Research Issues in Multiple Policy Optimization Using Collaborative Reinforcement Learning

Ivana Dusparic and Vinny Cahill
Distributed Systems Group
Department of Computer Science
Trinity College Dublin

E-mail: {ivana.dusparic, vinny.cahill}@cs.tcd.ie

Abstract

Self-organizing techniques have successfully been used to optimize software systems, such as optimization of route stability in ad hoc network routing and optimization of the use of storage space or processing power using load balancing. Existing self-organizing techniques typically focus on a single, usually implicitly specified, system goal and tune systems parameters towards optimally meeting that goal.

In this paper, we consider optimization of large-scale multi-agent ubiquitous computing environments, such as urban traffic control. Applications in this class are typically required to optimize towards multiple goals simultaneously. Additionally, these multiple goals can potentially be conflicting, change over time, and apply to various parts of the system such as a single agent, a group of agents, or the system as a whole. In contrast to existing self-organizing systems in which agents are homogeneous to the extent that they are working towards a common goal, agents in these systems are heterogeneous in that they may have differing goals. Thus, existing self-organizing optimization techniques must be extended to deal with multiple goal optimization and the resulting heterogeneity of agents. In this paper we present a research agenda for extending Collaborative Reinforcement Learning (CRL), an existing self-organizing optimization technique, to support multiple policy optimization.

1. Introduction

Self organizing algorithms have been used as a technique for engineering ubiquitous systems that are required to optimize system performance without global knowledge or central control. Examples of such implementations are routing using Ant Colony optimization algorithms [3, 5], load balancing using Collaborative Reinforcement Learning [6], and player performance in the robot soccer championship RoboCup using evolutionary algorithms [11].

The above systems optimize with respect to only one system policy expressing the single goal of the system. However, the use of self-organizing techniques for the engineering of large real-world decentralized systems requires these techniques to be able to optimize with respect to multiple policies representing multiple goals of these systems. For example, Urban Traffic Control (UTC) applications must deal with optimizing general traffic throughput, but also minimize public transport and emergency vehicle waiting times.

Multiple policies can apply to a system simultaneously, or the system can switch between policies over time, influenced by the system's state or environment. These multiple policies can be of differing importance to the system, i.e., they can be of the same importance for system operation and therefore have the same priority, or some policies can have higher priorities than others. Policies can apply to different levels of the system; they can be global, i.e., apply to a system as a whole, apply only to parts of the system, i.e., to a specific group of agents, or only to a single agent. in contrast to single-policy systems where agents are homogeneous and work towards a single common goal, this leads to a situation where agents are heterogeneous. Heterogeneity of agents brings up a number of issues. Agents can compete with each other and be selfish in implementing only their own policies. Alternatively, agents implementing different policies could collaborate to meet global system goals even if that means temporarily sacrificing performance w.r.t their local policies if they are of lesser importance to the system as a whole. Game theory [8] deals with competing agents but is out of scope of this work as we will initially concentrate on heterogenous collaborating agents. In the situation where collaborating agents implement different policies, protocols for exchange of feedback between agents

International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'07)
0-7695-2973-9/07 $20.00 © 2007 IEEE

and the type of information passed should be modified to include information on the sending and receiving agents' current policies. Agents need to be able to decide which agent's feedback to take into account and to what degree. Techniques used to engineer large-scale multi-agent decentralized systems should be able to deal with these characteristics and more generally with heterogeneity of agents.

Our approach to addressing these problems is an extension of the self-organizing technique Collaborative Reinforcement Learning. Section 2 of this paper presents background on CRL, background on multiple policy optimization for a single agent using basic reinforcement learning algorithms, and background on applications of RL in Urban Traffic Control that will be our initial application area. Section 3 defines elements of multiple-policy CRL and the relationships between them and discusses the research challenges and implementation decisions we face in extending CRL. Section 4 introduces our approach to answering relevant research questions using Urban Traffic Control, and Section 5 concludes the paper.

2. Background

The self-organizing multiple policy optimization technique that we propose has its roots in Reinforcement Learning (RL) [17]. In order to motivate the design decisions and challenges, we provide a summary of basic RL, a cooperative multi-agent extension of RL called CRL [6], existing work on multiple-goal learning using RL, and existing work in applying RL to UTC.

2.1. Collaborative Reinforcement Learning

Reinforcement Learning algorithm is a single-agent, single (implicit) policy, unsupervised learning technique [17]. Decision making is modelled as Markov Decision Process [17]. An agent can be in one of the predefined states in the state set. State set depends on the agent's circumstances relevant to performing w.r.t. the specified goal. Agents are capable of performing actions. Performing an action can leave an agent in the current state, or cause transition to another state in the state space. This transition function is probabilistic. Each action an agent performs is rewarded based on whether the new state is preferred over the state in which the agent was prior to performing the action (see Figure 1). Over time, agents learn the best action to perform for each particular state, i.e., the action that will maximize their long term reward, the so-called value function [17]. Optimization w.r.t to a particular goal is achieved by agents learning to perform actions that maximize their long term accumulated reward. Most popular algorithms for learning and selecting the best action are Q-Learning and Sarsa [17].

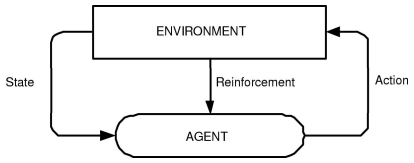

Figure 1. Reinforcement Learning

Collaborative Reinforcement Learning [6] extends RL to multi-agent learning. CRL introduces exchange of action rewards between the agents so that agents learn from each other's actions as well (see Figure 2). By exchanging feedback on actions performed, agents converge towards the solution optimal for the system performance, not to their individually best solutions. This overcomes the lack of central control and global knowledge. CRL has so far been used to implement only single policy optimization in load balancing, ad-hoc network routing, and urban traffic control [7, 6].

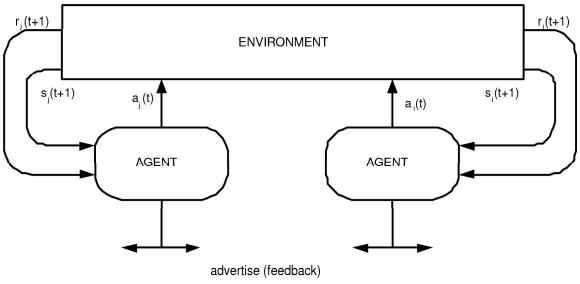

Figure 2. Collaborative Reinforcement Learning

2.2. Multiple Policy optimization

Use of basic RL for multiple goal optimization has been a subject of a lot of research as it is recognized that most goals do not exist in isolation. Most of the techniques involve learning for each goal in isolation, comparing the selected outcomes for each learning process, and selecting one action based on various criteria [10]. Learning is performed separately as combining state spaces for each learning process, i.e., policy, leads to very complex state spaces. Another technique suggested is using reduced combined state spaces [4], where all goals have a joint state space, but the state space is reduced by using prior knowledge to eliminate invalid state-action pairs. Also, there is a debate as to which particular RL algorithm should be used when implementing multiple policy optimization [16, 9], or should new variations of RL algorithms be introduced [10].

0-7695-2870-9/07 $20.00 © 2007 IEEE

2.3. Reinforcement Learning in UTC

Various reinforcement learning techniques, alone or in combination with other algorithms, have been applied in UTC, but there is currently no fully decentralized implementation that deals with multiple policies. Cunningham et al. [7], Wiering et al. [18], Abdulhai et al. [1], Pendrith [14], and Camponogara and Kraus [2] apply Q-learning to optimization of a single-policy relating to traffic congestion. Cunningham et al base the control decisions on traffic arrival rates, Abdulhai uses queue length, Weiring uses cars' own estimates of travel time, while Pendrith's decisions are made centrally based on a single shared policy. Richter et al. [15] use natural actor-critic algorithms, which is a combination of four different reinforcement learning techniques, to optimize traffic volumes. Unlike in other UTC applications of RL, they model the environment as Partially Observable Markov Decision Process (POMDP) [17], where the state of the system is not observable but the probability of the system being in a certain state is calculated and action selection is based on that probability. Mikami et al. [13] as well as Yang et al. [19] combine reinforcement learning implemented at local agents (traffic lights) with global centralized genetic algorithms that tune the parameters for local learning. Li et al. [12] introduce four traffic-light control strategies, including the emergency vehicle strategy, but they suggest that the scheduling and planning is performed by a central manager, and they do not provide implementation details.

3. Extension of CRL for Multiple Policies

Multiple policy CRL (MPCRL) will have to combine the characteristics of CRL and multiple policy RL in order to meet the requirements of large-scale decentralized systems with multiple goals. Goals will have to be explicitly stated in system's policies, which opens a number of issues about how to map policy to CRL elements such as state space and reward. Additionally, there are a number of issues arising from introducing multiple policy learning to CRL that will be outlined in this section. However, it is first crucial to identify what exact constructs are involved in this extended learning technique, what are their characteristics, and how do they relate to one another. The first part of this section provides some definitions of MPCRL constructs and the second part discusses open research questions we will tackle in extending CRL.

3.1. Elements of Multiple Policy CRL

Large decentralized multi-agent systems whose performance is to be optimized w.r.t. to multiple policies using MPCRL consists of the following elements:

- Set of agents $\{a_1...a_n\}$ where an agent is characterised by the following:
  - Agents sense their environment including the entities in the environment.
  - Agents perform actions that affect the environment.
  - Agents receive variable rewards for actions they perform.
  - Agents learn the best actions to perform for given environmental conditions and entity characteristics, based on the rewards received.
  - Agents have means of exchanging feedback and actions learnt with other agents.
- Sets of entities$\{e_{x1}...e_{xn}\}$ where:
  - An entity is part of the environment on whose characteristics agents' actions depend.
  - x is a type of entity distinguished from other types of entity by its static characteristics. Policies can assign different priority levels to entity type x.
  - Entities have dynamic characteristics that agents are able to sense. Policies can assign different priority levels to entity type x based on its dynamic characteristics.
- Set of metrics $\{m_1...m_n\}$ where each m is a function of the dynamic characteristics that an agent a is capable of sensing about entity e or group of entities E_x and optionally a function of time t.
- Reward r associated with a single action performed by a single agent in a given state, received immediately after the action is performed.
  - The reward is a function of the metrics changed and observed as a result of a performed action.
  - Rewards can be positive, in the case where the new state is preferred over the previous state, or zero or negative, if the previous state was the same or preferred over the new state.
  - The goal of each agent is to maximize the long term reward (value function [17]) received rather than immediate the single action reward.
- Set of policies $\{p_1...p_n\}$ where:
  - p is a policy that specifies a single goal of a group of agents (where the members and size of the group may differ for different policies - global, regional, local policies).

0-7695-2870-9/07 $20.00 © 2007 IEEE

- Policy can contain the following information:
  - policy identifier and description
  - list of entity types with which it is concerned where each entity type is associated with one or more metrics required to be observed for the implementation of the policy as well as the optimal value(s) of the metric that the policy is aiming to achieve (minimize, maximize, reach threshold value, set to true or false)
  - list of zero or more policy triggers (agent's states, context conditions, threshold values or patterns).
- Policies must have a priority level, either relative to other policies, or absolute priority for the set of agents.
- Policies can be compatible or in conflict with each other. Two policies are compatible if, as the result of the learning process, an agent's preferred actions for one policy in all relevant states are the same as the agent's preferred actions for the other policy in the corresponding states. If this condition is not satisfied, policies are said to be conflicting.

- Set of agent's state spaces s such that:
  - states are grouped into state space groups S based on the policy to which they are relevant.
  - agent's complete state space consists of all possible combinations of all state spaces in all policies that the agent is implementing.

3.2. Design and Implementation Challenges

There are number of open issues associated with applying CRL in multi-agent decentralized systems, such as the choice of appropriate rewards for actions performed, and dealing with the uncertainty present due to the lack of global knowledge and relying on sensors to obtain a picture of the world. Additional challenges arise from extending CRL to multiple policies, as this requires taking into account the heterogeneity of agents and the different policies that different agents can be implementing. Our research will focus on these challenges, specifically on the issues outlined in the following paragraphs, relating to three main elements of Multiple Policy CRL: state space, learning process, and collaboration.

STATE SPACE

- Should an agent's state space be separate for each policy, or should there be a combined state space for all policies?
  - How does either option influence deploying new policies at an agent? If state spaces are separate then a new policy deployed will have its own separate state space and should not affect other policies, however if the state spaces are combined, deployment of the new policy will involve changing the existing state space for existing policies to include combinations of the new states relevant to the new policy.
  - How does either option influence the complexity of the learning process? A combined state space will be much larger than individual state spaces and we need to investigate the impact of that on the performance of the system, on time needed to perform learning processes, and on quality of the actions learnt.

- How can we design adequate state space partitioning for a given policy?
  - Should state space partitioning be absolute, i.e., have fixed values, or should it be relative to the metrics observed at other agent's in the system? To illustrate this problem we can use an example of traffic congestion in UTC. If the system is optimizing for maximum vehicle throughput, the state space should contain information about the number of vehicles waiting at each traffic light. Absolute state partitioning would partition the state space by an actual number of vehicles, for example state A would be 0 to 10 vehicles, state B would be 11 to 50 vehicles and so on. In a relative state partitioning state space would be divided so that state A means high congestion, state B medium congestion, and state C low congestion, where high, medium and low will have different numerical values at different times, relative to the count of vehicles at other traffic lights in the system. We need to investigate which state partitioning technique is more efficient in optimizing the system's performance.
  - Does suitability of absolute and relative partitioning depend on the policy type and how? We need to investigate should we use the same partitioning technique for all policies, or are there policy characteristics that influence the suitability of the technique.

0-7695-2870-9/07 $20.00 © 2007 IEEE

LEARNING

- Should learning be done separately per policy or should an agent learn for all policies combined? Investigation of this relates to the above mentioned grouping of the states by policy or having a combined state space.
  - Does learning separately result in suboptimal behavior for combined policies? A separate learning process assumes that policies exist in isolation and that once an agent executes an action it affects only a single policy. We need to investigate how an action chosen to implement a single policy in isolation influence system performance in the presence of other policies.
  - How much does the learning for all policies simultaneously increase the complexity of learning and time needed for convergence towards optimal behaviour? How does the behaviour that emerges in this way compare to the behaviour obtained by separate learning? We need to investigate if and to what extent we will have to trade off time of convergence vs quality of the solution.
- What happens with the learning process when policy is triggered and when policy is put on hold? Does it affect the learning process for other policies? This question has to considered both in the case of separate and combined learning.

HETEROGENEITY OF AGENTS

In collaborative self-organizing techniques agents perform actions, receive feedback from the environment on performing them, and then exchange the information about the feedback received with their neighboring agents. The feedback protocol is relatively straight forward in single policy systems as all agents are working towards a common goal, but in multiple policy optimization the goals of the agent can differ. In order to investigate the implications of heterogeneity of agents we need to answer the following questions on learning process and feedback exchange.

- Multiple policy systems can be designed so that individual agents only have knowledge of the policies they are implementing, or so that each agent has knowledge of the policies any agent with which it exchanges feedback is implementing. What are the implications of either design decision on the exchange of the feedback, calculation of the overall reward that is taking that feedback into account, and quality of the learning outcome?
- If agents are aware of each other's policies should they be aware of relative priorities of their policies versus other agent's policies and how does priority specification affect the feedback protocol, calculation of the reward based on the feedback, and learning process? Agents should be able to learn to perform actions that are the best for the policy with the highest priority, even if they are not currently implementing that policy themselves and even if that policy is in conflict with the policies they are currently implementing, i.e. the action performed might not be optimal for their local policy. Learning process should prevent greedy local agents from dragging the system into converging towards suboptimal global performance only because that behaviour optimizes performance of an individual agent.

Answers to any of the above questions relating to state space, learning process and feedback exchange between heterogeneous agents are difficult to predict and should be tackled experimentally.

4. Research Testbed

Our initial application area for MPCRL is Urban Traffic Control (UTC). Research is currently being done on applying CRL to single policy optimization of traffic throughput in UTC [7]. We believe UTC is a suitable application area for MPCRL because real world adaptive traffic systems have multiple objectives to deal with apart from just maximizing traffic throughput. An example scenario is presented in Figure 3. Traffic lights, modelled as autonomous agents, cooperate in order to minimize the waiting time for vehicles in the system, while simultaneously prioritizing emergency vehicles and public transport vehicles.

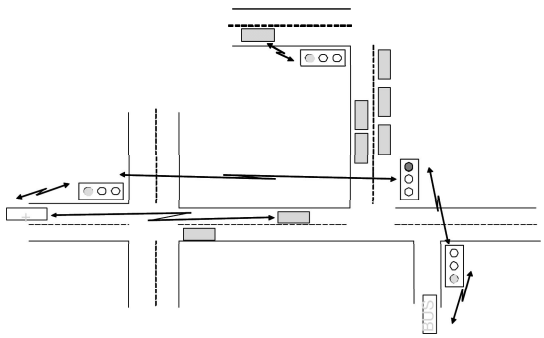

Figure 3. Application of Multiple Policy CRL in Urban Traffic Control

We plan to conduct a number of experiments on simulation of UTC system to answer the research questions

0-7695-2870-9/07 $20.00 © 2007 IEEE

discussed in 3.2. The experiments will initially involve a single junction with two or three approaches and two different policies, emergency vehicle priority and maximizing traffic throughput. The state space for emergency vehicle policy needs to include the information about the presence of the emergency vehicle at the approach, and the traffic throughput policy state space will contain the information about the number of vehicles queued. We will conduct experiments that simulate separate state spaces and learn for each policy (so traffic lights will learn the optimal cycle based on the length of the queues in one case, and based on the presence of an ambulance in the other case), as well experiments that simulate combined learning (where traffic lights will learn the optimal cycle taking both policies into account). Also, we will vary state space partitioning in this experiment to include queues of different absolute and relative length to test the influence of state space partitioning on learning processes. We will also vary the feedback that approaches exchange to include information about the state space, policies, and policy priorities of the sending and receiving agents, and vary the weight of feedback based on the policy priority.

Based on the results obtained from these experiments we plan to extended the simulation to include various additional policies on a global, regional or local level, such as priority of cars based on waiting time or on number of passengers, priority of public vehicles based on their timetable, priority of vehicle fleets, priority of traffic from or to certain areas etc.

5. Conclusion

Self-organizing algorithms are suitable for engineering large scale decentralized systems as they enable system to deal with the lack of global knowledge and central control. We believe these algorithms could find a wider application in engineering decentralized and ubiquitous systems once they are also capable of dealing with optimizing the system's performance towards multiple policies, rather than a single policy as the majority of existing techniques currently do. We have discussed a specific self-organizing technique CRL and have outlined a number of open research questions that have to be answered in order to apply it to multiple policy optimization. One of the major issues that arises is the heterogeneity of agents, meaning that agents need to collaborate to optimize global system performance, even if the performance towards their immediate local policy needs to be traded off. Additionally, we need to investigate the impact of multiple policies on the learning process, and evaluate the quality of system performance in the cases of separate learning processes for each policy or common learning process that encapsulates all policies simultaneously. In order to answer these questions, we will conduct a number of experiments by simulating multiple policy optimization of Urban Traffic Control systems using various parameters and relationships between MPCRL elements.

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] E. Camponogara and W. K. Jr. Distributed learning agents in urban traffic control. In *EPIA*, pages 324–335, 2003.

[3] G. D. Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communication networks. *Journal of Artificial Intelligence Research*, 9:317–365, December 1998.

[4] H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira. Learning multi-goal dialogue strategies using reinforcement learning with reduced state-action spaces. In *Proc. of Interspeech-ICSLP 2006*, 2006.

[5] G. Di Caro, F. Ducatelle, and L. M. Gambardella. AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications, Special Issue on Self-organization in Mobile Networking*, 2005.

[6] J. Dowling. *The Decentralised Coordination of Self-Adaptive Components for Autonomic Distributed Systems*. PhD thesis, Trinity College Dublin, 2005.

[7] J. Dowling, R. Cunningham, A. Harrington, E. Curran, and V. Cahill. Emergent consensus in decentralised systems using collaborative reinforcement learning. In *Self-star Properties in Complex Information Systems*, pages 63–80, 2005.

[8] D. Fudenberg and J. Tirole. *Game theory*. The MIT Press, 1991.

[9] Z. Gabor, Z. Kalmar, and C. Szepesvari. Multi-criteria reinforcement learning. In *International Conference on Machine Learning*, Madison, WI, July 1998.

[10] M. Humphrys. Action selection methods using reinforcement learning. In M. M. M. J.-A. P. J. Maes, P. and S. W. Wilson, editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, 1996.

[11] M. Lekavý. Optimising Multi-agent Cooperation using Evolutionary Algorithm. In M. Bieliková, editor, *Proceedings of IIT.SRC 2005: Student Research Conference in Informatics and Information Technologies, Bratislava*, pages 49–56. Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, April 2005.

[12] M. Li, J. Hallam, L. Pryor, S. Chan, and K. Chong. A cooperative intelligent system for urban traffic problems. In *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*, September 1996.

[13] S. Mikami and Y. Kakazu. Genetic reinforcement learning for cooperative traffic signal control. In *International Conference on Evolutionary Computation*, pages 223–228, 1994.

[14] M. D. Pendrith. Distributed reinforcement learning for a traffic engineering application. In *AGENTS '00: Proceedings of the fourth international conference on Autonomous*

0-7695-2870-9/07 $20.00 © 2007 IEEE

agents, pages 404–411, New York, NY, USA, 2000. ACM Press.

[15] S. Richter, D. Aberdeen, and J. Yu. Natural actor-critic for road traffic optimisation. In *Advances in Neural Information Processing Systems*.

[16] N. Sprague and D. Ballard. Multiple-goal reinforcement learning with modular sarsa. In *International Joint Conference on Artificial Intelligence*, August 2003.

[17] R. S. Suton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book. The MIT Press, Cambridge, Massachusetts, 2002.

[18] M. Wiering. Multi-agent reinforcement learning for traffic light control. In *Proc. 17th International Conf. on Machine Learning*, pages 1151–1158. Morgan Kaufmann, San Francisco, CA, 2000.

[19] Z.-S. Yang, X. Chen, Y.-S. Tang, and J.-P. Sun. Intelligent cooperation control of urban traffic networks. In *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, August 2005.

0-7695-2870-9/07 $20.00 © 2007 IEEE