem formulation for such UTC methods is not considered a straightforward issue [20]. In [6] a heuristic/fuzzy model for a cooperative and a non-cooperative traffic simulation based on a variation of the Green Light District (GLD) [40] simulator is presented. Nevertheless, there is no significant improvement for the average junction waiting time using their cooperative approach against the non-cooperative one. Bazzan A. [4] proposes an evolutionary game theory based approach where agents try to balance between their local and global goals. However, the evaluation was based on a macroscopic simulation of a single arterial street controlled by multiple intersection agents.

(Multi)-Agent Systems (MAS) [25], RL and numerous decentralized RL methods are being customized for UTC. This is a new way of achieving highly-adaptive and responsive near-optimal solutions for the UTC problem. Abdulhai *et al.* [2, 1] have shown that the use of RL, particularly Q-Learning [38], for providing adaptive traffic control solutions is a promising approach to pursue. They argue that the use of Q-Learning is encouraging since it is an off-policy unsupervised learning approach that does not need a predefined model for the environment. In [2] results from using Q-Learning for an isolated traffic light controller showed that it outperformed the pre-timed control scheme for variable traffic flows. Q-Learning either slightly outperformed or was equal to the pre-timed control when traffic flows were uniform or constant. Weiring *et al.* [40, 39] researched the benefits of using multi-agent model-based RL for traffic control. Their approach is car-centric where each car estimates its waiting time and communicates it to the nearest traffic light. The traffic lights that they use are RL-based agents that implement Q-Learning. Moreover, they experiment with different local and global communication scenarios where traffic lights can exchange knowledge for better decision making. Steingröver *et al.* [35] provide a similar approach in addition to taking into account congestion levels at a given junction. A simple pair of connected traffic light junctions each running a Q-Learning-based agent is presented in [7] where they model and control the small traffic network using a stochastic game scheme. Their results showed that Q-Learning outperformed random and best-effort policies. Moreover, the average number of waiting vehicles was reduced by ~30% when both agents were using Q-learning as opposed to it being used by one agent at a time. Pendrith [27] proposed a distributed Q-Learning scheme in which optimization is aimed at controlling vehicle speed. The basic model used is a 3×3 grid of mobile vehicles where the learning agent (vehicle) is positioned in the middle. Vehicles are presumed to be equipped with radar sensors that enable a given vehicle to determine the states of the surrounding vehicles if any. More complex RL techniques were used by Richter *et al.* [32]. They exploited the Natural Actor-Critic (NAC) [28] algorithm that is based on 4 different RL methods, i.e., policy gradient, natural gradient, temporal difference and least-square temporal difference. In their simplified simulation they had 5 scenarios and every junction on the grid had 4 phases. NAC managed to outperform a SCATS inspired technique (namely, SAT) in a 10×10 junction grid simulation while optimizing for vehicle average travel time. Furthermore, Cao *et al.* have used a form of RL classifier system to build a distributed learning control scheme for traffic light junctions [8]. In order to provide (intelligent) cooperation schemes among RL-based traffic control agents, RL has been coupled with different genetic algorithms in several cases [23, 34].

It is very rare to find a large-scale urban traffic simulation based on real city maps. Moreover, the optimization problem in many proposed systems was occasionally not clearly defined, i.e., non-specific reward model or a vague environment representation (state-action space) and agent collaboration specifics. The lack of a generic design framework for RL and CRL applications has also limited the experimentation with different design choices, (e.g., different reward models, state-action spaces, learning strategies and action selection techniques). In addition, the use of a model-based RL approach as in [40, 39] adds unnecessary complexities compared to using model-free Q-Learning as argued by Abdulhai *et al.* [2, 1]. We argue as well that it is more realistic to assume the future pervasiveness of relatively cheap and well-understood sensor technologies, e.g., global positioning devices, rather than relying on expensive and inaccurate infrastructure, e.g., radar or camera sensors.

3. The CRL Framework

Sutton *et al.* [36] introduce RL as *"learning how to map situations to actions so as to maximise a numerical reward signal"*. Any RL solution is based on two basic elements, namely, a reward function and a value function. Optionally, some RL solutions rely on a model of the environment to predict the reward and next state after taking an action in a given state. The reward function is meant to provide an immediate goodness measure for a certain action in a given state. The value function tries to indicate what is best in the future by accumulating the relevant immediate rewards throughout a (finite) horizon. Interaction with the environment eventually provides the RL-based agent with a policy that defines what is the best action to take in any state at any given time. Moreover, action selection can occur using exploratory strategies, (e.g., (ϵ-)greedy or Boltzmann) or non-exploratory strategies, (e.g., random or greedy). If a form of direct collaboration is required among originally RL-based agents, they are then expanded to be CRL-based agents.

Q-Learning is a well-established model-free RL technique based on the concept of discounted expected rewards. An RL-based agent that uses Q-Learning usually learns with a specific rate $0 \leq \alpha < 1$ and a certain discount factor $0 \leq \gamma < 1$ through a Markov Decision Process (MDP) representation of the environment.

We have designed a generic framework to support the implementation of both RL and CRL applications. The use of a framework enables us to experiment with different application designs in a more structured and flexible manner. The CRL framework is a C++ library that provides the programmer with all the components needed to build an RL application, e.g., agents, learning strategies, action selection strategies, states, actions, MDP representation, and model. By model in the CRL framework we mean the structure where the learnt values are stored and indexed by some key, for instance, a key can be in the form of (state_ID, action_ID) if we are using Q-Learning. The framework also supports CRL application development by providing, in addition to the common RL application needs, a feedback or an advertisement strategy, neighbourhood management and caching. In that case, the model has caching support for the information communicated from neighbouring agents. Figure 1 represents the CRL framework class diagram.

Furthermore, each CRL agent manages its view of its neighbours using a class that provides an interface for communication. A certain advertisement strategy followed by every CRL agent determines how the latter should update its knowledge from its neighbours and how to decide on what information is to be communicated to them.

4. Decentralized ARR-RL-based and ARR-CRL-based UTC Simulation of Dublin

The most popular designs for UTC simulations are based on either a microscopic or a macroscopic approach [17].

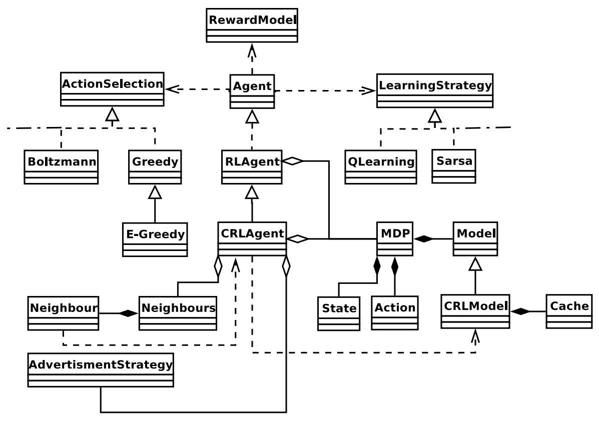

Figure 1. CRL framework class diagram

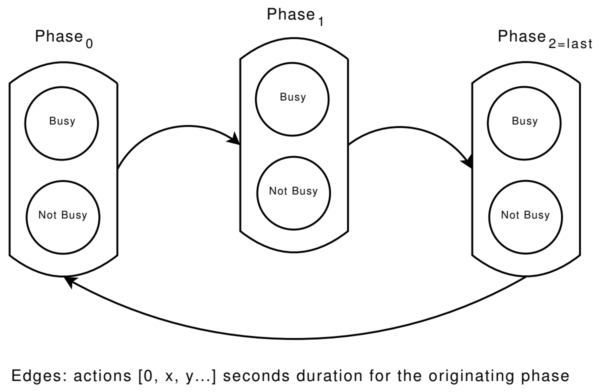

Figure 2. State-Action space for an ARR-(C)RL traffic controller

The latter commonly models traffic flow based on concepts inspired by fluid dynamics. It deals with vehicles collectively and on a homogeneous basis. On the other hand, the microscopic approach models the details of individual vehicle/driver behaviour, for instance, lane switching, vehicle following, and speed control. In addition, it differentiates between various types of vehicles, which is closer to real life. The UTC simulator [30] we use is based on the microscopic approach. 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.

4.1. Traffic Light Controller Design

We focus on experimenting with ARR-RL-based and ARR-CRL-based traffic light control scenarios that provide near-optimal policies for the UTC problem. In the ARR-RL controller, a given signalized junction's state-action space is modelled based on every available phase and its status, (i.e., busy/not busy). A given phase's status depends on all the incoming approaches of that phase. A pair of a phase and its status is considered a state in the model, see Figure 2.

A given phase's status is determined by comparing the total number of vehicles within range on its incoming approaches against a specific threshold value. The ARR-RL controller specifies 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 ARR-RL controller to skip unnecessary phases while exploring for the near-optimal policy. Furthermore, an ARR-CRL controller uses the same state-action space model as in the ARR-RL controller but allows for knowledge exchange among collaborating ARR-CRL controllers. The collaboration is governed by a specific advertisement strategy. This strategy defines the frequency of communication, the nature of communicated data, and the groups of ARR-CRL agents every controller is allowed to send to and receive from, (i.e., agents to collaborate with).

The map of Dublin's inner city centre is presented in Figure 3. The pattern of traffic we experiment with is based on 4032 vehicles uniformly inserted over ~133 minutes duration. Those vehicles are of the same type and travel in both directions between the map's opposing edges. The map comprises 64 signalized junctions which makes it a challenging UTC optimization task given the aforementioned pattern.

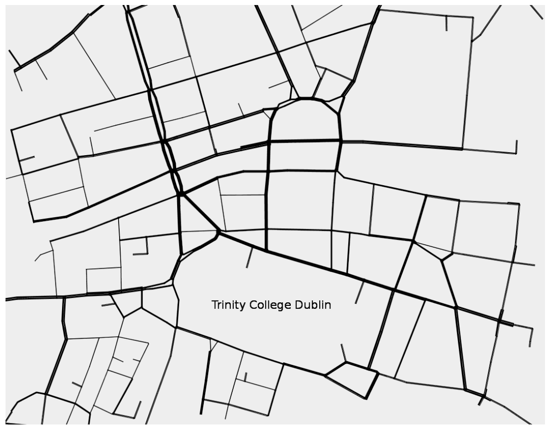

Figure 3. Dublin's inner city centre map

Our baselines for comparison are a SAT-like [31] algorithm that roughly emulates SCATS' behaviour of saturation balancing in addition to a Fixed Time Traffic Controller (FT-TC) scenario where a FT-TC is assigned to each signalized junction. The FT-TC cycles through a given junction's phases giving a fixed phase time to each. We chose to experiment with 20- and 40-second phase times assuming that 20-second is a reasonable average phase time. The SAT-like algorithm tries to achieve a 90% saturation level and uses a 20-second minimum phase time and a maximum cycle length of $[min\_phase\_time \times factor \times number\,of\,phases]$ where the factor in this case is set to 2 while the number of phases depends on the junction. Essentially, we experiment with two different scenarios, a decentralized ARR-RL scenario and an ARR-CRL scenario that share the basic design choices. Each signalized junction in the ARR-RL and ARR-CRL scenarios is assigned an ARR-RL agent or an ARR-CRL agent respectively. In both scenarios agents are designed with a set of {0, 20, 40} second actions available within their state-action space. These agents learn using Q-Learning and chose their actions based on a Boltzmann action selection technique. In the ARR-CRL scenario, agents employ a common advertisement strategy that allows a given ARR-CRL agent to exchange its rewards collected during 240 seconds. These exchanged rewards are discounted based on age using a Net Present Value (NPV) [21] inspired formula (1). The d_rate in the NPV formula is the discount rate used to diminish the significance of older rewards. An r_t is the reward obtained at index *t* in the exchanged reward vector. The most recent reward has the highest *t* value while the first has $t = 0$, hence, $0 \leq t < rv\_size$ and rv_size is the reward vector size. We set the d_rate to 0.1.

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

We design a reward model for the ARR-RL scenario based on the number of vehicles that manage to clear the junction (v_cleared) during the selected action (phase set) duration and on the number of vehicles that are still waiting (v_waiting) after the action execution. The reward is calculated locally using $r_{local} = (v\_cleared - v\_waiting)$. An ARR-CRL agent uses the same local reward model as in the ARR-RL agent to calculate the rewards that are to be exchanged with its identified neighbours. The actual reward model used by any ARR-CRL agent to update its learnt policy is hence a composite of its local reward value and the normalized discounted rewards received from its neighbours, see formula (2). In the ARR-CRL scenario, an ARR-CRL agent's send/receive neighbours are the first signalized junctions positioned up and down stream.

$$exch\_rewards = \sum_{n \in neighbours} \frac{\sum_{t=0}^{t=rv\_size_n - 1} NPV(r_{nt})}{rv\_size_n}$$

$$r_{ARR-CRL} = r_{local} + \frac{exch\_rewards}{number\,of\,neighbours} \tag{2}$$

4.2. Experimental Results

We run the simulation for 140 minutes in order to give the most recently inserted vehicles the possibility to reach their destinations. In case of the ARR-RL and the ARR-CRL scenarios the agents are bootstrapped with initial Q-values (warm models) based on explored models resulting from the simulation of three similar traffic traces (training traces) following the same traffic pattern. We conduct a number of experiments using different Q-Learning discount factors γ and learning rates α. The Boltzmann temperature is set initially to 10000 and cools down uniformly as time advances throughout the training simulation, until it reaches 1 half-way through the experiment where the warm models are extracted. The results we present in Table 1 are based on the best performing settings of γ and α for the corresponding ARR-RL and ARR-CRL bootstrapped simulations using a different traffic trace from the training traces but following the same traffic pattern. The Boltzmann temperature is fixed to 1, (i.e., exploitation) during the bootstrapped simulation. The metrics used to evaluate our results are the Average Waiting Time (AWT) and the Average Travelling Time (ATT) per arrived vehicle. The travelling time is defined as the time the vehicle spends with its speed > 0 while the waiting time is the time spent still at signalized junctions.

Metric	FT-TC 20sec	FT-TC 40sec	SAT-like	ARR-RL	ARR-CRL
Average Waiting Time (**AWT**)	276.124	822.432	422.464	232.135	180.974
Average Travel Time (**ATT**)	220.499	270.064	227.469	208.328	205.631
Number of vehicles that arrived	3858	2990	3693	3922	3936

Table 1. Average waiting and travel time per arrived vehicle in seconds and the number of vehicles that managed to arrive to their destinations

Table 2 presents the relative performance improvement achieved using the ARR-RL and ARR-CRL scenarios. Experiments show that the ARR-RL scenario results in ~15.9% and ~71.7% lower AWT compared to the FT-TC 20sec and the FT-TC 40sec respectively. The SAT-like scenario fails to outperform both ARR-RL and ARR-CRL that showed ~45% and ~57% better results in the case of their AWT respectively. On the other hand, the ARR-CRL scenario results in ~22% improvement on the ARR-RL AWT. The AWT metric shows a significant improvement in both the ARR-RL and ARR-CRL experimental results. Concerning the ATT metric, the ARR-RL scenario reduces travel time by ~5.5%, ~22.8% and ~8.4% in contrast to the FT-TC 20sec, the FT-TC 40sec and the SAT-like scenarios respectively. Moreover, ARR-CRL showed a small difference of ~1.29% better ATT performance compared to the ARR-RL scenario.

Scenario	ARR-RL	ARR-CRL
FT-TC 20sec	(15.9% , 5.5%)	(34.4% , 6.7%)
FT-TC 40sec	(71.7% , 22.8%)	(77.9% , 23.8%)
SAT-like	(45% , 8.4%)	(57% , 9.6%)

Scenario	ARR-CRL
ARR-RL	(22% , 1.29%)

*Percentages are presented in the following format:
(~Average Waiting time (AWT)% , ~Average Travel Time (AWT)%)

Table 2. Relative performance improvement of the different scenarios

In Figure 4 we present the accumulated waiting time throughout the simulation for all the vehicles in the map. The simulation's granularity is quite fine where the plotted accumulation occurs every 250 milliseconds hence the large numbers on the y-axis.

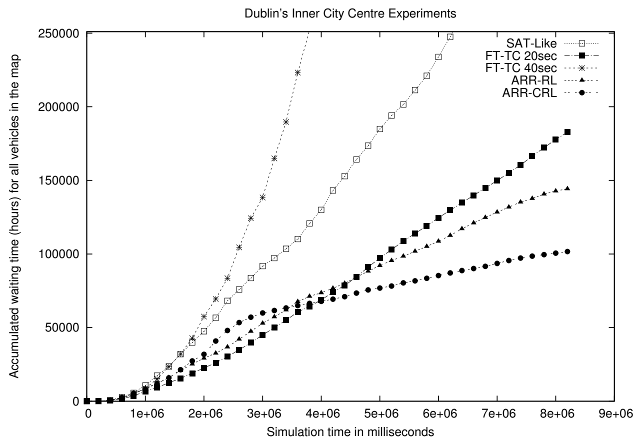

Figure 4. Accumulated waiting time

In effect, the graph reaffirms the results presented in Table 1. It is also noticeable that the FT-TC 40sec and the SAT-like accumulated waiting times for all vehicles grows out of scale. Both ARR-RL and ARR-CRL scenarios outperform the FT-TC 20sec scenario while ARR-CRL maintains lowest measures starting half way through the simulation. We believe the ability of both ARR-RL and ARR-CRL designs to skip unnecessary phases and to provide suitable timing per phase are key to their good performance.

5. Future Work

The work presented in this paper intended to establish a platform for future experiments involving uncertainty in such large-scale dynamic environments. We would like to provide an uncertainty model that could possibly deal with noisy sensor data during the exploration for better optimization. We are also exploring different collaboration techniques among intelligent agents and potentially better environmental representations. The CRL framework allows us to experiment with different action selection techniques such as ϵ-greedy which could be an interesting future comparison against Boltzmann action selection.

6. Conclusion

In this work we showed that RL and CRL are promising approaches to providing optimization solutions to dynamic environments especially the UTC problem. Moreover, we noticed that improvements are possible when collaboration is used among agents in such environments. The reduction in average waiting time per arrived vehicle achieved by both the ARR-RL and the ARR-CRL scenarios is quite significant despite the steady/uniform nature of the pattern we experimented with. One would expect that the FT-TC scenarios, given its uniform phase cycling, to perform very well with such a pattern but regardless the ARR-RL and ARR-CRL scenarios proved to be more suitable in terms of performance. The SAT-like scenario maintained better results than the FT-TC 40sec only where that was still insufficient.

Acknowledgements

The work described in this paper was partly supported by Science Foundation Ireland (as Investigator award 02/IN1/I250) and by the Irish Higher Education Authority Programme for Research in Third Level Institutions (as the Networked Embedded Systems Centre). The authors would like to thank Vinny Reynolds for his work on the UTC simulator and Mikhail Volkov for implementing the SAT-like algorithm.

References

[1] B. Abdulhai and P. Pringle. Autonomous multiagent reinforcement learning - 5gc urban traffic control. In *Annual Transportation Research Board Meeting*, 2003.

[2] B. Abdulhai, P. Pringle, and G. Karakoulas. Reinforcement learning for true adaptive traffic signal control. In *ASCE Journal of Transportation Engineering*, volume 129(3), pages 278–284, 2003.

[3] N. Ahmadabadi, M. Asadpour, and E. Nakano. Cooperative Q-learning: The Knowledge Sharing Issue. *Journal of Advanced Robotics*, 15(8):815–832, 2001.

[4] A. L. C. Bazzan. A distributed approach for coordination of traffic signal agents. *AAMAS*, 10(1):131–164, 2004.

[5] C. Bielefeldt, C. Diakaki, and M. Papageorgiou. TUC and the SMART NETS project. In *Proceedings of the IEEE ITS'01*, pages 55–60, 2001.

[6] E. Bitting and A. A. Ghorbani. Cooperative multiagent systems for the optimization of urban traffic. In *Proceedings of IAT'04*, pages 176–182, Washington, DC, USA, 2004. IEEE Computer Society.

[7] E. Camponogara and W. K. Jr. Distributed learning agents in urban traffic control. In F. Moura-Pires and S. Abreu, editors, *EPIA*, volume 2902 of *Lecture Notes in Computer Science*, pages 324–335. Springer, 2003.

[8] Y. J. Cao, N. Ireson, L. Bull, and R. Miles. Design of a traffic junction controller using classifier systems and fuzzy logic. In *Proceedings of the 6th International Conference on Computational Intelligence, Theory and Applications*, pages 342–353, London, UK, 1999. Springer-Verlag.

[9] C. Diakaki, M. Papagerogiou, and K. Aboudolas. A multivariable regulator approach to traffic-responsive networkwide signal control. *Control Engineering Practice*, 10:183–195, February 2002.

[10] V. Dinopoulou, C. Diakaki, and M. Papageorgiou. Applications of the urban traffic control strategy tuc. *European Journal of Operational Research*, 175(3):1652–1665, 2006.

[11] J. Dowling, R. Cunningham, E. Curran, and V. Cahill. Building autonomic systems using collaborative reinforcement learning. *Knowl. Eng. Rev.*, 21(3):231–238, 2006.

[12] K. Dresner and P. Stone. Multiagent traffic management: Opportunities for multiagent learning. In K. T. et al., editor, *LAMAS 2005*, volume 3898 of *Lecture Notes In AI*. Springer Verlag, Berlin, 2006.

[13] European-Commission. European transport policy for 2010 : time to decide. Brussels, Belgium, 2001.

[14] European-Commission. On the intelligent car initiative - raising awareness of ICT for smarter, safer and cleaner vehicles. Brussels, Belgium, 2006.

[15] J. Farges, J. Henry, and J. Tuffal. The PRODYN realtime traffic algorithm. In *Proceedings of the IEE International Conference on Road Traffic Signalling*, pages 307–312, 1983.

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

[17] S. P. Hoogendoorn and P. H. L. Bovy. State-of-the-art of vehicular traffic flow modelling. volume 215, pages 283–303(21), 19 August 2001.

[18] P. B. Hunt, D. I. Robertson, and R. D. Bretherton. The SCOOT on-line traffic signal optimization technique. *Traffic Eng. Control*, 23:190–192, 1982.

[19] P. K.Fehon. Adaptive traffic signals are we missing the boat? In *ITE District 6 Annual Meeting*. DKS Associates, 2004.

[20] C. O. Kim, Y. Park, and J.-G. Baek. Optimal signal control using adaptive dynamic programming. In O. G. et al., editor, *ICCSA (4)*, volume 3483 of *Lecture Notes in Computer Science*, pages 148–160. Springer, 2005.

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

[22] P. Lowrie. SCATS: The sydney co-ordinated adaptive traffic system-principles, methodology, algorithms. In *Proceedings of the IEE International Conference on Road Traffic Signalling*, pages 67–70, 1982.

[23] S. Mikami and Y. Kakazu. Genetic reinforcement learning for cooperative traffic signal control. In *ICEC*, pages 223–228, 1994.

[24] P. Mirchandani and L. Head. A real-time traffic signal control system: architecture, algorithms, and analysis. *Transportation Research Part C: Emerging Technologies*, 9:415–432(18), December 2001.

[25] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *AAMAS*, 11(3):387–434, 2005.

[26] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialosa, and Y. Wang. Review of road traffic control strategies. In *Proceedings of the IEEE*, volume 91, pages 2043–2067, December 2003.

[27] M. D. Pendrith. Distributed reinforcement learning for a traffic engineering application. In *AGENTS'00*, pages 404–411, New York, NY, USA, 2000. ACM Press.

[28] J. Peters, S. Vijayakumar, and S. Schaal. Natural actor-critic. In *ECML*, pages 280–291, 2005.

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

[30] V. Reynolds, V. Cahill, and A. Senart. Requirements for an ubiquitous computing simulation and emulation environment. In *InterSense'06*, page 1, New York, NY, USA, 2006. ACM.

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

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

[33] S. Sen and L. K. Head. Controlled optimization of phases at an intersection. *Transportation Science*, 31(1):5–17, 1997.

[34] Z. sheng Yang, X. Chen, Y. shan Tang, and J. ping Sun. Intelligent cooperation control of urban traffic networks. In *Proceedings of 2005 ICMLC*, volume 3, pages 1482–1486, 2005.

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

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

[37] M. Tan. Multi-agent reinforcement learning: independent vs. cooperative agents. In *Readings in agents*, pages 487–494, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[38] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

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

[40] M. Wiering, J. Veenen, J. Vreeken, and A. Koopman. Intelligent traffic light control. 2004.