Using Reinforcement Learning for Multi-Policy Optimization in Decentralized Autonomic Systems - An Experimental Evaluation

Ivana Dusparic and Vinny Cahill

Lero – The Irish Software Engineering Research Centre
Distributed Systems Group
School of Computer Science and Statistics
Trinity College Dublin, Ireland
{ivana.dusparic, vinny.cahill}@cs.tcd.ie

Abstract. Large-scale autonomic systems are required to self-optimize with respect to high-level policies, that can differ in terms of their priority, as well as their spatial and temporal scope. Decentralized multi-agent systems represent one approach to implementing the required self-optimization capabilities. However, the presence of multiple heterogeneous policies leads to heterogeneity of the agents that implement them. In this paper we evaluate the use of Reinforcement Learning techniques to support the self-optimization of heterogeneous agents towards multiple policies in decentralized systems. We evaluate these techniques in an Urban Traffic Control simulation and compare two approaches to supporting multiple policies. Our results suggest that approaches based on W-learning, which learn separately for each policy and then select between nominated actions based on current action importance, perform better than combining policies into a single learning process over a single state space. The results also indicate that explicitly supporting multiple policies simultaneously can improve waiting times over policies dedicated to optimizing for a single vehicle type.

1 Autonomic Systems

Autonomic computing systems are systems that self-manage and self-adapt to varying circumstances without human intervention [8]. The need for autonomic capabilities arises due to the increasingly large scale, decentralization and complexity of computing systems rendering the traditional manual, centralized and hierarchical approaches to system management infeasible [21]. Autonomic systems are only given high-level objectives while the details of how to meet those

objectives are left up to the systems themselves. Therefore, autonomic systems need to self-optimize their performance, even in the changing environment conditions. Rather than being managed by a central component, an autonomic system can be modelled as a group of autonomic elements, that are capable of sensing their environment and making their own local decisions [8]. The optimal local decisions cannot be predefined for each situation in which an autonomic element might happen to be, but is often required to be learnt by the element itself. As autonomous agents [17] have the capabilities required by autonomic elements, it is believed that multi-agent systems are a suitable technique for the implementation of autonomic behaviour [21]. Examples of such techniques already successfully applied in decentralized large-scale susyems include ant colony optimization in load balancing [10], particle swarm optimization in wireless networks [7], evolutionary computing in routing [5] and reinforcement learning (RL) in load balancing [4].

1.1 Multi-policy optimization

The systems mentioned above focus on optimizing system performance with respect to only a single high-level goal. However, autonomic systems might often be required to meet multiple goals simultaneously. These goals can be expressed as system policies, which are used to guide system behaviour. Therefore, optimization techniques need to be able to address multiple goals (or policies) simultaneously. We hypothesize RL might be a suitable basis for the implementation of such a technique, as it has already been successful as a learning technique for optimization towards a single policy in decentralized systems, as well as a learning technique for multiple policies on a single agent (see Section 2). We test our hypothesis in a simulation of an Urban Traffic Control (UTC) system. We believe UTC systems are representative of large-scale autonomic systems, as existing centralized techniques are failing to deal with the pressure of high traffic loads and new decentralized adaptive learning techniques are being investigated to deal with increasing traffic congestion (see Section 2). UTC systems may also need to optimize for multiple policies that have different characteristics. The policies can often be conflicting, highly dependent on one another, have different levels of priority, and different spatial and temporal scope.

Policy classification We use three main criteria to classify the characteristics of policies in decentralized systems:

- priority - can range from low to high, based on how important it is for a system to meet this particular goal in relation to meeting its other goals;
- spatial scope - can be local, regional, and global, based on the area of a system over which a policy is implemented and its performance measured;
- temporal scope - can be continuous or temporary (sporadic), based on whether a system is required to work towards this goal continuously during its operation, or only occasionally under a certain set of conditions.

We illustrate the classification with a few examples of policies from UTC. The main task of a UTC system is to optimize global traffic flow in the system, by minimizing travel and waiting times for all vehicles in the system. This policy is classified as global (as it affects the whole system), continuous (as UTC systems need to implement this policy as long as there are any vehicles present in the system), and of a standard priority. Occasionally, emergency vehicles, such as ambulances, fire engines, or police cars, appear in the UTC system, and the system's task is to give them priority over other vehicles in the system. This policy that prioritizes emergency vehicles is said to be regional (as it affects only the region in which the emergency vehicle is travelling, generally major traffic routes), sporadic (as emergency vehicles are not always present in the system but only when the need arises), and it has a high priority (since it is more important to meet this policy than to minimize the travel time for other vehicles on non time-critical journeys). Policies can also be local, where, for example, at a very busy pedestrian crossing, pedestrians may be given priority over vehicles.

Agent heterogeneity and dependency This wide variety of policies and their characteristics leads immediately to heterogeneity of the agents that implement them. For example, consider Figure 1. Agent A might be in charge of contributing to the implementation of a global policy P_a, together with all of the other agents in the system. Agent B could also be in charge of contributing to the implementation of a policy P_b , implemented only by agent B and its neighbours, while agent C could also be in charge of a local policy P_c, being the only agent implementing it. These policies can be concerned with addressing different parts of the environment, e.g., P_a might only be interested in optimizing travel time for cars, while P_c might only be dealing with pedestrians. In terms of RL, this will cause the state spaces of agents A, B, and C to differ, as, for example, the information required to be encoded in the state space of a policy optimizing for cars will need to be different from the information relevant to the policy optimizing for pedestrians.

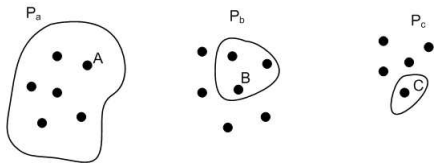

Fig. 1. Agent heterogeneity

Furthermore, agent heterogeneity can also arise from the differences in the agents' environments and capabilities. For example, in a UTC system, the layout of junctions can differ; each junction can have a different number of approaches and exits, resulting in a different set of traffic-light phases being possible at that junction. In an RL implementation this maps to agents having different state

spaces as well as different action sets. A junction with two approaches and one exit will have a significantly smaller state space and action set than a junction with four approaches and four exits. The size of the state space and the number of possible actions directly influence the duration of a learning process, so agents will significantly differ in the number of learning steps that it takes them to learn what they consider to be the optimal action for each state visited.

Agents acting in a shared environment may potentially be highly dependent, i.e. affected by each other's actions. In UTC, agents share the same road network with limited road space. Therefore, any decision that a traffic-light agent makes might have direct consequences on its neighbouring agents, and by extension, on most of the other agents in the system. For example, if a backlog of traffic is left uncleared at an approach, the queue can spill over to the upstream junction. The traffic will not be able to go through the upstream junction regardless of the actions taken by an agent controlling it, as there is no road space available.

Such dependencies are particularly difficult to deal with in environment controlled by heterogeneous agents. Some agents could be contributing towards optimizing traffic flow, while others are contributing to prioritizing emergency vehicles. However, cars and emergency vehicles share the same road network, and therefore the performance of agents in implementing one policy will directly influence the performance towards the other. For example, if an agent that is implementing emergency vehicle prioritization releases an approach at which an emergency vehicle is waiting, it might create a traffic backlog on one of its other approaches, negatively affecting the junction upstream from that approach.

In summary, large-scale autonomic systems can consist of multiple agents, implementing multiple, dependant, and potentially conflicting policies, where these policies differ between agents, causing agents to have different state spaces and different action sets. All of these issues will need to be addressed by RL techniques that are to be applied to large-scale autonomic systems and UTC in particular.

1.2 Reseach question

The goal of this study is to assess the suitability of multi-agent RL-based techniques for optimization in autonomic systems. In order to do so, we have implemented and evaluated several single and multi-policy UTC scenarios. We use single-policy scenarios to evaluate the impact that policies targeted at one vehicle type have on other vehicle types, as well as baselines for the evaluation of multi-policy scenarios.

As policy heterogeneity is a central issue, for the initial evaluation we selected two policies that differ in all three of our classification criteria: priority, temporal scope, and spatial scope. The single-policy scenarios we implemented are as follows:

1. Global Waiting time Only (GWO) - a global, continuous, standard-priority policy that aims to optimize waiting time for all the vehicles in the system.

2. Emergency Vehicles Only (EVO) optimization - a regional, temporary, high-priority policy that aims to prioritize emergency vehicles only.

We combined the policies above in two ways to implement the following multi-policy scenarios:

1. Combined state space (GWEV-c), where GWO and EVO are combined into a single learning process over a single state space.
2. W-Learning (GWEV-w), where GWO and EVO learn the best actions separately as two separate learning processes, but W-learning (see Section 2.1) is used to determine which action is to be executed.

One important consideration when addressing the agent dependency and heterogeneity in large-scale autonomic systems is whether agents should act independently (contributing only to implementing policies for which they are responsible), or whether they should collaborate with other agents (contributing to the implementation of policies that they are not directly responsible for as well). In the experiments we describe in this paper, we implement only independent agents. The scenarios above are designed to evaluate approaches for dealing with multiple heterogeneous policies, while in the future, we also plan to evaluate the impact of collaboration in multi-agent multi-policy approaches.

The rest of this paper is organized as follows. In Section 2 we give background on RL as well as its applications in UTC system. Section 3 describes our simulation environment. Section 4 describes the details of the scenarios that we implemented and the design of the agents, followed by the results and their analysis. Section 5 concludes the paper and outlines future work.

2 Background

Reinforcement Learning [20] is an unsupervised learning technique whereby an agent learns how to meet its goal by interacting with the environment. Agents sense their environment, map their observations to a state space representation, execute an action and obtain a reward from the environment based on the suitability of that action in the given state. Therefore, a reward is the only guidance agents have when learning how to meet their objectives. We are particularly interested in Q-learning implementations of RL [20], because, as we'll see later in this section, it has already been successfully applied to certain types of UTC problems. In Q-learning, an agent uses a value function to estimate the accumulated future reward. In this way, agents learn to perform the actions with the highest long-term reward, rather than those that merely receive the highest immediate reward. Performance of a Q-learning process depends on the action selection strategy used. In our experiments we use Boltzmann [20] action selection, which uses a temperature parameter to determine the ratio of exploration and exploitation in the Q-learning process. The speed of learning and the weight given to recent vs. older actions are determined by two additional parameters, the learning rate α, and a discount factor γ, respectively.

2.1 Multi-goal Q-learning

Q-learning implementations can deal with the presence of multiple policies on a single agent in several ways.

Humphrys [6] introduced W-learning, where policies not only learn appropriate actions for each state, but also how important it is to that policy for that particular action to be executed, in comparison to an action that is best for some other policy. The action with the highest relative importance gets executed.

In contrast to W-learning, multiple goals can also be combined into a single learning process. However, on a larger scale this is prone to state explosion. One way to deal with this is to reduce the state space by eliminating states that are unlikely or impossible to occur, if there are any [3].

Nataraj et al. [11] deal with learning in the situations where relative weights of policies change over time. Shelton provides a means to balance the incomparable rewards received from multiple sources [19].

All of these multi-policy techniques have only been applied to a single agent.

2.2 Agent-Based Traffic Control Strategies

A large body of research exists showing the suitability of multi-agent systems, in particular those implementing RL, for optimization of performance of UTC systems. An increasing number of UTC systems are being managed by traffic-responsive algorithms, such as SCATS [13], however research shows that use of RL can outperform these algorithms as well.

In [22], cars estimate their projected waiting time, and make a decision about their route based on this. Projected waiting time is communicated to the traffic lights agents, modelled as Q-Learning agents, that then select the phase that minimizes waiting times. Wiering's experiments show improved performance of his approach over a fixed-time controller. Abdulhai et al. [1] model traffic-lights as Q-learning agents and conclude that Q-learning provides higher real-time adaptivity than other state-of-the-art techniques. Pendrith [14] models vehicles as Q-learning agents capable of sensing the speed and position of their neighbouring vehicles. Based on this information, vehicles decide on their speed and potential lane changes.

In the above examples, RL agents are implemented to act independently. Bazzan introduces traffic-light agent coordination using evolutionary game theory [2], while Salkham et al. [18] implement agent collaboration by a means of Q-value exchange between neighbouring agents.

All of the implementations mentioned are concerned with a single policy of optimizing traffic in general, whether by increasing throughput or by minimizing waiting time. Much less work is dedicated to the applicability of multi-agent systems, and RL in particular, for optimization towards other UTC policies. Oliveira and Duarte [12] incorporate emergency vehicle priority into their UTC system. Meignan et al. [9] simulate bus network performance, modelling both buses and passenger behaviour. They account for the influence of other traffic on the bus network, but do not account for the influence of public transport on other vehicles, nor do they include traffic signal priority for public transport.

2.3 Summary

RL is a single-agent, single-policy learning technique. It has been extended to deal with multiple policies on a single agent, as well as to multiple agents implementing a single policy, either independently or collaboratively. However, no RL technique deals with multiple policies on multiple agents simultaneously. Existing RL applications in UTC also concentrate on a single policy only. Our work has been inspired by the success of such applications in optimization of traffic waiting times, emergency vehicle priority, and bus networks performance individually, as well as by the lack of an integrated approach for optimizing for all policies and vehicle types simultaneously. We evaluate the use of RL techniques in multi-agent environments in dealing with multiple UTC policies, while simultaneously modelling the influence that these policies have on one another's performance.

3 UTC Simulation Platform

In our experiments we use an urban traffic simulator developed in Trinity College Dublin [15]. The simulator uses a microscopic traffic simulation approach, and can simulate traffic over any road network defined by a map provided in an XML format. The simulator can distinguish between multiple vehicle types, such as cars, public transport vehicles, and emergency vehicles. Vehicles implement different behaviours based on their type; e.g. emergency vehicles are capable of driving above the allowed speed limit, as well as driving through red lights if it is safe to do so.

The map we used for our initial experiments presented in this paper is shown in Figure 2. The map is based on road layout details provided by Dublin City Council and corresponds to one of the busiest areas of Dublin's road network, O'Connell Street, Dublin's main street, and several side roads that feed traffic onto this road. Using a real-world map provides a more realistic simulation; many of the simulations used for evaluation of multi-agent systems in UTC covered in Section 2.2 use either a single junction, or multiple junctions and road links that have similar layouts, while the map that we use includes junctions of various layouts (e.g. junctions with two, three, and four approaches and exits), roads of differing width (e.g. two, three, and four lane roads), as well as one-way and two-way roads. The map covers 8 junctions, 5 of which are signalised junctions and are controlled by the agents described in the following section. Each agent has a set of available phases, or combinations of compatible red and green settings on all traffic lights controlling one intersection. Phases are generated based on intersection layout and allowed traffic directions. Each phase is mapped to an action agent is able to execute.

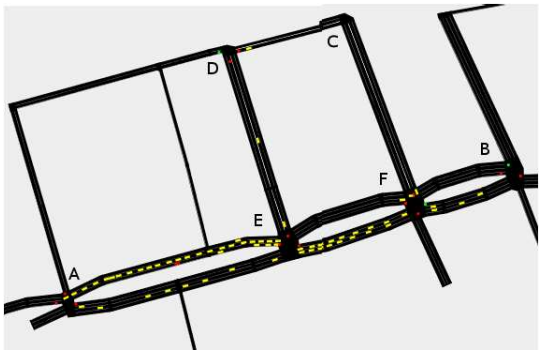

Fig. 2. UTC Simulator

4 Traffic Light Agents

In this section we describe the agents that we designed to implement the policies and study the RL approaches to be evaluated (as listed in Section 1.2), as well as agents used as a basis for comparison.

4.1 Baseline agents

Round robin As a baseline used for the evaluation of the performance of the RL agents we ran experiments using a Round Robin (RR) junction controller. The RR agent, at each junction, continuously loops through all available phases at that junction. The duration of each phase is 20 seconds.

SAT We also compare the performance of the RL agents to a simple SCATS-like traffic-responsive algorithm SAT, as defined by Richter [16], that adjusts phase duration based on the degree of saturation at a junction. The degree of saturation is defined as a ratio of the effectively used green time to the total available green time. At each junction, SAT, similarly to SCATS, aims to keep the junction saturation as close to 90% as possible, by shortening or lengthening the phase duration. In our experiments the minimum duration for each phase is set to 20 seconds.

4.2 Single-policy RL scenarios

GWO - Optimizing global waiting time The first policy we implemented, optimizing Global Waiting time Only (GWO), optimizes vehicle waiting time in the whole system. Since global waiting time is a sum of waiting times for all cars at all junctions in the system, and we assume no collaboration between agents, we aim to minimize the waiting times at each individual junction.

Each agent is capable of sensing the number of vehicles at each of its approaches, and maps that to a state space that orders approaches according to

their congestion. For example, on a junction with two approaches, a_1 and a_2, a state could be "Congestion order: a_1, a_2", meaning that approach a_1 has more traffic waiting than a_2. Note that, since junctions have different layouts, the size of the state space will depend on the number of approaches. The state space does not encode how many vehicles are waiting at which approach as the numbers are relative to overall congestion in the system. It also contains information about whether the total number of vehicles waiting at the junction is more or less than at the previous phase change (e.g. "Congestion order: a_1, a_2, less vehicles than before"). Note that arrival rates are assumed to be uniform in this experiment, so a change in numbers of vehicle waiting is caused only by an agent's action. Such a state space is created to facilitate rewarding an agent (100 points in our experiments) for being in a state with less traffic waiting than at the previous decision point, i.e. to motivate it to execute actions that clear more traffic than arrives at the junction during the action execution. Agents learn to reduce the number of vehicles waiting at the junction's approaches, thus reducing global waiting time for the system.

EVO - Prioritizing emergency vehicles The other single RL policy that we implemented minimizes waiting times for Emergency Vehicles Only (EVO). The agents' state space encodes information about which approach(es), if any, have emergency vehicles waiting. Agents receive a reward (200 points in our experiments) for being in a state where there is no emergency vehicle present at any of the approaches. This encourages agents to, as soon as possible, return to the state with no emergency vehicle present, by enabling emergency vehicles to pass. This policy does not address any other vehicle types and only takes emergency vehicles into account when making action decisions.

4.3 Multi-policy RL scenarios

GWEV-c: Merging RL processes One way to combine multiple policies on a single agent is to encode all the information relevant for all the policies into a single state space and a single learning process. We combined GWO and EVO into a single policy, GWEV-combined (GWEV-c). The state space of GWEV-c consists of the cross product of the state spaces for GWO and EVO. An agent receives 100 points reward for being in any of the states with less traffic than in the previous phases (i.e. states for which GWO receives a reward for), a 200 points reward for any of the states with no emergency vehicles present (i.e. states for which EVO receives a reward for), and the sum of both rewards for being in a state that satisfies both criteria. We acknowledge that as the number of policies to be combined increases this approach will not be scalable due to state space explosion, but we believe that comparing its performance to other techniques can give us a useful insight into how multiple policies should be dealt with.

GWEV-w: W-learning W-Learning is a multi-goal technique proposed in [6] that builds on Q-Learning. First, each agent runs separate Q-Learning process for each policy that it is implementing. In our experiments, we ran the two

individual single goal policies described in the previous section, GWO and EVO, and on top of them implement W-learning (GWEV-w). After GWO and EVO have learnt Q-values for their state-action pairs, the process of W-learning starts. In W-learning, an agent learns how important it is that, for each of its policies and for each state in which an agent could be, the action a policy nominates is in fact executed, i.e. what weight that action carries. W-values are updated based on the reward received, and further action selection is based on these W-values. Each policy nominates an action, based on its Q-values, together with an associated W-value for the state in which the agent is currently. The action proposed by a policy with the highest W-value is executed. In our experiments, since EVO is a temporary policy, we deem it inactive when there are no emergency vehicles present, and set the weight of the action that the EVO policy nominates in that state to zero.

4.4 Simulation setup

Cars enter the simulation at four different points (A, B, C, D) and exit the system at two different points (A, B), following 1 of 4 paths: A to B, B to A, C to A, and D to B (see Figure 2). Emergency vehicles tend to use major routes wherever possible, so in our simulation they only travel on paths A to B, and B to A. Therefore, the EVO policy is only deployed on agents A, B, E, and F. All vehicles follow the shortest path from source to destination. Vehicle routes are the same for all of the experiments we ran.

Agent performance is tested for three different traffic loads to simulate different traffic conditions. The loads are as follows: low load (a total of 28,140 vehicles are inserted, 7,000 cars on each of the car routes and 70 ambulances on each of the emergency vehicle routes), medium load (a total of 56,280 vehicles, 14,000 cars on each of the car routes and 140 ambulances on each of the emergency vehicle routes) and high load (a total of 100,500 vehicles, 25,000 cars on each of the car routes and 250 ambulances on each of the emergency vehicle routes).

Each signalised junction in the simulation has a different set of available phases, automatically generated based on junction layout. For this set of experiments, the duration of each phase is set to 20 seconds. Junctions can cycle through their available phases using RR, or can be controlled by SAT or one of the RL agents described in the previous section.

4.5 Experiment Parameters

Each of our RL experiments is run in two parts: 2010 simulation minutes of exploration, and 2010 minutes of exploitation. The duration of 2000 minutes enables Q-learning to execute 6000 learning steps (as our actions are 20 seconds duration each) which, we consider sufficient for agents to learn the Q-values for their state-action pairs. Additional 10 minutes were added to allow a chance for last inserted vehicles to leave the system. GWEV-c has a much larger state space than the other policies and therefore was given a longer exploration phase of 20000 minutes to enable a larger portion of the state space to be visited a

sufficient number of times. Each experiments is repeated three times, and average results from exploitation phase are presented in this paper.

Each RL process has been run multiple times to determine the best combination of α and γ . The final combinations used for the experiments presented are, for GWO: $\alpha = 0.1$ and $\gamma = 0.3$, for EVO: $\alpha = 0.9$ and $\gamma = 0.1$, for GWEV-c: $\alpha = 0.1$ and $\gamma = 0.1$ and for GWEV-w: $\alpha = 0.1$ and $\gamma = 0.7$.

The performance of SAT also varies based on the size of the steps in which the phase duration can be incremented or decremented, as well as the maximum duration of the cycle factor. The actual maximum duration of the cycle for a junction is a function of this factor and the number of available phases for that junction. The best parameters determined for SAT performance with a minimum action duration of 20 seconds are 10 for the increment step, and 1.2 for the maximum duration of the cycle factor.

4.6 Results and Analysis

Metrics We compared the performance of the RL agents based on the following metrics:

- Density - measured as the ratio of occupied road space to available road space [2]. For the same traffic arrival rate, higher density means worse agent performance, since traffic that is not successfully cleared and is still in the system is creating higher density.
- Waiting time - average waiting time per vehicle for the duration of the experiment. We separate waiting times per vehicle type, so we can measure performance towards each of individual policies described in Section 4.2.

Density Density results are summarized in Table 1.

	RR	SAT	GWO	EVO	GWEV-c	GWEV-w
Low	2.96	2.76	1.66	12.30	1.60	1.49
Medium	5.60	5.20	3.31	11.37	3.47	3.09
High	11.04	9.50	5.84	15.85	6.03	5.06

Table 1. Average density per load level ratio

We see that GWEV-w has the lowest density across all three loads, indicating that it is the most successful approach to managing the general traffic flow. GWO and GWEV-c have similar densities, with GWEV-c being better at the low load, and GWO at all other loads. We believe this is due to GWO addressing cars, which make up 99.5% of the total traffic, so its performance is very close to GWEV-c, which addresses all traffic. At the low loads, SAT and RR perform similarly, while the difference becomes more obvious at high load, where SAT performs better.

EVO has by far the highest density for all three loads. We believe this is due to the fact that this policy addresses only emergency vehicles, which make up only 0.5% of traffic in our simulation. Cars, which make up remaining 99.5% of the traffic, are not addressed, and create a backlog in the system. In Figure 3 we see an example of the effect of this backlog on the density. In the EVO implementation of the UTC system fills up with the traffic not adequately addressed by the policy, creating higher density and worse performance. This confirms the high dependency between policies due to the shared infrastructure, i.e. road space.

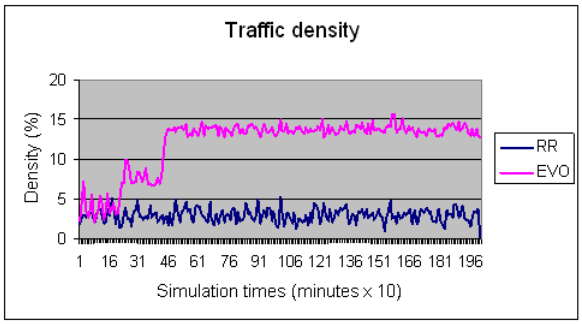

Fig. 3. Density during low load

Emergency-vehicle waiting time Our initial expectations were that EVO, whose only goal is to prioritize for emergency vehicles, would yield the best waiting times for emergency vehicles. However, this approach turned out to have the worst performance due to the high dependency between emergency vehicle performance and the performance of private vehicles which are not addressed by this policy. Not only do emergency vehicles suffer high waiting times, but a large number of vehicles had to be turned away, as the system was backlogged and there was no available road space for them to join. For this reason, EVO waiting time results are not comparable to other results and we exclude them from subsequent graphs.

Figure 4 shows average waiting times for both cars and emergency vehicles for the medium traffic load for all the policies we implemented apart from EVO. For the moment, we focus on the emergency vehicle waiting times. On the graph shown, GWEV-w is the best policy for emergency vehicles, but at low and high loads, GWO slightly outperforms this policy. The similar performance of single-policy GWO and multi-policy GWEV-w suggests the high dependency between the performance of different vehicle types. It emphasises the importance of clearing general traffic, as GWO does, to free up the road space for emergency vehicles, and suggests a high dependency between a policy that addresses emergency vehicles and one that addresses private vehicles.

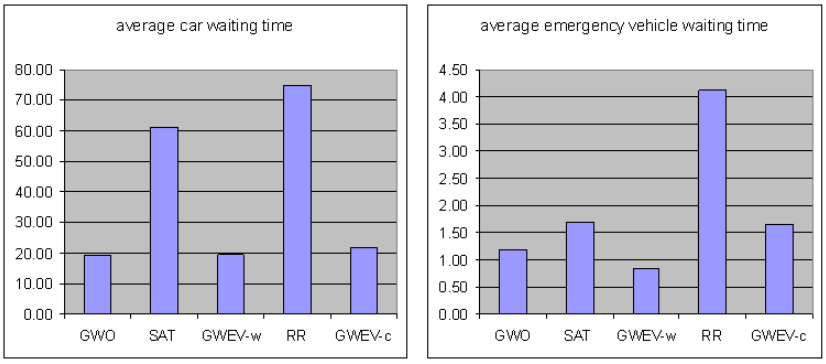

Fig. 4. Average waiting time per vehicle type

GWEV-w clearly outperforms GWEV-c at all loads, reducing emergency vehicle waiting time to between ~40% and ~80% of their waiting time in GWEV-c. We believe that worse performance of GWEV-c is caused by the size of the agents' state spaces. These results suggest that GWEV-c, even though it still outperforms SAT and RR, is not effective even for combinations of only two policies and would not be scalable to the addition of any further policies.

GWEV-w also outperforms our baselines, reducing average emergency vehicle waiting time to between ~7% and ~50% of their waiting time in SAT, and between ~7% and ~20% of their waiting time in RR. These results suggest that GWEV-w is a suitable technique for multi-policy optimization in UTC, as it outperforms the other evaluated multi-policy technique GWEV-c, single policy EVO, as well as both of our baselines.

Car waiting time GWO, GWEV-w and GWEV-c have similar performance for all loads in terms of average car waiting time, with GWEV-w slightly outperforming GWEV-c, and GWO slightly outperforming both of the multi-policy approaches (see Figure 4). From this we conclude that the best waiting times for cars are achieved when the system optimizes only for cars (GWO), but in the presence of multiple policies, specifically one with a higher priority such as emergency vehicles, GWEV-w is the best approach. GWEV-w also outperforms our baselines, and reduces waiting time for cars to between ~32% and ~42% of their waiting time in SAT and between ~26% and ~38% of car waiting time in RR.

It is also interesting to observe the performance of our baselines, SAT and RR, in relation to each other, both in terms of car waiting time and emergency-vehicle waiting time. At the medium load (see Figure 4) and at the high load, the adaptive SAT algorithm performs better, as we expected, but at the low load RR actually performs better. This indicates that when the loads in the system are very low, running an adaptive algorithm, SAT, might have adverse effects on traffic performance, possibly due to extending phase times to longer than it

is required and creating larger backlogs. However, these results also emphasize the importance of adaptation at higher loads.

Overall analysis From the experiments we performed we have made following main observations. Both RL-based techniques, GWEV-w and GWEV-c, outperform our baselines, both in terms of emergency vehicle and car waiting times, showing that RL-based techniques are promising approaches to multi-policy optimization in autonomic systems. GWEV-w performs better than GWEV-c, indicating that W-learning-based approach is a more suitable approach for multi-policy optimization than combining learning processes into a single learning process. We also observe high dependency between the policies reflected in their performance. The policy that addresses only emergency vehicles (EVO) generates a backlog of other vehicles, and as a result performs very badly both in terms of car and emergency-vehicle waiting times. GWO, which addresses only cars, also performs well in terms of emergency vehicle waiting times, as clearing cars creates less congested roads and enables emergency vehicles to proceed. Our results also show that the importance of the optimization increases with the traffic load, where the gap between the performance of adaptive techniques (e.g. SAT) and nonadaptive techniques (e.g. RR) grows larger.

5 Conclusions and Future Work

In this paper we presented the challenges of multi-policy optimization in decentralized autonomic systems. We have evaluated several proposed multi-policy optimization RL techniques in UTC and our results indicate that W-learning is a suitable approach for optimization towards multiple policies in multi-agent heterogeneous autonomic environments. In future work, we will extend the scope of our experiments to additional policies with different characteristics, to establish wider applicability of W-learning-based techniques. We will also investigate potential for performance improvement by agent collaboration by enabling W-learning agents to cooperate with each other in order to meet system goals.

6 Acknowledgements

This work was supported by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre (www.lero.ie). The authors would also like to thank As'ad Salkham for his work on the implementation of RL libraries used in our evaluation, and Vinny Reynolds, Anurag Garg, Raymond Cunningham, Mikhail Volkov, and Sylvain Cabrol for their work on the traffic simulator.

References

1. B. Abdulhai, R. Pringle, and G. Karakoulas. Reinforcement learning for the true adaptive traffic signal control. *Journal of Transportation Engineering*, 129(3):278–285, May/June 2003.

2. A. L. Bazzan. A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems*, 10(1):131–164, 2005.
3. H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira. Learning multi-goal dialogue strategies using reinforcement learning with reduced state-action spaces. In *Int. Journal of Game Theory*, pages 547–565, 2006.
4. J. Dowling. *The Decentralised Coordination of Self-Adaptive Components for Autonomic Distributed Systems*. PhD thesis, Trinity College Dublin, 2005.
5. L. He and N. Nort. Hybrid genetic algorithms for telecommunications network back-up routeing. *BT Technology Journal*, 18(4), October 2000.
6. M. Humphrys. *Action Selection methods using Reinforcement Learning*. PhD thesis, University of Cambridge, 1996.
7. B. A. Kadrovach and G. B. Lamont. A particle swarm model for swarm-based networked sensor systems. In *SAC*, pages 918–924, 2002.
8. J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, January 2003.
9. D. Meignan, O. Simonin, and A. Koukam. Simulation and evaluation of urban bus-networks using a multiagent approach. *Simulation Modelling Practice and Theory*, 15(6):659–671, July 2007.
10. A. Montresor, H. Meling, and O. Baboglu. Messor: Load-balancing through a swarm of autonomous agents. In G. Moro and M. Koubarakis, editors, *Proc. of the 1st Workshop on Agent and Peer-to-Peer Systems (AP2PC'02)*, number 2530 in Lecture Notes in Artificial Intelligence, pages 125–137, Bologna, Italy, 2003. Springer-Verlag.
11. S. Natarajan and P. Tadepalli. Dynamic preferences in multi-criteria reinforcement learning. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 601–608, New York, NY, USA, 2005. ACM.
12. E. Oliveira and N. Duarte. Making way for emergency vehicles. In *Proc. of the 2005 European Simulation and Modelling Conference*, pages 128–135.
13. M. Papageorgiou, C. Diakaki, and V. Dinopoulou. Review of road traffic control strategies. *Proc. of the IEEE*, 91(12), December 2003.
14. M. D. Pendrith. Distributed reinforcement learning for a traffic engineering application. In *AGENTS '00*, pages 404–411, New York, NY, USA. ACM Press.
15. V. Reynolds, V. Cahill, and A. Senart. Requirements for an ubiquitous computing simulation and emulation environment. In *InterSense '06*, NY, USA. ACM.
16. S. Richter. Learning traffic control - towards practical traffic control using policy gradients. Technical report, Albert-Ludwigs-Universitat Freiburg, 2006.
17. S. Russell and P. Norvig. *Aritifical Intelligence - A Modern Approach*. Prentice Hall, 2003.
18. A. Salkham, R. Cunningham, A. Garg, and V. Cahill. A collaborative reinforcement learning approach to urban traffic control optimization. In *International Conference on Intelligent Agent Technology*, December 2008.
19. C. R. Shelton. Balancing multiple sources of reward in reinforcement learning. In *Neural Information Processing Systems-2000*, pages 1082–1088, 2000.
20. R. S. Suton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book. The MIT Press, Cambridge, Massachusetts, 2002.
21. G. Tesauro, D. M. Chess, W. E. Walsh, R. Das, A. Segal, I. Whalley, J. O. Kephart, and S. R. White. A multi-agent systems approach to autonomic computing. In *AAMAS '04*, pages 464–471.
22. M. Wiering. Multi-agent reinforcement learning for traffic light control. In *Proc. of 17th Int. Conf. on Machine Learning*, pages 1151–1158. Morgan Kaufmann, San Francisco, CA, 2000.