Distributed W-Learning: Multi-Policy Optimization in Self-Organizing Systems

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

Abstract*—Large-scale agent-based systems are required to self-optimize towards multiple, potentially conflicting, policies of varying spatial and temporal scope. As a result, not all agents may be implementing all policies at all times, resulting in agent heterogeneity. As agents share their operating environment, significant dependencies can arise between agents and therefore between policy implementations. To address self-optimization in the presence of agent heterogeneity, policy dependency and the lack of global knowledge that is inherent in large-scale decentralized environments, we propose Distributed W-Learning (DWL). DWL is a reinforcement learning (RL)-based algorithm for collaborative agent-based self-optimization towards multiple policies, which relies only on local interactions and learning. We have evaluated the DWL algorithm in a simulation of a self-organizing urban traffic control (UTC) system and show that using DWL can improve the performance of multiple policies deployed simultaneously, even over corresponding single-policy deployments. For example, in UTC, optimizing simultaneously for cars and public transport vehicles reduces the waiting times of cars to 78% of their waiting times in the best-performing single-policy deployment that optimizes for cars only, while also outperforming the widely-deployed round-robin and saturation-balancing traffic controllers that we used as baselines[1].

I. Introduction

As computing systems become increasingly large-scale and geographically dispersed, traditional centralized and hierarchical management of such systems becomes intractable. Such large-scale decentralized systems also need to be able to adapt to changing operating conditions, not all of which can be anticipated at design time. It is widely believed that multi-agent approaches based on self-organizing principles are a suitable mechanism for management of such systems [1]. In multi-agent systems, autonomous agents can learn their behaviour for given environmental conditions and coordinate their actions with immediate neighbours while the global system behaviour emerges from local learning, actions and interactions. A number of agent-based techniques have already been successfully used to support self-organization and emergence in large-scale decentralized systems. Examples include ant-colony optimization in load balancing [2], particle swarm optimization in wireless network routing [3] digital evolution in autonomous robot navigation [4], and reinforcement learning (RL) in load balancing [5] and resource allocation [6].

These techniques have mostly been applied to systems with only a single explicit objective (or multiple objectives implemented as a single policy); however, many large-scale systems will need to self-optimize towards multiple, often conflicting objectives with different temporal and spatial scope and of different priority. Such policy heterogeneity leads to agent heterogeneity, making the coordination required for self-organization particularly challenging.

To address self-optimization in such heterogeneous environments, we have developed Distributed W-Learning (DWL), an RL-based algorithm for multi-policy optimization in large-scale decentralized agent-based systems, that has been inspired by W-Learning [7]. DWL, which we briefly introduced in [8], enables agents to engage in different levels of cooperation with each other in order to optimize the performance of the system towards its goals, while respecting the priority of these goals. Collaboration mechanisms are independent of the policies that agents are implementing, therefore enabling collaboration between heterogeneous agents. All required actions and interactions are performed on an agent level, removing the need for central control or global knowledge.

We have evaluated DWL in a simulation of an agent-based self-organizing UTC system. We use UTC examples throughout the paper to illustrate issues in multi-policy optimization and to explain the details of DWL. The results of our evaluation show that in UTC, DWL can reduce the waiting time of vehicles to between 13% and 30% (depending

Presented at 3rd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO'09). ©IEEE 2009, Proceedings available at http://doi.ieeecomputersociety.org/10.1109/SASO.2009.23

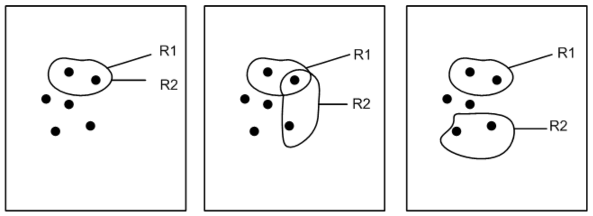

Figure 1. Policy relationships

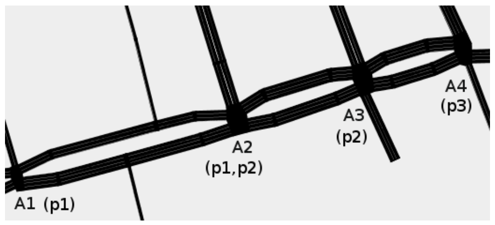

Figure 2. Agent and policy heterogeneity

on the vehicle type and traffic load) of their waiting time in the traditional round-robin or saturation-balancing adaptive traffic controllers that we used as baselines for comparison. DWL can also improve the performance of multiple policies deployed simultaneously, even over corresponding single-policy deployments, reducing the waiting time of vehicles to between 22% and 88% (depending on the vehicle type and traffic load) of their waiting time in single-policy scenarios.

The remainder of this paper is organized as follows: Section II describes agent-based self-organizing systems and the implications of the presence of multiple policies. Section III covers related work, while Section IV describes the details of DWL. Our simulation environment is presented in Section V and the results and analysis of our experiments in Section VI. Section VII concludes the paper.

II. Multi-Policy Optimization

The policies that large-scale decentralized systems are required to implement can have different characteristics. For example, policies can be deployed on different sets of agents (i.e., have different spatial scope), can be active at different times (i.e., have different temporal scope), and can have different levels of importance to the system (i.e., have different priorities). In terms of spatial scope, policies can be local (deployed on a single agent), regional (deployed on only a subset of the agents in the system) or global (deployed on all of the agents in the system). In terms of their temporal scope they can be sporadic or continuous, and their priority can range from low to high.

Additionally, policies can have different relationships to one another even if their characteristics are otherwise the same. For example, consider Fig. 1, in which R1 and R2 are both regional, continuous policies of the same priority. However, due to their regional scope, the sets of agents that implement them can be the same (i.e. all agents that implement R1 also implement R2 and vice versa), they can overlap (i.e. some, but not all, agents that implement R1 also implement R2 and vice versa), or they can be disjoint (i.e. no agent implements both R1 and R2).

Heterogeneity of policies and their characteristics leads to heterogeneity of agents while different policy relationships can lead to different levels of dependency between policies and between agents. Therefore, multi-policy optimization techniques in multi-agent environments need to not only explicitly address meeting multiple system goals, but also address the implications of the presence of multiple policies, namely agent dependency and heterogeneity, as well as policy dependency.

A. Agent Dependency

The performance of an agent in a multi-agent system can be influenced, directly or indirectly, and both positively and negatively, by other agents' actions [9].

For example, in a UTC system, the performance of one junction can be affected by some or all of its upstream and/or downstream neighbours. Consider Fig. 2 which shows several linked junctions controlled by agents. If the junction controlled by agent A2 is oversaturated, traffic can come to a standstill and queues at that junction can spill over upstream to block the junction controlled by A3. No traffic will then be able to go through this junction regardless of the actions of A3, as there will be no space available on the downstream road. Likewise, the performance of A1, A3 and any agents at other junctions that feed traffic to A2 can cause oversaturation at A2 if they are letting through more traffic than A2 is clearing. The dependency can also extend to the agents further upstream from A2, such as A4, as that agent influences the performance of A3, which in turn influences the performance of A2, causing a potential dependency between non-neighbouring agents A2 and A4. As a result, there is a potential dependency between all of the agents in a UTC system. Due to this dependency, agents that control the junctions should, when selecting an action to execute, consider not just their own direct benefit, but also the influence of that action on other agents, particularly their immediate neighbours.

B. Agent Heterogeneity

It is particularly difficult for agents to take the needs of other agents into account when agents are heterogeneous and implement different policies. The first source of heterogeneity is the difference in the agents' operating environment and capabilities. For example, in a UTC scenario, intersections can have different layouts, i.e., the number of incoming and outgoing approaches and the allowed traffic manoeuvres can be different. In RL, this maps to agents having different state spaces and different action sets, since the combinations of traffic signal settings (i.e., traffic-light phases) available to agents controlling junctions of different layout differ. This results in agents not having a common interpretation of the meaning of particular states or actions.

Another source of heterogeneity results from different policies being deployed in the system. For example, agents in a

UTC system might be required to optimize global traffic flow, but also to prioritize emergency vehicles and public transport, or deal with pedestrian crossing requests. Due to the different spatial and temporal scope of these policies, agents in the system can end up implementing different sets of policies at a given point in time. For example, consider again Fig. 2. Agent A2 is required to implement policies p1 and p2, while one of its neighbours, agent A1, is only implementing p1, and the other neighbouring, agent A3, only p2. The further downstream neighbour A4, implements only its own local policy p3.

C. Policy Dependency

Agent dependency and agent heterogeneity, as discussed in previous sections, can cause dependencies between policies as well. For example, we discussed how in Fig. 2 the actions of A3 can influence the performance of agents A2 and A4. This implies that the implementation of the policy p2 (implemented by A3) can influence the performance of policies p3 (implemented by A4) and p1 (implemented by A2). Therefore, when an agent makes an action selection, it should not only consider actions that are optimal for the implementation of its own policies, but also the effect of those actions not only on other agents, but also on other policies, particularly on those that its immediate neighbours are implementing.

Dependency between policies has been confirmed in our previous work with non-collaborative agents [10]. We observed that implementing a single policy that only addresses emergency vehicles creates a backlog of other vehicles in the system, which in turn prevent emergency vehicles from proceeding. These observations on policy dependency were addressed in our design of DWL.

D. Algorithm Requirements

In summary and based on the characteristics of the systems discussed above, we can derive a set of requirements for a multi-policy multi-agent optimization technique. There is a potential for agents to benefit from mutual cooperation, as there is a dependency between them due to their shared environment. Therefore, an algorithm should enable agents not just to learn the optimal actions for their own performance, but also which other agents are influenced by their actions and how. Algorithms should also allow multiple policies deployed on the agents to be addressed simultaneously, as there may be a dependency between policies, again due to the shared environment in which they are deployed. Finally, the algorithm should only require local learning and interactions, as no agent has a global view of the entire system.

III. Background

This section introduces the basic elements required to understand RL-based techniques and reviews closely related work.

A. Reinforcement Learning

Reinforcement learning is an unsupervised learning technique in which an agent learns how to meet its goal by receiving feedback on its actions from the environment in the form of a reward [11]. Q-learning [12], a model-free implementation of RL, is considered particularly suitable for optimization in situations where no pre-specified model of the environment is available [13], as is likely in complex large-scale decentralized systems. In Q-learning, an agent learns to associate an action a with the expected long-term reward of taking that particular action in a particular state s (i.e., in a particular set of environment conditions). This is represented by a Q-value, $Q(s,a)$ [12].

Q-learning is a single-agent, single-policy learning technique. A number of multi-policy extensions have been implemented, as agents often have to simultaneously meet multiple policies. We reviewed some of those techniques in our previous work [10] and evaluated two approaches: combining policies' state spaces into a single learning process [14] and W-Learning [7]. W-learning was shown to yield better results in the scenarios evaluated, and to be suitable as a multi-policy optimization technique for UTC [10] when deployed on non-cooperating agents. Based on these results we hypothesize that W-Learning performance could be improved by enabling collaboration between agents.

W-learning is an action-selection technique developed by Humphrys [7], where each policy is implemented as a separate Q-Learning process with its own state space. Agents learn Q-values for state-action pairs for each policy and, at every time step, each policy nominates an action based on these Q-values. Using W-Learning, agents also learn, for each of the states of each of their policies, what happens, in terms of reward received, if the action nominated by that policy is not obeyed. This is expressed as a W-value, $W(s)$. $W(s)$ is updated after each action selection according to Formula (1), where r_i is the immediate reward received, s is the current state of the policy, s' is the next state, a_i is the current action, and a'_i is the next action for policy i.

$$\begin{aligned} W_i(s) := \quad & (1-\alpha)W_i(s) + \\ & \alpha(Q_i(s,a_i) - (r_i + \gamma max Q(s',a'_i))) \end{aligned} \tag{1}$$

At each learning step, multiple policies on a single agent nominate their actions, with associated W-values for the states in which they are, and the agent selects an action to execute based on those W-values. The winning policy can be selected in several ways, for example, agents can select the policy with the highest nominated individual W-value or the one with the highest cumulative W-value over all policies.

Multi-agent learning and associated issues have been the subject of extensive research in the area of distributed artificial intelligence [15], [9]. As much of the work in this area is application area specific, in the next section we review multi-agent RL-based techniques used in our initial application area UTC.

B. Related Work

In urban traffic control implementations, RL has been shown to improve the performance of a single isolated intersection [13], as well as of several, non-collaborating intersections [16]. Various implementations of collaborative RL-based agents have also been used to improve the performance of UTC systems, such as in [17], [18], [19], [20]. These implementations focus only on improving general traffic flow without distinguishing between multiple policies or different vehicle types. In [21], bus network performance is simulated but its influence on the rest of the traffic is not modeled, while in [22] authors introduce simple rule-based ambulance priority at intersections on top of optimizing the traffic flow. None of these systems learn the appropriate behaviours for multiple policies simultaneously, e.g., no system learns how to optimize global traffic flow, while also learning how to prioritize emergency vehicles and public transport. We hypothesize that being able to address all vehicle types (i.e., all policies) simultaneously, can result in further performance improvements for all policies.

IV. Distributed W-Learning

From our previous experiments [10] we have observed that policy deployments can benefit from combining policies using W-learning independently at each agent. We have also observed a high dependency between policy deployments. Given that all agents operate in a shared environment, there is also a potential for high dependency between agents. If this is the case, agents, and the system as a whole, could benefit from cooperating to select actions that are not only suitable for their own policies, but for the other agents' policies that they affect as well. However, designing an algorithm for cooperative problem solving raises numerous issues, particularly in heterogeneous environments with no global view. We outline these challenges in the next section.

A. Issues in Multi-Policy Collaboration

In large-scale agent-based systems optimal performance of the system as a whole might not be as simple as optimizing the performance of all entities individually, as an agent might need to sacrifice its local performance in order to improve global behaviour. If this is the case, a way to motivate an agent to cooperate is needed, as, in terms of local reward received, it might be better for an agent to act selfishly. This is particularly challenging if agents implement different policies - we need to motivate an agent to sacrifice its local reward and sacrifice performance towards its own policies, to help other agents meet their policies. It is particularly important for an agent to be willing to engage in cooperative behaviour when other agents' policies have a higher priority for the system than the policies that a local agent is implementing.

Even when agents are motivated to engage in cooperation with their neighbours, we need to address the issue of how they should cooperate, i.e., what information should agents exchange and how will that information be interpreted. Heterogeneity of environments and policies makes this particularly difficult. For example, if agents only exchange their latest rewards [23], a receiving agent might not know why a sending agent received that reward, for which policy was it received, for which particular state, for which action taken, or which policy at that agent nominated that action to be taken. Heterogeneous agents are not able to exchange learnt experiences either, as their state spaces and actions differ, so the knowledge acquired at one agent is not applicable to other agents with different state-action pairs.

Once an agent is motivated to cooperate, and has a means to do so, it needs to determine with whom to cooperate. For example, it might need to cooperate only with other agents implementing the same policies as its local ones, or with all agents regardless of their policies. However, if agents only have local views they might not be aware of which other agents, if any, are implementing the same policies as they are. Additionally, agents implementing a low-priority policy might need to help agents that implement higher priority policies. Again, due to the lack of a global view, agents might not know the relative priorities of their local policies and other agents' policies. As the levels of dependency between agents might differ, agents might only need to collaborate with other agents whose actions it is influenced by, and agents that are influenced by its local actions.

B. DWL Overview

To address the above issues, our DWL algorithm enables an agent to not only learn to select actions that are suitable for its local policies, but also to learn how its actions affect its immediate neighbours, and give varying levels of weight to its neighbours' preferences when making an action selection. To motivate an agent to take into account its neighbours' action preferences (i.e., to collaborate), each agent, as well as its own policies, implements a "remote" policy "help neighbour N_i to implement its policy P_{ik}" for each of the policies deployed on each of its immediate neighbours. This policy receives a reward each time policy P_{ik} receives a reward on neighbour N_i. By using remote policies, DWL also enables cooperation between heterogeneous agents, i.e., agents that implement different policies, and have different state space and action sets. DWL does not require any central component or global knowledge, and it relies only on local learning, local actions, local rewards from the environment, and interactions with immediate neighbours.

C. Definition of DWL

A DWL-based system consists of the following:

- A set of agents A = {A_1, ..., A_n}, where each agent controls a set of actuators. In our UTC simulation, an agent controls the traffic lights at a single junction, and only signal-controlled junctions have agents associated with them.

- Each agent A_i has a set of neighbours N_i = {N_{i1}, ..., N_{im}} consisting of all agents $\mathrm{A}_j \in$ A that are one-hop neighbours of the agent A_i. In our UTC simulation, if the first downstream/upstream junction on a link

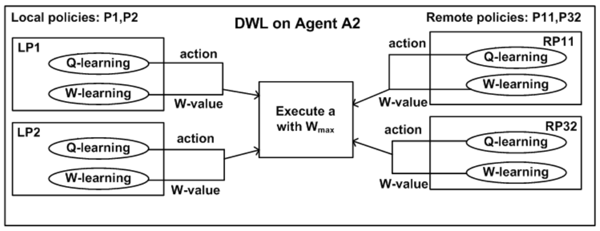

Figure 3. DWL action nomination

is not a signal-controlled junction, the first following downstream/upstream agent will be considered as a neighbour of A_i.

- A set of policies LP_i = $\{LP_{i1}, ..., LP_{ip}\}$ is deployed at A_i. We refer to these policies as *local policies* of A_i. Local policies can be active or inactive at each time step.

- Each agent A_i has a set of policies RP_i = $\{RP_{ij1}, ..., RP_{ijr}\}$ whose goal is to contribute to the implementation of each local policy LP_{jk} deployed at each $A_j \in N_i$. We refer to these policies as *remote policies* of A_i.

- Each policy on each agent is implemented as a combination of a Q-learning and a W-learning process. It has Q-values associated with each of its state-action pairs and W-values associated with each of its states. An agent's current action is denoted as a_i and previous action a_{i-1}. A policy's current state is denoted as s_i and previous state s_{i-1}.

The learning process is initialized by an agent receiving from each of its immediate neighbours state-space representations for each of the policies that they implement, as described in Algorithm 1. An agent initiates learning processes for its local policies with local states and actions, and learning processes for its remote policies with remote states and local actions.

Algorithm 1 DWL Initialization

```
For each agent $A_i$ in A
  //Init local policies
  For each $LP_{il}$ in $LP_i$
    InitQLearning($LP_{il}$ states, $A_i$ actions)
    InitWLearning($LP_{il}$ states)
  End for
  //Create remote policies
  For each $A_j$ in $N_i$
    For each $LP_{jk}$ in $LP_j$
      Add corresponding $RP_{ijk}$ to $RP_i$
    End for
  End for
  //Init remote policies
  For each $RP_{ijk}$ in $RP_i$
    InitQLearning ($RP_{ijk}$ states, $A_i$ actions)
    InitWLearning ($RP_{ijk}$ states)
  End for
End for
```

Algorithm 2 DWL at each learning step

```
For each simulation step
 For each $A_i$ in A
   //Get nominations by local policies
   For each $LP_{il}$ in $LP_i$
     Determine $LP_{il}$'s state $s_{il}$
     Get reward from $A_i$'s environment
     Update Q($s_{il-1}$, $a_{il-1}$) for $LP_{il}$,
     Update W($s_{il-1}$)
     Nominate action $a_{il}$ with max Q for $s_{il}$
     Get W($s_{il}$)
   End for
   //Get nominations by remote policies
   For each $RP_{ijk}$ in $RP_i$
     Get $RP_{ijk}$'s state $s_{ijk}$ from $A_j$
     Get reward for $s_{ijk}$ from $A_j$
     Update Q($s_{ijk-1}$, $a_{ijk-1}$)
     Update W($s_{ijk-1}$)
     Nominate action $a_{ijk}$ with max Q for $s_{ijk}$
     Get W($s_{ijk}$)
   End for
   //Select and execute action
   Pick winning action $a_i$ (see Formula 2)
 End for
End for
```

During the learning process an agent learns Q-values for remote-state/local-action pairs and W-values for remote states, i.e., it learns the effect of its local actions on its neighbours states. In order for an agent to learn this, it needs, at each time step, to receive information about its neighbours' current states and the rewards that they have received. At each time step, both local and remote policies nominate an action with an associated W-value. W-values of inactive policies are set to 0. Algorithm 2 outlines the details of the process and the communication between agents. Fig. 3 presents an example of one step of the DWL nomination process for an agent A2 from Fig. 2. Local policy action nominations are taken into account with their full W-values, while remote policy nominations are multiplied by a cooperation coefficient C, where $0 \leq C \leq 1$, to enable an agent to give varying weight to its neighbours' action preferences. $C = 0$ means that the local agent is non-cooperative, i.e., it does not consider its neighbours' performance when selecting an action, while $C = 1$ means that a local agent is fully cooperative, i.e., it cares about its neighbours performance as much as it cares about its own. The action that is executed at that time step, i.e., the one that wins the competition between policies at that time-step is the one with the highest W-value (W_{win}), after remote W-values have been scaled by the cooperation coefficient:

$$W_{win} = \max(W_{il},\ C\ \times W_{ijk}) \qquad (2)$$

where W_{il} are W-values nominated by local policies of A_i and W_{ijk} are W-values nominated by remote policies of A_i.

D. DWL Summary

Designed in the way discussed above, DWL meets the requirements we specified in Section II-D: the action selection of each agent is based on learning the optimal action for itself, as well as for its immediate neighbours, communication is enabled between heterogeneous agents (i.e., it does not

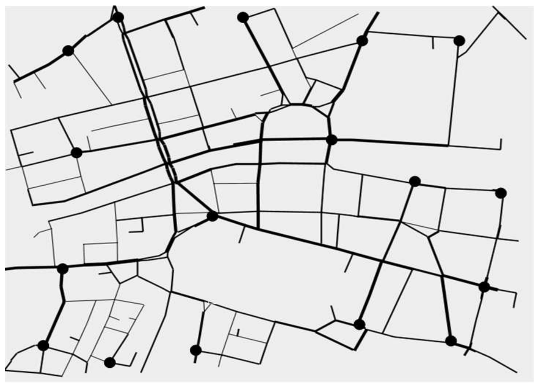
Figure 4. Simulation map

depend on the agent's environment and capabilities or on the policies deployed), and all learning and communication is local. It also addresses all the issues discussed in the introduction of this section: agents are motivated to cooperate by introducing remote policies for which they receive rewards, agents learn the agents with which they have the highest dependencies by learning the Q-values and W-values for the pairs of remote agent's states and their local actions, the relative priority of policies is determined by different rewards received for different policies (which is reflecting in the Q-values and W-values), and agents are able to exchange all the information required to be fully informed about the states of their neighbouring agents' policies, and the rewards associated with those states.

V. Experiment Setup

To evaluate the performance of our proposed algorithm in a UTC scenario, we use an urban traffic simulator developed in Trinity College Dublin [24]. The map we used is shown in Fig. 4 and corresponds to Dublin's inner city. The map covers 270 junctions, 62 of which are signal-controlled and can be controlled by our agents. We believe that using a road network based on a real city provides a more realistic simulation; the majority of the UTC simulations presented in Section III-B use a grid of junctions with roads connecting them having a similar layout, while our map includes junctions with different numbers of approaches, roads with different numbers of lanes, as well as one-way and two-way streets.

Cars enter the simulation at one of the 17 points marked on the map, travel along one of 260 different routes, and exit the simulation through one of the marked points. Public transport vehicles make up 5% of the overall traffic, travel along 20 bus routes, and stop at 20 bus stops along those routes, with at least one stop on each route. 46 of the total of 62 signal-controlled junctions are positioned on bus routes. We performed experiments described here with two traffic loads:

- low load - approximately 35,000 vehicles are inserted into the simulation over a 12-hour period.
- high load - approximately 60,000 vehicles are inserted into the simulation over a 12-hour period.

Each experiment is performed five times and the average results are presented. Traffic light phases are generated for each set of traffic lights based on the road layout and traffic rules. The default duration of each phase is 20 seconds.

A. Baselines

As baselines against which to compare the performance of our DWL-based agents we implemented two traffic control techniques, Round Robin (RR), and a simple adaptive technique referred to as SAT.

Round Robin: A Round Robin (RR) junction controller at a junction continuously loops through all available phases at that junction, with each phase lasting 20 seconds. Note that the layout of junctions is different, so the number and type of phases through which an RR controller cycles will vary from junction to junction.

SAT: SAT is a SCATS-like [25] saturation-balancing algorithm for controlling phase duration. Our implementation is based on the SAT implementation in [26]. SAT cycles through all available phases, adjusting the duration of each phase at the start of each cycle, based on the degree of saturation during the previous cycle. 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. A SAT implementation depends on three parameters: the minimum duration of each phase, the phase increment (the length by which a phase duration can increase or decrease in a single step), and a maximum cycle length factor (the maximum duration of a cycle is determined by multiplying the number of phases by the minimum phase duration and a cycle length factor). We simulated SAT performance with different combinations of these parameters and selected the best performing combination for the experiments presented in this paper. The experiments presented here have a minimum phase length of 20 seconds, a phase increment of 10 seconds, and a maximum cycle length factor of 1.2.

B. Agent Implementations

As policy heterogeneity is one of the central issues that DWL addresses, for the evaluation we implemented two policies with different characteristics - one is a standard-priority, global, continuous policy that addresses global waiting times for all vehicles in the system (GWO) and the other is a high-priority, regional, sporadic policy that optimizes public transport vehicles (PTO).

GWO - Optimizing global waiting time: This policy does not distinguish between vehicle types and aims to minimize the waiting time of all the vehicles in the system. To do this, each traffic light aims to minimize the time vehicles have to wait at its approaches. We assume each agent is able to get the information (from sensors such as traffic cameras, inductive loops and in-car GPS devices) on how many vehicles are waiting at each of its approaches. This is then mapped to a state space that represents the order of congestion of the

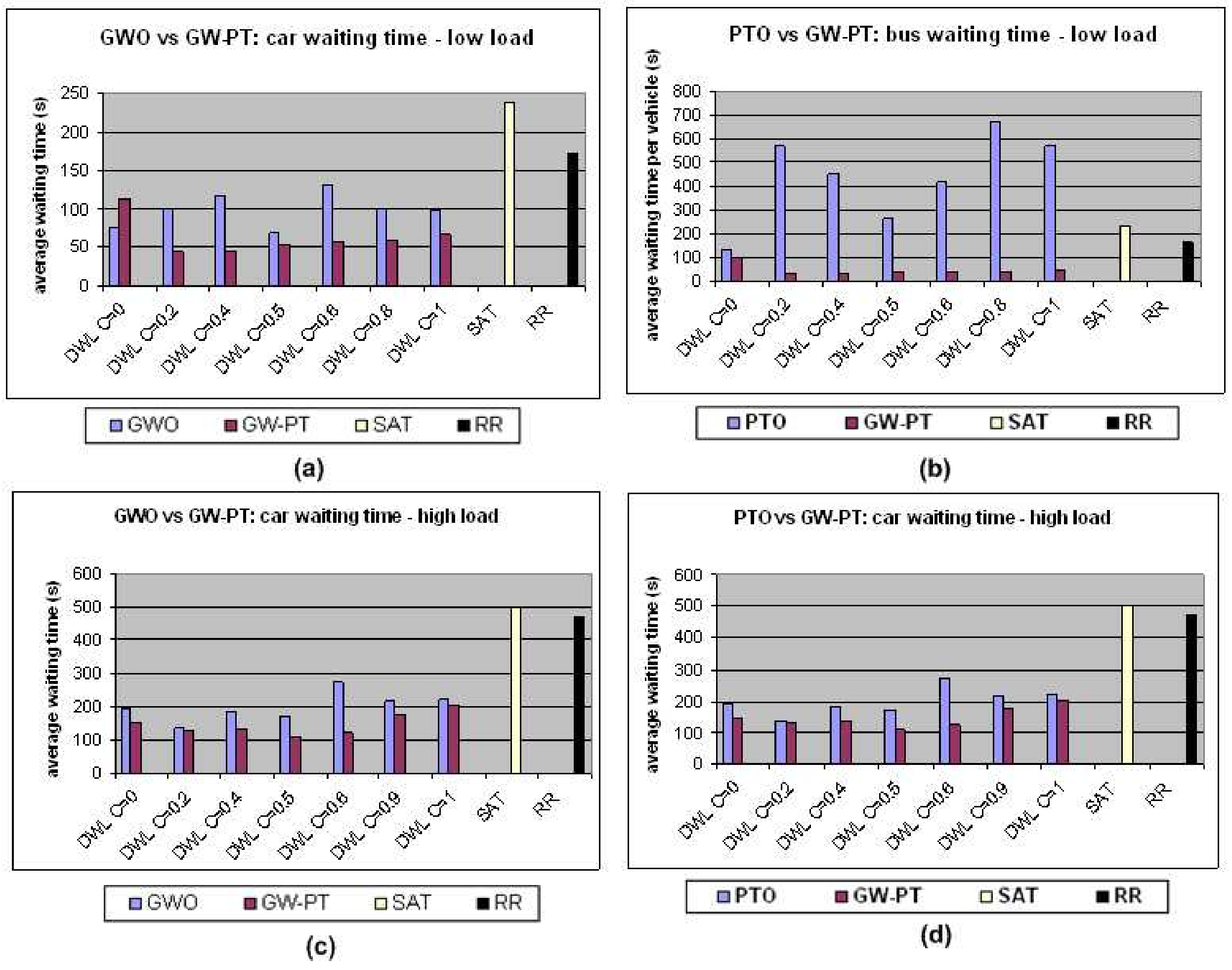

Figure 5. Average waiting times per vehicle type

junctions approaches. Additionally, the state space encodes information on whether the current congestion level is better or worse than the congestion level at the previous step. Note that we do not set any explicit thresholds to denote levels of congestion, we only measure whether it is better or worse than in the previous state. This design decision was made to facilitate rewarding an agent for clearing more traffic than arrives in the meantime, i.e., to learn to release the approaches with the highest arrival rates. Agents obtain a 100-point reward in each step that they are in one of the states in which congestion is better than before. GWO is implemented on all 62 agents for the whole duration of the experiment.

PTO - Prioritizing public transport vehicles: This policy addresses only the waiting time of the public transport vehicles (buses), aiming to give them priority over other vehicle types. We assume an agent is capable of sensing the presence of buses on its approaches, and maps that information to the state representation. An agent receives a reward for being in a state with no buses present, motivating it to release the approach with the bus waiting as soon as possible. The reward received for being in this state is 120; it is higher than the reward GWO agents receive, to motivate agents to give higher priority to clearing buses. PTO is implemented on 46 agents situated on bus routes and is triggered only when public transport vehicles are detected on the approach.

Experiment parameters: We ran three sets of experiments: GWO by itself, PTO by itself, and GWO and PTO simultaneously (GW-PT). In each set of experiments DWL is used for action selection. Each experiment has three phases. In the first phase agents learn Q-values, in the second phase they continue learning Q-values but also learn W-values, and in the third phase agents exploit the learnt Q-values and W-values learnt. Each phase of the experiment runs for 750 minutes, giving agents 1500 minutes to converge on optimal Q-values, 750 minutes to converge on optimal W-values, and 750 minutes to exploit the values learnt. The results presented are from the exploitation phase. Experiments were repeated for different values of cooperation coefficient C, to evaluate how different levels of cooperation influence performance, ranging from non-cooperative ($C = 0$), to fully cooperative ($C = 1$).

C. *Metrics*

We compared the performances of our UTC agents using two metrics: average vehicle waiting time and traffic density.

Average waiting time is measured per vehicle type, as our two policies are addressing the performance of different sets of vehicles. We compare average waiting time for cars and average waiting time for public transport vehicles, to observe the influence policies have on one another, and the influence of cooperation on the performance of particular policies.

We also measured traffic density, the ratio of occupied road space to available road space [27]. As the numbers of vehicles inserted into the simulation are the same for all policies and levels of cooperation, measuring density reflects levels of congestion in the system, i.e., how quickly and effectively a system is able to clear out the incoming traffic. The lower the density, the better the performance of the system.

VI. Results and Analysis

We have evaluated the performance of our DWL deployments against the baselines and compared the performance of single-policy and multi-policy scenarios. We have also compared the performance of independent versus collaborating agents and investigated the impact of different levels of collaboration in DWL on system performance. Fig. 5 summarizes the results of our experiments. Parts (a) and (b) show average waiting times for cars and buses during low traffic load, for both single-policy (GWO and PTO) and multi-policy (GW-PT) scenarios, for different levels of cooperation ($0 \leq C \leq 1$), as well as the performance of the SAT and RR baselines. Parts (c) and (d) show the corresponding set of results measured during high traffic load.

A. *DWL vs. Baselines*

Single-policy GWO outperforms both baselines regardless of the level of cooperation (see Fig. 5 (a) and (c)). It lowers the average waiting time for cars to 30% (at low load) and 28% (at high load) of their average waiting time in SAT and to 40 % (at low load) and 30% (at high load) of their average waiting time in RR.

However, single policy PTO performs badly in terms of average waiting time for the vehicles that it addresses (i.e., buses) for both loads (see Fig. 5 (b) and (d)). We believe that the reason for this is that public transport vehicles represent only 5% of total traffic, so attempting to optimize only their performance negatively affects the performance of the other 95% of vehicles. Other vehicles then create a backlog in the system, in turn affecting the performance of public transport vehicles themselves, as, given the shared infrastructure, there is no available road space for them to proceed. This is illustrated in Fig. 6 that compares average traffic density of PTO and GW-PT. While GW-PT maintains steady density, i.e. it clears all the incoming traffic, PTO is not able to deal with arriving traffic and the density keeps rising. While this may be seen as an obvious consequence of the shared infrastructure, it nevertheless shows the importance of optimizing for multiple policies simultaneously.

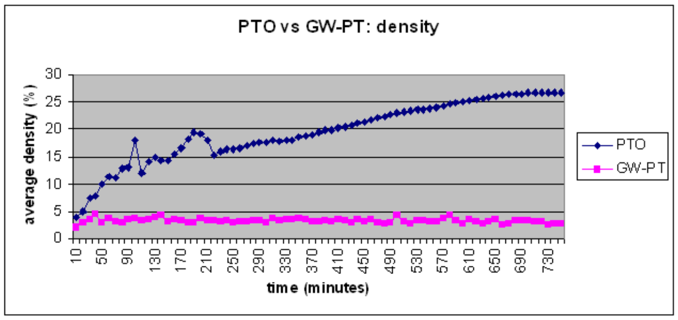

Figure 6. PTO density

Multi-policy GW-PT outperforms both baselines with respect to both car and bus waiting times and regardless of the level of cooperation. Average waiting time for cars is lowered to 17 % (at low load) and to 22% (at high load) of their average waiting time in SAT and to 24% (at low load) and to 23 % (at high load) of their average waiting time in RR. Average waiting time for buses is lowered to 13% (at low load) and to 24% (at high load) of their average waiting time in SAT and to 18% (at low load) and 30% (at high load) of their average waiting time in RR.

From the above results, we observe that DWL is at least a viable algorithm for multi-policy self-optimization in UTC, as, in these experiments, all deployments of GW-PT (and GWO) outperform techniques commonly deployed in UTC systems. These results also highlight policy dependency, as shown by the inferior performance of single-policy PTO.

B. *Single-Policy vs. Multi-Policy Scenarios*

The best performing deployment of DWL for both cars and buses is multi-policy GW-PT. It lowers the average waiting time of cars to 88% of their average waiting time in the best performing single-policy scenario at low load and to 78% at high load. For buses, it lowers the average waiting time to 22% of their average waiting time in the best performing single-policy scenario at low load and to 40% at high load. From this we observe that DWL's capability of addressing multiple policies simultaneously can improve the performance for both policies over their single-policy deployments. The improvement comes from the fact that when the next action matters more to one policy than to another, DWL enables the selection of the action nominated by the policy with higher action weight. In that way, policies can take turns in selecting actions as and when it matters to them most, enabling good performance of all policies. Therefore, when policies have mutual interest in each other's optimal performance DWL enables that optimal performance.

There is a one case of GWO outperforming GW-PT, at low load for non-cooperative $C = 0$ scenario (see Fig. 5 (a)). This is not surprising, as once we start giving priority to public transport in a multi-policy scenario, car waiting time increases. However, when cooperation is introduced, and when loads in the system are higher, cars benefit from good performance of public transport as well, due to the shared infrastructure.

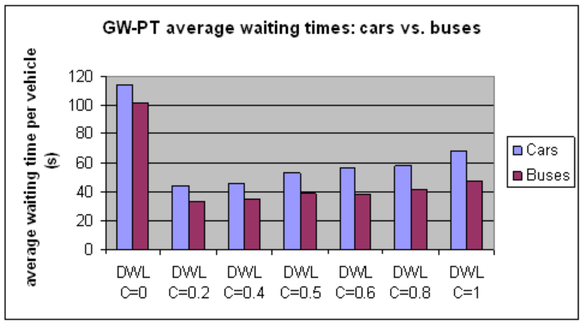

Figure 7. Cars vs. buses: average waiting time

We also observe that, for all GW-PT scenarios at both loads, average waiting time for buses is lower than that of cars (see Fig. 7) from which we conclude that DWL is able to respect the relative priorities of the policies.

C. Independent vs. Cooperating Agents

We performed all the experiments with a range of cooperation coefficients. The $C = 0$ case is the equivalent of standard W-learning (see Section III-A) being deployed on each agent individually.

For single-policy GWO, cooperation can lower car waiting time to 93% of the waiting time in a non-collaborative deployment at low load and to 71% at high load (see Fig. 5(a) and (c)). For PTO, both independent and collaborative scenarios performed badly, due to the reasons already discussed when comparing PTO to baselines.

The benefit of cooperation is particularly observed in the multi-policy scenario, GW-PT. The waiting time of cars is lowered to 38% of their waiting time in the non-collaborative deployment at low load and to 72% at high load. For public transport vehicles, the waiting time is lowered to 32% of their waiting time in the non-collaborative deployment at low load and to 74% at high load. These results emphasize DWL's capability to allow heterogeneous agents to engage in performance improving cooperation. The improvement comes for the fact that DWL enables agents to take into account the influence of their actions on their neighbours when making action selections. Agents execute actions that are good not just for them locally, but also good for their neighbours, leading to better overall performance of the system. Additional benefit is observed when cooperation is combined with multi-policy optimization. DWL enables not just local policies, but any of the neighbour's policies as well, to influence the selection of the next local action, even if a local agent is not implementing that policy itself. In effect, a higher number of agents is contributing to the implementation of each policy (i.e., all agents that directly contribute to it plus all of their neighbours that do not implement the policy directly) leading to better policy performance.

D. Cooperation Coefficient Impact

From the experiments we observe that the optimal level of collaboration differs with traffic load.

At low load, small C is the most suitable and the best results are achieved with $C = 0.2$. This indicates that an agent's neighbour's W-value has to be at least five times larger than any of the W-values nominated by local policies in order to influence an agent's actions. We believe that the most common situation when this is possible is when local W-values are close to 0. $W \cong 0$ indicates that the next action is irrelevant to local policies, so the agents can defer to their neighbours on the action to be executed.

At high load, it appears that agents are not often in the situation when they do not have a strong preference about what their next action is; there is a drop in waiting time when C is changed from 0 to 0.2, but not as large as during low load. The biggest improvement in performance is achieved when $C = 0.5$. This indicates that the biggest benefit of cooperation during high traffic loads is to be able to execute actions nominated by neighbours only when it really matters to them what action is executed.

We have also observed, that for both levels, full cooperation ($C = 1$) is the worst performing collaboration level apart from the fully non-cooperative ($C = 0$) scenario. This emphasizes the importance of DWL's capability to enable agents to engage in different levels of cooperation controlled by the cooperation coefficient, as neither fully cooperative nor fully non-cooperative scenarios produce the best results, and finer control over levels of cooperation is needed.

E. Summary

Our experiments indicate that DWL is a promising technique for multi-policy multi-agent optimization in self-organizing systems. In the case of UTC, DWL outperforms both SAT and RR. By allowing simultaneous deployment of multiple policies, it enables agents to exploit the benefits of multi-policy optimization over single-policy optimization. It also enables performance-improving cooperation between heterogeneous agents, and allows flexibility in terms of the level of cooperation in which agents engage. This is an important characteristic as we have observed that different levels of cooperation are suitable for different traffic loads. Our experiments also highlight the dependency between agents, as cooperative scenarios outperform non-cooperative scenarios, as well as the dependency between policies, as multi-policy scenarios outperform single-policy scenarios.

VII. Conclusions and Future Work

This paper addressed the problem of multi-policy optimization in agent-based self-organizing systems. We have discussed the need for cooperation and the challenges posed by the heterogeneity of agents and policies in these systems. We have presented Distributed W-Learning (DWL), an algorithm that enables optimization towards multiple policies on multiple agents simultaneously, enabling agents to engage in different levels of cooperation. Our experiments show that, in a UTC scenario, DWL consistently outperforms our baselines SAT and RR. Performance is improved by DWL's capability to deploy multiple policies simultaneously (as multi-policy scenarios outperform single-policy scenarios), as well as by its capability to enable cooperation between agents implementing different policies (as cooperative scenarios can outperform non-cooperative scenarios). We have also observed that different levels of cooperation are suitable for different system loads. Currently, all agents in the system engage in the same level of cooperation. In future work, we will investigate the possibility of agents learning how much cooperation to engage in, based on their relative importance in the system. We also plan to evaluate DWL in other application areas, to confirm its wider suitability for agent-based self-organizing systems. Additionally, we will evaluate DWL for other combinations of policies with different characteristics and different relationships, to observe any potential links between various degrees of heterogeneity and DWL's performance. Finally, a known drawback of RL is its inconsistent performance during exploration, which we have also observed in our simulation and will need to address before DWL can be deployed in practice.

Acknowledgment

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

References

[1] G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos, "Self-organization in multi-agent systems," *Knowledge Engineering Review*, vol. 20, no. 2, pp. 165–189, 2005.

[2] A. Montresor, H. Meling, and O. Baboglu, "Messor: Load-balancing through a swarm of autonomous agents," in *AP2PC '02*, ser. Lecture Notes in Artificial Intelligence, G. Moro and M. Koubarakis, Eds., no. 2530. Bologna, Italy: Springer-Verlag, pp. 125–137.

[3] B. A. Kadrovach and G. B. Lamont, "A particle swarm model for swarm-based networked sensor systems." in *SAC*, 2002, pp. 918–924.

[4] H. J. Goldsby, B. H. C. Cheng, P. K. McKinley, D. B. Knoester, and C. A. Ofria, "Digital evolution of behavioral models for autonomic systems," in *ICAC '08*. IEEE Computer Society, pp. 87–96.

[5] J. Dowling, "The decentralised coordination of self-adaptive components for autonomic distributed systems," Ph.D. dissertation, Trinity College Dublin, 2005.

[6] G. Tesauro, "Reinforcement learning in autonomic computing: A manifesto and case studies," *IEEE Internet Computing*, vol. 11, no. 1, pp. 22–30, 2007.

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

[8] I. Dusparic and V. Cahill, "Distributed W-learning: An algorithm for multi-policy optimization in decentralized autonomic systems (poster)," in *Proceedings of the 6th International Conference on Autonomic Computing and Communications*, 2009.

[9] K. Sycara, "Multiagent systems," *AI Magazine*, vol. 19, no. 2, 1998.

[10] I. Dusparic and V. Cahill, "Using reinforcement learning for multi-policy optimization in decentralized autonomic systems - an experimental evaluation," in *The Proceedings of the 6th International Conference on Autonomic and Trusted Computing*, 2009.

[11] R. S. Suton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: A Bradford Book. The MIT Press, 1998.

[12] C. J. C. H. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992.

[13] B. Abdulhai, R. Pringle, and G. Karakoulas, "Reinforcement learning for the true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, May/June 2003.

[14] H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira, "Learning multi-goal dialogue strategies using reinforcement learning with reduced state-action spaces," in *International Journal of Game Theory*, 2006, pp. 547–565.

[15] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *AAMAS '05*, vol. 11, no. 3, pp. 387–434.

[16] B. C. da Silva, D. de Oliveira, A. L. Bazzan, and E. W. Basso, "Adaptive traffic control with reinforcement learning," in *AAMAS '06*, May.

[17] E. Bitting and A. A. Ghorbani, "Cooperative multiagent systems for the optimization of urban traffic," in *IAT '04*. Washington, DC, USA: IEEE Computer Society, pp. 176–182.

[18] M. D. Pendrith, "Distributed reinforcement learning for a traffic engineering application," in *AGENTS '00*. New York, NY, USA: ACM Press, pp. 404–411.

[19] A. Salkham, R. Cunningham, A. Garg, and V. Cahill, "A collaborative reinforcement learning approach to urban traffic control optimization," in *IAT '08*.

[20] M. Wiering, "Multi-agent reinforcement learning for traffic light control," in *ICML '00*. Morgan Kaufmann, San Francisco, CA, pp. 1151–1158.

[21] D. Meignan, O. Simonin, and A. Koukam, "Simulation and evaluation of urban bus-networks using a multiagent approach," *Simulation Modelling Practice and Theory*, vol. 15, no. 6, pp. 659–671, July 2007.

[22] E. Oliveira and N. Duarte, "Making way for emergency vehicles," in *Proceedings of the 2005 European Simulation and Modelling Conference*, pp. 128–135.

[23] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *ICML '93*.

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

[25] P. Lowrie, "SCATS: The sydney co-ordinated adaptive traffic system - principles, methodology, algorithms," in *Proceedings of the IEE International Conference on Road Traffic Signalling*, 1982.

[26] S. Richter, "Learning traffic control - towards practical traffic control using policy gradients," Albert-Ludwigs-Universitat Freiburg, Tech. Rep., 2006.

[27] A. L. Bazzan, "A distributed approach for coordination of traffic signal agents," *Autonomous Agents and Multi-Agent Systems*, vol. 10, no. 1, pp. 131–164, 2005.