AN ABSTRACT OF THE THESIS OF

Newsha Khani for the degree of Master of Science in Mechanical Engineering presented on May 29, 2009.

Title: Learning from Action Not Taken in Multiagent Systems

Abstract approved: _____

Kagan Tumer

Coordination in large multiagent systems in order to achieve a system level goal is a critical challenge. Given the agents' intention to cooperate, there is no guarantee that the agent actions will lead to good system objective especially when the system becomes large. One of the primary difficulties in such mulitagent systems is the slow learning process. Agents need to learn how to interact with other agents in a complex and dynamic system while adapting in the presence of other agents that are simultaneously learning. Presented in this thesis is a unique multiagent learning approach that significantly improves both learning speed and system level performance in multiagent systems by having an agent update its estimate of the reward (e.g., value function in reinforcement learning) for all its available actions, not just the action that was taken. This method is based on the agent receiving the reward for the actions they do not take by estimating the counterfactual reward it would have received had it taken

those actions. The experimental results illustrate that the rewards on such "actions not taken" are helpful early in the learning process. The agents then use their team members to estimate these rewards resulting in principally learning as a team. Finally, it is shown that fast learning is essential in a dynamic environment. The ANT reward with teams presents improvement in speed that results in more stability in following the changes in such an environment.

Learning from Action Not Taken in Multiagent Systems

by

Newsha Khani

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented May 29, 2009
Commencement June 2009

Master of Science thesis of Newsha Khani presented on May 29, 2009.

APPROVED:

Major Professor, representing Mechanical Engineering

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Newsha Khani, Author

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my supervisor, Dr. Kagan Tumer for all the time he has given toward my education and for his perpetual enthusiasm, patience and guidance during my research. I wish to extend my warmest thanks to Dr. Belinda Batten for always being supportive by her friendly help and valuable advice. I greatly appreciate all of the contributions of my graduate committee, especially for giving me a learning environment to grow me personally as well as professionally.

I would also like to thank my research partners in Adaptive Agents and Distributed Intelligence for their support. In particular, I am deeply indebted to my colleague Matt Knudson for his friendship, kindness and various helpful suggestions throughout my study at Oregon State University.

TABLE OF CONTENTS

		Page
1	Introduction	1
2	Related Research and Contributions	5
2.1	Reinforcement Learning	6
2.2	Multiagent Learning	13
2.3	Contributions of this Thesis	15
3	Problem Definition and Learning	17
3.1	Congestion Problems	17
3.2	Agent Learning Algorithms	19
3.3	Agent Rewards	20
4	Action Not Taken (ANT) Rewards	24
4.1	ANT Reward Derivation	25
4.2	Basic ANT Reward Performance	26
4.3	ANT Reward with Early Stopping	28
5	Computing ANT Rewards using Team Information	31
5.1	ANT Rewards with Teams	31
5.2	ANT Rewards with Weighted Teams	35
6	Dynamic Environment	39
6.1	Unpredictable Changes to the Environment	40
6.2	Periodic Changes to the Environment	41
7	Conclusion	44
7.1	Summary	44
7.2	Future Work	47
Bibliography		49

LIST OF FIGURES

Figure		Page
3.1	The structure of the learning algorithm for each agent which is a simple reinforcement learner.	19
4.1	System performance versus training weeks.	27
4.2	System performance when actions not taken are stopped after week 6.	29
4.3	The impact of the stop week on system performance. The learning speed is directly related to the length of time the Action Not Taken reward is used.	30
5.1	System performance when only a subset of actions are explored by an agent.	36
5.2	System performance versus training weeks.	37
5.3	System performance for the weighted team rewards.	38
5.4	The impact of congestion on system performance for the weighted team rewards.	38
6.1	System performance when the number of agents in the system changed from 280, 140, 180, 100 each 40 time steps, for 7 actions and a capacity of 4.	40
6.2	System performance when the capacity of the system changes from 3 to 7 and back randomly every 70 time steps for 4 actions and 120 agents.	41
6.3	System performance versus training weeks. There were 8 actions with a capacity of 5. The standard difference reward D and D_{WT} are plotted with variations of 90 to 150 in the number of agents.	42
6.4	System performance when the number of agents changes periodically. There were 8 actions and 120 agents and capacity changed from 2 to 5 every 50 weeks.	43

Chapter 1 – Introduction

Designing a controller for a distributed system is an important challenge. Examples of these types of engineering problems include scheduling in manufacturing systems [5, 30, 43, 48], routing in data networks [6, 26, 49, 63, 71], as well as command and control of networked forces in adversarial environments [61, 69, 70].

As the system becomes more complicated and is confronted with restricted communication, local information and uncertainties, applying control methods becomes a difficult process. Both the complexity and dispersed nature of these problems bring about the appeal of multiagent approaches. In such an approach, the decision-making is distributed among the connected agents, which offers potential solutions for complex problems.

Multiagent systems have applications in a wide variety of domains. Several successful applications include distributed control [28], robocup soccer [44], web and data mining [29, 57], and air traffic management [87]. In such multiagent systems, an agent has to learn to acquire skills to interact with other agents in typically complex, dynamic and possibly non-stationary environments.

What makes this problem particularly challenging is that the agents in the system provide a constantly changing background in which each agent needs to learn its task. As a consequence, the agents need to extract the underlying "signal" (e.g., what the right action is) from the noise of the other agents acting within the

same environment.

In an environment where multiple agents coexist, evaluating the task of each agent is difficult because an agent's performance is dependent on the actions of all the other agents in the system. To achieve the goal of maximizing the system level global objective as well as its individual sub goals, each agent is required to take into account the other learning agents.

A major topic of interest in this research is in developing methods that build coordination among the different contributions of the agents to the system. In such a cooperation manner, how to assign credit to the action of an agent to attain a better system performance is an important problem. As with the basic requisite, we need robust algorithms for learning in multiagent systems [37, 96]. Learning sequences of actions for multiple agents has been widely studied [3, 39, 41, 78, 91, 93, 101]. The effort in multiagent learning is to respond to the system objective while effectively considering the needs of individual agents.

Though these methods have yielded tremendous advances in multiagent learning, they are principally based on an agent trying an action, receiving an evaluation of that action, and updating its own estimate on the "value" of taking that action in a given state. Though effective, such an approach is generally slow to converge, particularly in large and dynamic environments.

In this thesis, we explore the concept of agents learning from actions they do not take by estimating the rewards they would have received had they taken those actions. These counterfactual rewards are estimated using the theory developed for structural credit assignment, and prove effective in congestion games. Furthermore,

a team structure can be used to provide the required information for the agents to compute these reward estimates [67, 82]. A key benefit of this approach is that an increase in the number of agents can be leveraged to improve the estimates of actions not taken, turning a potential pitfall (e.g., how to extract useful information from the actions of so many agents) into an asset (e.g., learn from the experiences of other agents).

Chapter 2 provides a background on multiagent systems. It also provides an overview of previous work and various studies in the area of learning in multiagent systems. It describes mathematical models of learning for both single and multi-agent systems as well as research in the area of learning where coordination and communication are decisive factors that benefit performance.

Chapter 3 introduces congestion problems as a good domain for studying multiagent learning. The definition of the congestion problem is presented and utilizes traffic as an illustration of the congestion concept. The chapter then discusses the multi-day bar problem which is an abstraction of the congestion problem that is used in the reported experiments. Finally, the basic agent learning architecture is described in addition to exploring three base reward functions for each agent.

In Chapter 4, the basic Action Not Taken reward is introduced. Derivation of the reward is described in detail along with the assessment of its performance in comparison to the previous rewards. The results show that although ANT reward learns faster than the preceding defined reward functions, it provides agents with extra information that deteriorates agent performance. Two possible solutions are presented: (i) use ANT reward early in the process, but stop and switch to a

basic reward after some stop week; (ii) and select only a subset of the actions to receive the Action Not Taken reward. Experimental results for the first option are provided in this chapter.

Chapter 5 focuses on the second alternative to improve the basic ANT reward. In order to provide required knowledge and limit the actions explored by agents, a team structure is used. The reward derivation is provided, followed by the experimental results and conclusions. In addition, a combination of two concepts of team formation and early stopping is developed along with an illustration of its benefit to performance. Furthermore, the modified form of ANT reward with weighted teams is presented and the experiments that are addressed show the impact of congestion on system performance.

Chapter 6 explores the application of these rewards to dynamic domains where the rapidly changing conditions put a premium on learning quickly. It describes how to implement the proposed methods in two types of dynamic environments. First, it examines the affect of random changes both in number of agents and capacity, and then it explores periodic changes of both types.

In Chapter 7 we discuss the results and provide directions for future research.

Chapter 2 – Related Research and Contributions

Learning in large multiagent systems is a critical area of research with a broad range of applications. Successful examples of application domains include network routing [14, 25] sensor networks [52, 60], robot soccer [44, 47, 75, 76] and additionally, information retrieval and management [46], rover coordination [58], trading agents [74, 100], and air traffic management [87]. Defining an appropriate algorithm in multiagent domains is a key in the learning process. Properly determining such a learning algorithm is required in order to obtain good coordinated behavior.

Various studies have investigated how to deal with the complexity of the learning problem. The feasibility to achieve the learning criteria by applying pure single-agent learning has been shown in [77]. However, the fact that the environment is non-stationary is usually ignored. Even if an agent ignores the involvement of other agents, its reward from the environment will be affected by their behaviors.

Typically, two learning problems are coupled where the agent needs to both solve a temporal credit assignment problem (how to assign a reward received at the end of sequence of actions to each action) and a structural credit assignment problem (how to assign credit to a particular agent at the end of a multiagent task) [3, 39, 41, 78, 91, 93, 101]. The temporal credit assignment problem has been extensively studied [24, 41, 78, 83, 84, 92, 97], and the structural credit assignment problem has recently been investigated as well [7, 19, 34, 59, 65, 89].

The equivalence between the structural and temporal credit assignment was shown in [1], and methods based on Bayesian analysis were shown to improve multiagent learning by providing better exploration capabilities [17]. Furthermore, learning sequences of actions for multiagent systems has blended structural and temporal credit assignment and has led to key advances [17, 19, 35, 75, 94]. In these cases, the learning needs of the agents are modified to account for their presence in a larger system [4, 34, 36, 64, 89, 90]. We now review reinforcement learning and multiagent learning, and then state the contribution of this thesis.

2.1 Reinforcement Learning

One of the popular approaches that has been developed is reinforcement learning. Reinforcement learning techniques are model-free methods that have to learn based on the feedback received while interacting with the environment. The agents learn what to do by repeatedly modifying their behavior based on the reward of their action while exploring the environment [12, 78]. At each step, an agent chooses an action on its current state. It takes the action, resulting in a change of state, and receives a subsequent reward. In reinforcement learning, it is required for an agent to have a policy in order to operate in an environment. The policy defines the probability of taking an action a when the agent is in state s (Equation 2.1).

$$\pi(s, a) = p(a_t = a | s_t = s) \tag{2.1}$$

To achieve the optimal behavior, the agent needs to determine how to select

a strategy to be aligned with its prospects. The agent should behave in a way that results in maximization of the sum of rewards it received during its entire performance. Several models have been created to achieve this:

- The finite horizon model offers the action chosen by the agent in a direction that maximizes its expected reward for the next h-steps. In the next step, the model proposes the action resulting in an optimal $(h-1)$-step reward, and so on, until it selects the last and best action and terminates (Equation 2.2).

- The receding-horizon-control model, in which the agent always chooses the h-step optimal action. In this approach, the agent elects how far it needs to look ahead so as to make the best decision about how to perform at present.

$$R = \sum_{0}^{h} r_t \tag{2.2}$$

- The infinite-horizon discounted model which is more functional than the finite-horizon model. Its desirability is due to the fact that agents' lifetimes in many cases are unknown. Therefore, the agent attempts to maximize the discounted sum of its rewards where a reward received k steps in the future is worth only γ^{k-1} $(0 < \gamma < 1)$ times what it would be worth now because there is more uncertainty. This concept is applied by considering a discount factor γ for the subsequent rewards.

$$R = \sum_{0}^{\infty} \gamma^t r_t \tag{2.3}$$

The basic algorithm for solving for the optimal policy, which is known as a Markov Decision Process, returns for each state the best possible action [11, 66]. The model is Markov if the state transitions are independent of any previous environment states and it retains all the information relevant to future actions and rewards. MDPs are important in reinforcement learning because decisions and values are based on agent observation of current state and new action. Formally the MDP model is defined by:

- S: set of states of the environment
- a: set of actions possible in state $s \in S$
- $P^a_{ss'}$: probability of transition from s to s' given a

$$P^a_{ss'} = p\left(s_{t+1} = s' \mid s_t = s, a_t = a\right) \tag{2.4}$$

- $R^a_{ss'}$: expected reward on transition s to s' given a

$$R^a_{ss'} = r\left(r_t \mid s_t = s, a_t = a, s_{t+1} = s'\right) \tag{2.5}$$

As we mentioned before, in multiagent learning the agent tries to maximize its overall reward. Thus, throughout the reinforcement learning process, the agent needs to assign an appropriate reward to each state. However, a reward measures an instantaneous return from the recent state that may be proceeded by several punishments which by considering subsequent reward does not define a good pol-

icy to follow. To consider these unpredictable alterations, the agent needs another way of evaluating observed states. In other words, it has to estimate the value of whichever state it is in. Given a state, the value function approximates the expected reward for the agent to be in that state. Also, to evaluate the future rewards there should be a previously defined map so that the agent can refer to it when calculating the forthcoming rewards. Therefore, the value of each state is dependent on a specified policy because the value is the expected future reward when starting from s and following the given policy. The mathematical representation of the value function under policy π based on MDPs for any state is defined as follows:

$$V^{\pi}(s) = E_{\pi}(R_t|s_t = s) \tag{2.6}$$

Furthermore, the value function for each state-action pair is presented as follows:

$$Q^{\pi}(s, a) = E_{\pi}(R_t|s_t = s, a_t = a) \tag{2.7}$$

The values play an important role for determining the best possible policy. The higher the values of chosen states, the better the task will be accomplished, resulting in a more suitable policy. Determining a good policy is critical in reinforcement learning. It is an agents duty to find an optimal policy and map states to best actions available to the agent. A policy π is better than a policy π' if its expected future reward V^{π} is greater than or equal to that of $V^{\pi'}$ for all states.

There always exists a policy that is higher or at least equal to all other policies in respect of overall expected rewards, which is called an optimal policy.

$$\pi \geq \pi' \Longleftrightarrow V^{\pi}(s) \geq V^{\pi'}(s) \quad \forall s \in S \tag{2.8}$$

As mentioned before, the optimal value function is based on an optimal policy. Hence, the optimal state-value function is defined as:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \tag{2.9}$$

Consequently, optimal action-value function is described as:

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{2.10}$$

One way to learn the state-value function is the Monte Carlo method which utilizes experiments to estimate the expected total future discounted outcome. Each agent averages the reward it received after any observation of a particular state. Through these experiences and by visiting the state more than one time, the agent makes better sense of the actual reward of that specific state. The value of a state is updated using Equation 2.11:

$$V(s_t) \longleftarrow V(s_t) + \alpha(r(s_t) - V(s_t)) \tag{2.11}$$

where α is a discount factor introduced to prevent value from diverging and to promote considering the recent rewards more than the previous ones to calculate

and attain the value. Normally α is set between 0 and 1.

Temporal difference learning is introduced to eliminate several deficiencies associated with Monte Carlo policy evaluation. One of the many concerns about Monte Carlo is the delay in receiving the reward in addition to ignoring the value of subsequent states:

$$V(s_t) \longleftarrow V(s_t) + \alpha((r(s_t) + \gamma V(s_{t+1}) - V(s_t)) \tag{2.12}$$

The SARSA algorithm [78], which is an extended temporal difference algorithm, considers a state-action function estimator rather than a state-value function. This method tries to converge to optimal function values of the policy currently being executed; ($Q(s_t, a_t)$ gives the value of taking action a from state s at time t)

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha(r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{2.13}$$

Another method that considers transitions from a state-action pair to another state-action pair and that yields a breakthrough in reinforcement learning is Q-learning [78, 98]. It approximates the optimal state-action function independent of the policy implemented. This algorithm for two-player zero-sum games is first introduced in [54]. Hu and Wellman developed an algorithm for two-player general-sum games [35]. The convergence proof is presented in [53, 55, 80]. Q-learning is policy-independent because it uses the best possible action after the current state. Q values are updated as follows:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha(r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)) \tag{2.14}$$

The main difference between SARSA and Q-learning is that to update the Q-values it is not necessary to use the maximum reward for the next state. Instead, SARSA follows the existing policy that determines which action to take, leading to attain a certain reward. However, in Q-learning, the value of the next state is independent of the policy and is based on searching through all the subsequent states to find the state possessing highest value.

Furthermore, in many applications in the real world, providing the agents with full information about their environment is impractical. Since an agent's observations from the world are neither complete nor adequate and provide only partial information about the current state, it makes it harder for the agent to come up with an appropriate decision. Moreover, full observation is essential for learning methods based on MDP. Therefore, an extension to the basic MDP framework, called a Partially Observable Markov Decision Process or POMDP, has been developed to solve this deficiency. A factored MDP also developed in [15, 16] is an MDP in which the transition functions are factorized by using a dynamic Bayesian network.

2.2 Multiagent Learning

From the single-agent perspective, existing learning techniques have proven to be powerful tools that can discover the dynamics of the environment and accomplish the established goal [21, 62, 85]. Most of the single-agent models for sequential decision making are derived from [41, 66, 78]. These learning models focus on how one agent improves its individual skills by repeatedly interacting with its environment. The decision of the agent is solely affected by its sequence of actions and the dynamics of its environments. However, existence of multiple agents in a system generates serious consequences and complexity to the solution of the learning problems. The behavior of each agent affects other agents either directly or indirectly. Thus for the agents to suitably function in a shared atmosphere, extensively modified methods need to be developed [79, 95, 99].

An extension of an MDP to multiple agents is a Multiagent Markov Decision Process (MMDP) [13]. This model considers two assumptions: first, each agent completely observes its state at any given time. Second, each agent receives the same global reward. Furthermore, the modified form of the single-agent POMDP for multiagent learning is a decentralized POMDP (DEC-POMDP) [10]. Agents observations are doubtful and given by a probability which depends on the previous state, current state, and previous action.

In terms of knowledge in multiagent systems, storing a separate array for each possible action is intractable. Thus, Dynamic Decision Network (DDN) [23, 33] was presented that also considers actions of the agents in the system.

The Q-learning algorithm also can be applied to each agent in multiagent systems by having each agent simply disregard the other agents. Alternatively, not being explicitly aware of other agents builds an uncertainty about achieving the optimal performance. This led to work on learning where coordination and communication is needed.

The distributed Q-learning algorithm [50] solves the cooperative task that is only suitable in the deterministic setting. Each agent sustains an explicit policy $h(a)$, and a local Q-function. These parameters are updated only in the aim of increasing the estimate of Q function.

$$Q_{t+1} = max\{Q_t(s_t, a_t), r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a_t)\} \tag{2.15}$$

$$h_{t+1}(s_t) = \begin{cases} a_t & \text{if } Q_{t+1}(s_t, a_t) \neq Q_t(s_t, a_t) \\ h_t(s_t) & \text{otherwise} \end{cases} \tag{2.16}$$

A common method to solve the coordination problem is to promote an agent to make their selected action more aligned with the preference of other agents. Coordination can be simplified by simply allocating the local Q-functions which only process the small set of agents and have all the Q-functions form a global Q-function [34, 47]. Communication is used to negotiate action choices and exchange memories of observed strategies by other agents, as in [40, 68, 95].

These methods have proven to be effective in multiagent learning. Generally exploring large areas, receiving incomplete information, having limited communication and making noisy observations make the learning problem more com-

plex [31, 32, 67]. As a consequence, with these restrictions the learning methods are slow to converge, leading to poor performance in static environments, and worse performance in dynamic environments.

2.3 Contributions of this Thesis

In this thesis, we present a theoretical result that signicantly improves the learning speed of the agents by allowing the agent to receive rewards based on Actions Not Taken (ANT). This increase in speed is based on the agent receiving a counterfactual reward that estimates the reward an agent would have received had it taken a particular action. In particular, we:

- Derive an implementable ANT reward for agents.
- Demonstrate the applicability of this method in a generic congestion problem.
- Provide improvements to the ANT reward based on
  - Early stopping and switching to basic reward.
  - Selecting only a subset of the actions to compute the ANT reward.
  - Using weighted teams to improve the ANT reward.
- Demonstrate the application of weighted team rewards to dynamic domains with:
  - Random changes

– Periodic changes

As mentioned, processing all the information about the environment while all the agents influence others, as well as generating adequate performance especially in large and dynamic environment is hard to accomplish. Learning methods, even if effective, are required to have rapid responses when applied to real world applications in order to result in good behavior. One of the well-known examples of these applications, where faster convergence can be important, is robot navigation in which a learning algorithm is used for path planning purposes [20, 78]. Moreover, increasing aircraft performance through better segmented surfaces controlled by applying theory of collectives [89, 102] presented in [73] is another example. We propose a method to modify a standard difference reward function leading to progress in both the learning speed and the quality of the solution reached.

Chapter 3 – Problem Definition and Learning

3.1 Congestion Problems

An interesting domain in which to study the behavior of cooperative multiagent systems is congestion problems where system performance depends on the number of agents taking a particular action [9, 27, 56].

When the demand for a particular resource is more than the capacity of that resource, congestion occurs. Various types of congestion problems have been introduced such as computer networks, traffic, event-driven sensor networks, etc [72]. In computer networks, there are a large number of resources, such as buffers, link bandwidths, processor times, servers, and so forth. At the time the queue lines become very long and exceed the capacity, the system becomes a congestion problem. The transport of event impulses in event-driven sensor networks is likely to lead to congestion in the network [22]. Congestion as a traffic problem affects the schedule of people, human life, transportation, and economy and business activities. It influences air pollution, gas consumption and the environment [51, 81].

In congestion problems, agents need to learn how to synchronize (or not synchronize) their actions, rather than learn to take particular actions. The agents do not reach good rewards simultaneously, but since the actions of other agents are unpredictable, finding the best solution is challenging. This type of problem is

ubiquitous in routing domains (e.g., on a highway, a particular lane is not preferable to any other lane, but what matters is how many others are using a particular lane) [45, 88].

The multi-day bar problem is an abstraction of congestion games (and a variant of the El Farol bar problem [8]) and has been extensively studied [3, 8, 18, 38]. In this version of the congestion problem, each agent has to determine which day in the week to attend a bar. The problem is set up so that if either too few agents attend (boring evening) or too many people attend (crowded evening), the total enjoyment of the attending agents drop.

The system performance is quantified by a system reward function G. This reward is a function of the full system state z (e.g., the joint action of all agents in the system), and is given by:

$$G(z) = \sum_{day=1}^{n} x_{day} e^{\frac{-x_{day}}{C}} \tag{3.1}$$

where:

n is the number of actions (for example $n = 7$ if actions are days).

x_{day} is the total attendance on a particular day, and

C is a real-valued parameter that represents the capacity of the resource (e.g., the capacity of the bar).

What is interesting about this problem is that selfish behavior by the agents tends to lead the system to undesirable states. For example, if all agents predict

an empty bar, they will all attend (poor reward) or if they predict a crowded bar, none will attend (poor reward). This aspect of the bar problem is what makes this a "congestion game" and an abstract model of many real world problems ranging from lane selection in traffic to job scheduling across servers to data routing.

3.2 Agent Learning Algorithms

The agent action in this problem is to select a resource (day on which to attend the bar). The learning algorithm for each agent is a simple reinforcement learner (action value). Each agent keeps an n-dimensional vector providing its estimates of the reward it would receive for taking each possible action. The system dynamics are given by:

Initialize: week 0
Repeat until week > Max week

1. Agents choose actions
2. Agents' joint action leads to an overall system state
3. The system state results in a system reward
4. Each agent receives a reward
5. Each agent updates its action selection procedure (i.e., learning)
6. week $\leftarrow$ week + 1

Figure 3.1: The structure of the learning algorithm for each agent which is a simple reinforcement learner.

In any week, an agent estimates its expected reward for attending a specific day based on action values it has developed in previous weeks. At the beginning of each training run, each agent has an equal probability of choosing each action in the first week, resulting in a uniformly random distribution across actions. At the beginning of each training week, each agent picks a day to attend based on sampling this probability vector using a Gibbs distribution. Each agent has n actions and a value V_k associated with each action a_k:

$$P_k = \frac{e^{(V_k \cdot \beta)}}{\sum_{agent} e^{(V_k \cdot \beta)}} \tag{3.2}$$

where β is a temperature term that determines the amount of exploration (low values of β mean most actions have similar probabilities of being selected, whereas high values of β increase the probability that the best action will be selected). Each agent receives reward R and updates the action value vector using a value function V_k:

$$V_k = (1 - \alpha) \cdot V_k + \alpha \cdot R \tag{3.3}$$

3.3 Agent Rewards

In this thesis, we look at three "base" reward functions for the agents: Local, System and Difference. The first option for agent rewards is the **local reward** that focuses on an agent's selfish objectives. For each agent, the local reward function must reflect the reward for the action that it took on each day. This leads

to:

$$L = x_{day_i} e^{\frac{-x_{day_i}}{b}} \tag{3.4}$$

where day_i is the day agent i has chosen to attend, and x_{day_i} is the attendance on that day.

This reward function is highly sensitive to the actions of the agent, as it is based only on the action that the agent chooses and observations the agent makes regarding the attendance once the action has been taken. However, it is not aligned with the system reward as for each week it does not reflect the actions that other agents made and the impact of those actions on the agents' reward.

The second option is to provide each agent with the **full system reward** for each week. This leads to each agent receiving the reward given in Equation 3.1, and using that reward to update its value estimates for each action. Though this reward is by definition aligned with the system reward (it is in fact the same reward), it is not particularly sensitive to an agent's actions and may therefore be difficult to directly optimize.

The third option is to provide each agent with a **difference reward** that reflects its contribution to the overall system reward. The difference reward is given by:

$$D^i(z) = G(z) - G(z - z_i) \tag{3.5}$$

To specify the part of the system state controlled by agent i and agents $-i$, we

use the notation z_i and $z - z_i$ respectively, where $z - z_i$ contains all the variables not affected by agent i, in other words it specifies the state of the system without agent i[1]. Difference rewards are aligned with the system reward because the second term on the right hand side of Equation 3.5 does not depend on agent i's actions [87, 89]. Furthermore, it is more sensitive to the actions of agent i, reflected in the second term of D, which removes the effects of other agents (i.e., noise) from agent i's reward function.

Intuitively, this causes the second term of the difference reward function to evaluate the performance of the system without i, and therefore D measures the agent's contribution to the system reward directly. For the difference reward in the congestion problem, this amounts to having each agent estimate the system reward it would receive were it to take or not take a particular action. In this thesis, agents do not explicitly communicate with one another, and therefore, the only effect each agent has on the system is to increase the attendance, x_{day}, for day k by 1. This leads to the following difference reward:

$$\begin{aligned} D^i(z) &= G(z) - G(z - z_i) \\ &= x_{day_i} e^{\frac{-x_{day_i}}{C}} - (x_{day_i} - 1) e^{\frac{-(x_{day_i}-1)}{C}} \end{aligned} \tag{3.6}$$

where x_{day_i} is the total attendance on the day selected by agent i.

As a consequence, we use the **difference reward** as a starting point for the

[1]In this thesis, we will use zero padded vector addition and subtraction to specify the state dependence on specific components of the system.

reward an agent receives after each step. Earlier works have shown that the difference reward significantly outperforms both agents receiving a purely local reward and all agents receiving the same system reward [2, 3, 86].

Chapter 4 – Action Not Taken (ANT) Rewards

Though the difference reward given in Equation 3.6, provides a reward tuned to an agent's actions, it is still based on an agent sampling each of its actions a (potentially large) number of times. In this thesis, in order to increase the learning speed, we introduce the concept of Action Not Taken rewards, or ANT rewards. The goal with ANT rewards is to provide estimates of how the system would have turned out had an agent taken a particular action. The mathematics that allows the computation of the difference reward can be used to compute this type of reward.

In this thesis, rather than have a separate results section, we provide experimental results directly alongside the reward descriptions to motivate the improvements to the rewards and the derivation of new rewards. All results are based on 20 independent runs with the standard error plotted when large enough to be relevant. Unless otherwise specified (as with the scaling runs or congestion dependent runs), the number of agents in the system was set to 120, with $C = 6$ (capacity) and $n = 5$ (number of actions, or days).

4.1 ANT Reward Derivation

In many multiagent domains, the learning agent does not have a complete observation of the whole world. Normally agents do not have the rigorous analysis of the impact of other agents on the environment in which the agent is embedded. Accordingly, unawareness of the other agents involved with changes in the environment creates problems. Moreover, the learning process is based on an agent trying an action, receiving an evaluation of that action numerous times up to the point that agent can get a better sense of the response of its environment to a specific action, which causes the learning to be slow. We introduce a new approach to speed up the learning by having an agent gain benefit from the outcome of other agents operating in the same environment. The direct application of this concept is to have agents update their reward estimate based on the reward they would have received had they taken other actions.

Therefore, at each time step, agents perform a mathematical operation that simulates their taking a different action and compute the counterfactual reward that would have resulted from that action. For an agent i who selected action a at this step, the counterfactual reward for action b is given by:

$$D^{i\to b}(z) = G(z - z_i^a + z_i^b) - G(z - z_i^a) \tag{4.1}$$

where

$D^{i\to b}$ is the reward for agent i taking action b;

z_i^a is the state component where agent i has taken action a;

z_i^b is the state component where agent i has taken action b.

The second term of Equation 4.1, $G(z - z_i^a)$, is the same as the second term of Equation 3.5. Namely the reward for the state where agent i has not taken the particular action that it took. The first term though is the key to the ANT reward. In this case, we compute the reward that would have resulted had agent i taken action b rather than action a.

Utilizing this structure, D_{ANT}^i can then be formulated as follows:

$$D_{ANT}^i = \begin{cases} G(z) - G(z - z_i^a) & \text{for } i \rightarrow a \\ G(z - z_i^a + z_i^b) - G(z - z_i^a) & \text{for } i \rightarrow b \neq a \end{cases} \tag{4.2}$$

where $i \rightarrow a$ means that agent i has taken action a. Note, the removal of the state in which agent i has taken action a in the second term represents the system state without agent i. Because agent i had taken action a, this removal results in a state where agent i has taken neither action a nor action b (which it has never taken). Hence the second term is the same for both conditions of Equation 4.2.

4.2 Basic ANT Reward Performance

Figure 4.1 shows the learning curves for D and D_{ANT} along with results where agents directly use the system reward G and a local reward L to learn. The local reward L is based on the agents simply receiving the reward for the action they took; in this instance, it is the component of Equation 3.1 corresponding to the

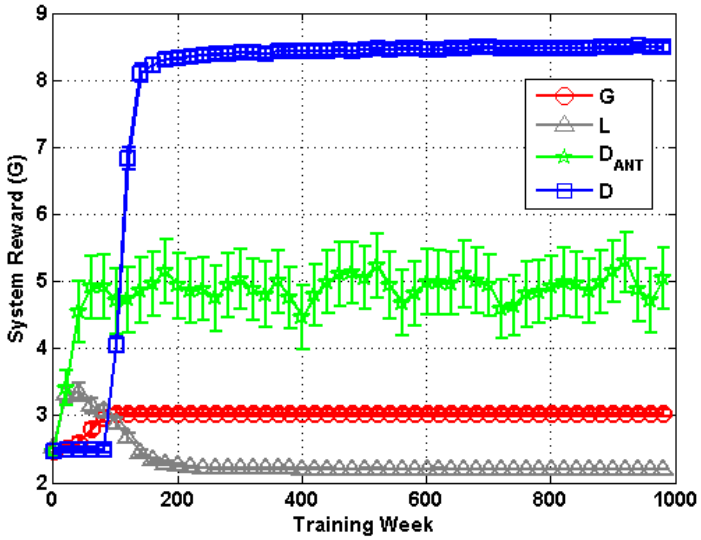

Figure 4.1: System performance versus training weeks.

day they attended the bar ($L = x_{day}e^{\frac{-x_{day}}{c}}$).

In all the experiments, the system performance is measured with respect to G, regardless of how the agents were trained. As previously noted, these results confirm that agents using D significantly outperform agents using G or L in this domain. G learns little, and L learns to do the wrong thing: The agent rewards are not aligned; agents aiming to maximize their own reward lead to poor system states. We include the results for agents using G and L here for completeness, but we will omit them in subsequent figures.

The results here show that although D_{ANT} learns faster than D, it struggles to reach good solutions and shows a lower and noisier performance. This shows that the Action Not Taken reward has a difficult time estimating the reward for

most actions once those actions have been sampled. In spite of being beneficial to learning speed, informing the agents about other available actions and related rewards turns out to be more confusing. Even though the agents take advantage of such rewards and learn faster in the first weeks of training, there is a time after which these additional rewards become detrimental to the learning process. This problem is due to the fact that agents do not always need to be aware of estimate of all actions; however, they need to get assistance from other agents to light the road ahead. On the other hand, if all the roads become lit, finding the best way will turn into an annoying concern for the agent and will add to complexity to learning procedure. This suggests two possible solutions, which we explore in the next two sections:

1. Use the ANT-reward early in the process, but stopping and switching to basic D after a "stop week" (described in section 4.3).

2. Select only a subset of the actions to receive the Action Not Taken reward (described in chapter 5).

4.3 ANT Reward with Early Stopping

First, let us consider the early stopping concept to mitigate the noisy feedback agents receive for their actions. This modification is based on the observation that the Action Not Taken rewards are better than random rewards, but not as good as rewards that have been updated by actually taking the actions. As shown

in Figure 4.2, having agents use ANT-rewards early significantly speeds up the learning process, though does not result in agents reaching higher performance.

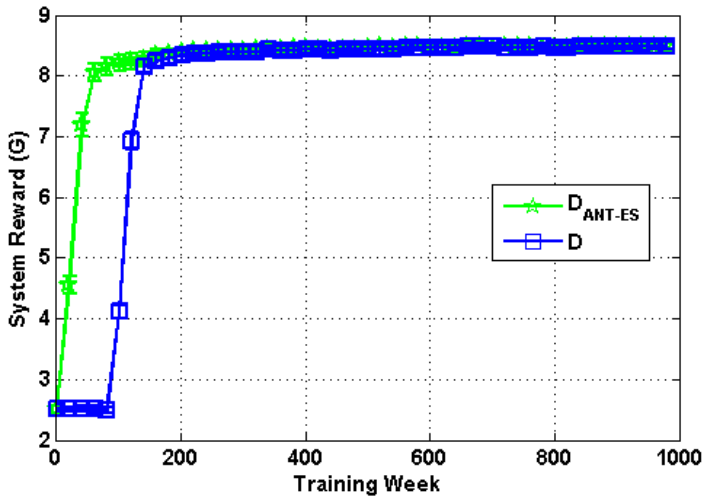

Figure 4.2: System performance when actions not taken are stopped after week 6.

Figure 4.3 shows the dependence of the system performance on the length of time the Action Not Taken reward is used. The learning speed is stable for small values of the stop week, but starts to drop slowly as the actions not taken are used more extensively. There is a steady rightward shift as the stop week moves from 6 to 100, at which point, the system learns more slowly than D alone.

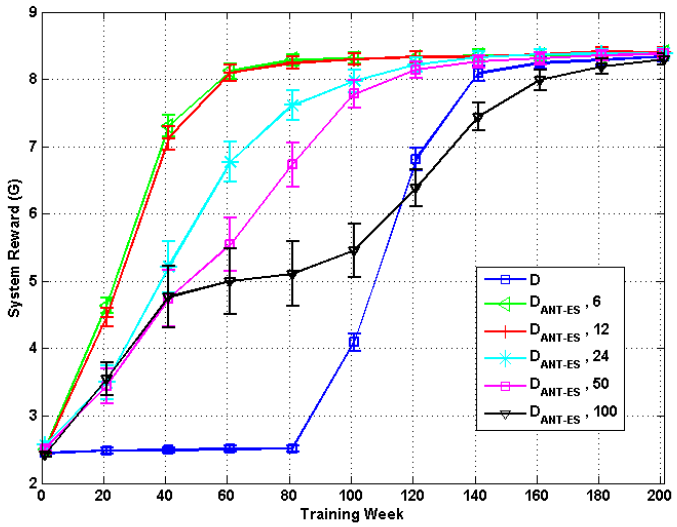

Figure 4.3: The impact of the stop week on system performance. The learning speed is directly related to the length of time the Action Not Taken reward is used.

Chapter 5 – Computing ANT Rewards using Team Information

As discussed before, the first approach to using the rewards of other agents was to stop early, before the reward become noisy. Now we investigate the second option, which is to include only some agents in the computation of the ANT rewards.

5.1 ANT Rewards with Teams

In this section we limit the actions that an agent updates based on counterfactual rewards to reliable actions sampled by agent i's team members (some subset of agents form a team). This formulation is given by:

$$D^i_{ANT-L} = \begin{cases} G(z) - G(z - z^a_i) & \text{for } i \rightarrow a \\ G(z - z^a_i + z^b_i) - G(z - z^a_i) & \text{for } i \rightarrow b \in T_i \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

Where $i \rightarrow b \in T_i$ means agent i selects actions b that are sampled by agent i's teammates T_i. As previously, the removal of agent i in the second term represents the system state without agent i having taken either action a (which it had taken) or action b (which it had not taken), leading to the term being the same in both cases.

Furthermore, instead of using the team members as information sources only,

we increase the connections among team members by providing them all with the same reward. That is, all team members attending a particular day will receive the same reward. The learning strategy is to use team information only during the first weeks (3 in the reported results, but the performance is similar for minor changes to this parameter) of learning and switch to the regular difference reward (Equation 3.6) for the rest of the training period.

The key aspect of this approach is that the team members measure the impact of a team not taking a particular action, rather than an individual agent. As a result, agents learn with their team in a smaller state space defined by the world minus their team space instead of the entire world. This is conceptually similar to the reward described in Equation 5.1 but where the impact of the whole team, rather than agent i is removed, leading to:

$$D^i_{Team} = \begin{cases} G(z) - G(z - z^a_{T_i}) & \text{for } T_i \rightarrow a \\ G(z - z^a_i + z^b_j) - G(z - z^b_{T_i} - z^a_i) & \text{for } i \rightarrow b \in T^i \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

Where $z^a_{T_i}$ is the state component of team members of agent i taking action a. In this formulation, the impact of all of agent i's teammates are removed before the reward is calculated. Note in this case, unlike in Equations 4.2 and 5.1, the second term is different for the two actions. This is because this term estimates the impact of removing all team members of i that had taken a particular action. When agent i changes its action, this also changes the team members taking the same action as i. For the action a selected by agent i, we only need to remove

all its team members who took that action. But to find the counterfactual reward for action b, we need to remove the actual action of agent i (action a) and then remove the team members who had taken action b. Though conceptually similar to previous rewards, the presence of team members leads to this subtle difference in the computation of the team Action Not Taken reward.

Now, let us explicitly compute D^i_{Team} for the congestion problem considered in this thesis. First, for the action taken by agent i (first line of Equation 5.2), the reward becomes:

$$\begin{aligned} D^{i\to a}_{Team} &= G(z) - G(z - z^a_{T_i}) \\ &= \sum_{day} x_{day} e^{\frac{-x_{day}}{C}} - \left(\sum_{day \neq day_i} x_{day} e^{\frac{-x_{day}}{C}} \right. \\ &+ \left. \left(x_{day_i} - \left| T^i_{day_i} \right| \right) e^{\frac{-(x_{day_i} - \left| T^i_{day_i} \right|)}{C}} \right) \end{aligned} \tag{5.3}$$

where:

$z - z^a_{T_i}$ is the state component in which agent i and its teammate taking action a have no effect;

day_i is the day agent i selects to attend;

x_{day_i} is the attendance on the day agent i selects to attend; and

$\left| T^i_{day_i} \right|$ is the number of agent i's teammates that choose day_i to attend.

Second, let us focus on the actions not taken by agent i (second line of Equation 5.2). This is the reward agent i would have received had it taken the actions b chosen by some of its teammates, leading to:

$$\begin{aligned} D_{Team}^{i\to b} &= G(z - z_i^a + z_i^b) - G(z - z_{T_i}^b - z_i^a) \\ &= \sum_{day \neq day_{i\to a,b}}^{day} x_{day} e^{\frac{-x_{day}}{C}} + (x_{day_{i\to a}} - 1) e^{\frac{-(x_{day_{i\to a}} - 1)}{C}} \\ &+ (x_{day_{i\to b}} + 1) e^{\frac{-(x_{day_{i\to b}} + 1)}{C}} \\ &- \left(\sum_{day \neq day_{i\to a,b}} x_{day} e^{\frac{-x_{day}}{C}} + (x_{day_{i\to a}} - 1) e^{\frac{-(x_{day_{i\to a}} - 1)}{C}} \right. \\ &+ \left. \left(x_{day_{i\to b}} - \left|T_{day_{i\to b}}^i\right|\right) \cdot e^{\frac{-\left(x_{day_{i\to b}} - \left|T_{day_{i\to b}}^i\right|\right)}{C}} \right) \end{aligned} \tag{5.4}$$

where:

$z - z_i^a + z_i^b$ is the state component in which agent i takes action b rather than action a;

$z - z_{T_i}^b - z_i^a$ is the state component on which agent i (taking action a) and its teammates taking action b are removed from the state;

$x_{i\to b}$ is the attendance resulting from agent i taking action b;

$\left|T_{day_{i\to b}}^i\right|$ is the number of agent i's teammates that choose to attend on day resulting from action b.

In this formulation, if agent i's team members have taken all the possible actions, each action that agent i had not taken will still be updated. Otherwise, only actions taken by i's teammates will be available for reward information and therefore updated.

Team ANT Reward Performance:

Figure 5.1 shows the results when an agent randomly selects team members. By limiting the number of actions that are updated, the variability of the reward is reduced, but there is no discernable improvement in the quality of the solution. However, from a computational and communication perspective, this is an interesting result, which points to a significant reduction in the need for counterfactual reward computation without loss of convergence speed.

Figure 5.2 shows the learning curves for D and D_{TEAM}. Agents using D_{TEAM} not only learn faster, but also reach higher system rewards than agents using the baseline D. In this instance, not only information from team members was used, but the reward of each team member was the same, resulting in a larger "block" of agents receiving a reward, and removing a significant amount of noise from the rewards.

5.2 ANT Rewards with Weighted Teams

The use of team rewards provided tangible benefits, though all information received from team members was treated equally. Yet, one can consider that the more team

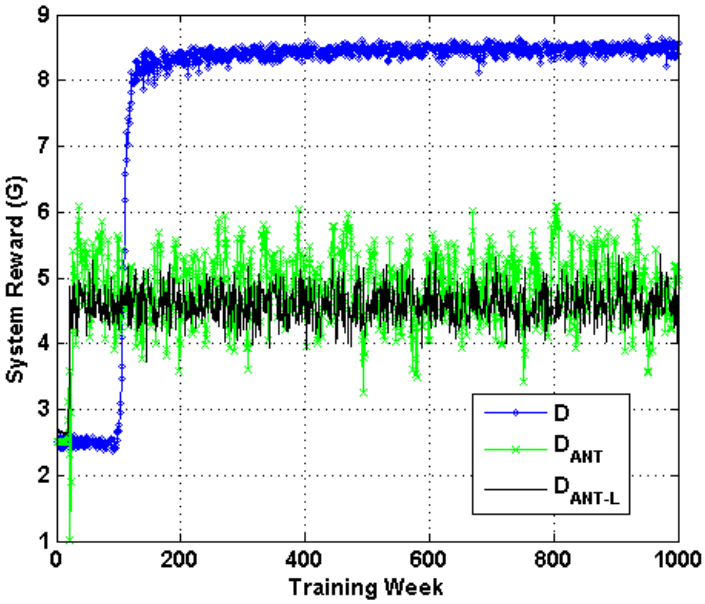

Figure 5.1: System performance when only a subset of actions are explored by an agent.

members take a particular action, the more reliable the estimate for the reward of that action would become. This becomes particularly relevant when the congestion in the system increases.

A simple solution to this problem is to use a weighting factor for the second term of the counterfactual reward function. In this thesis, we use the average number of team members selecting particular actions, though more sophisticated methods can also be used. This leads to modifying Equation 5.2, that for agent i and action b leads to a weighted team reward D_{WT}:

$$D_{WT}^{i \to b} = G(z - z_i^a + z_i^b) - \mu_{\left|T^i_{day_{i \to b}}\right|} \cdot G(z - z^b_{T_i} - z_i^a) \tag{5.5}$$

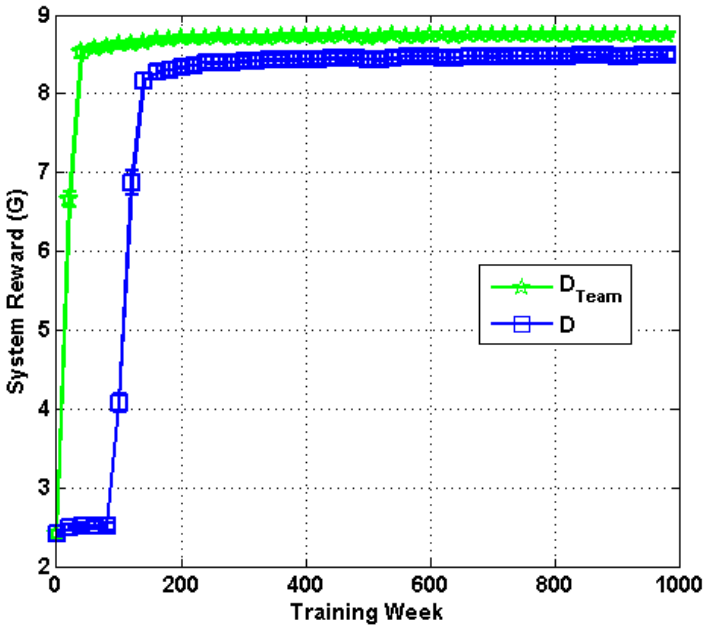

Figure 5.2: System performance versus training weeks.

where $\mu_{\left|T^i_{day_{i\to b}}\right|}$ is the average number of team members taking action b.

Weighted Team ANT Reward Performance:

Figure 5.3 explores this idea for 280 agents in a system with 5 actions and a capacity of 6. Because the optimal capacity in this case is $6 \times 5 = 30$, this creates significant congestion. The results show that traditional D starts to suffer in this case, and that the weighted D_{WT} outperforms D_{TEAM}. Figure 5.4 shows the impact of congestion directly as the number of agents in the system increases from 120 to 280. D_{WT} handles the increased congestion better than D.

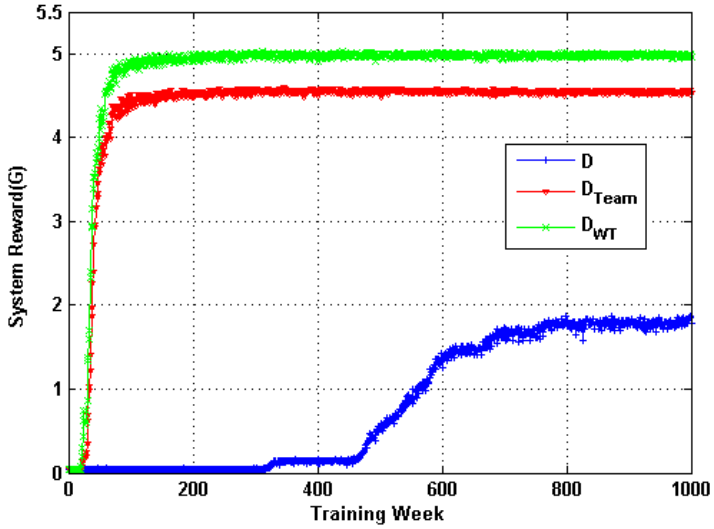

Figure 5.3: System performance for the weighted team rewards.

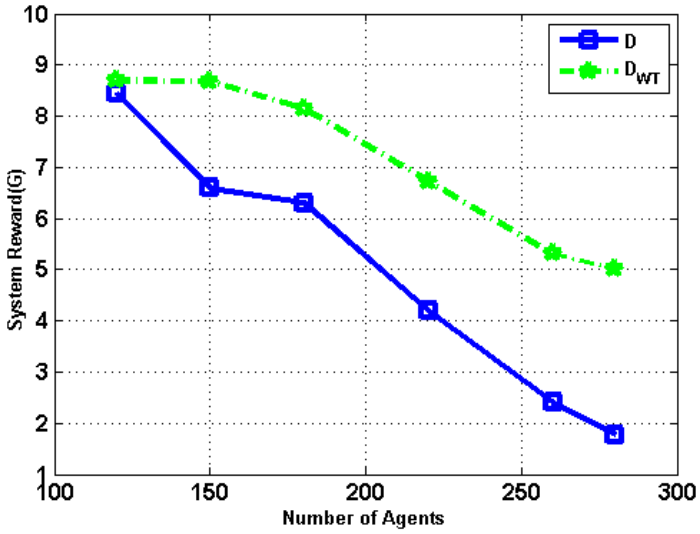

Figure 5.4: The impact of congestion on system performance for the weighted team rewards.

Chapter 6 – Dynamic Environment

One of the key advantages to rapid learning is the ability to adapt to dynamic environments where the conditions may change faster than a traditional learner can adapt. In this chapter, we test the performance of the Action Not Taken rewards with weighted team reward on two types of dynamic environments. First, we explore seemingly random changes in agent numbers and capacities, and then we explore faster, but periodic changes of both types.

Throughout the learning process, all the agents coherently pick their actions and their desire is to accomplish their contribution to the system to the maximum. Such mentioned learning process compels the agents to adjust to the changes in the environment. As a consequence, the average range of time for agents to learn the model of its changing environment increases and learning process raises a robust strategy to interact with other agents. The problem becomes more complicated when in addition to the changes of the action of other agents, the environment itself changes. In this section, we explore the ability of D_{WT} to adjust to unexpected changes in the system.

6.1 Unpredictable Changes to the Environment

Figure 6.1 shows the system response to changes in the number of agents. In this case, the number of agents changed every 40 weeks from 280, to 140, to 180, to 100. As demonstrated at week 120, D_{WT} not only recovers rapidly, but learns to exploit the new condition: after the initial drop caused by the change, agents using D return to their previous state, but agents using D_{WT} reach a higher system reward value.

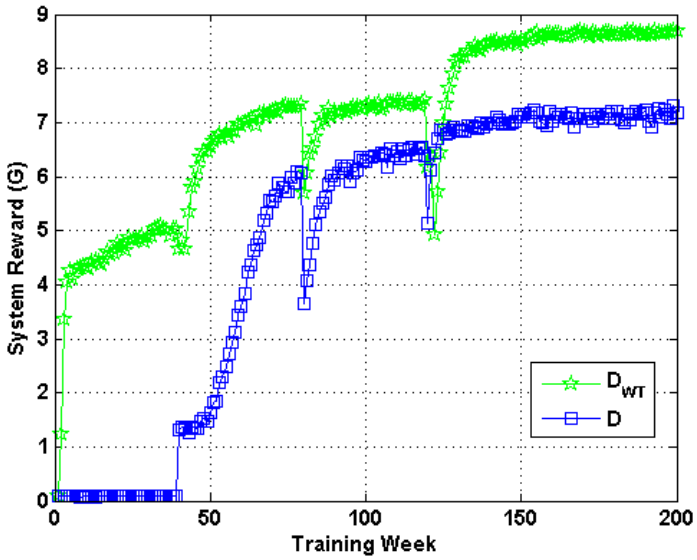

Figure 6.1: System performance when the number of agents in the system changed from 280, 140, 180, 100 each 40 time steps, for 7 actions and a capacity of 4.

Figure 6.2 shows the system response to the capacity changing from 3 to 7 every 70 weeks. D_{WT} learns faster early on and reaches slightly higher performance, but this experiment shows that D also can track slow changes in the environment.

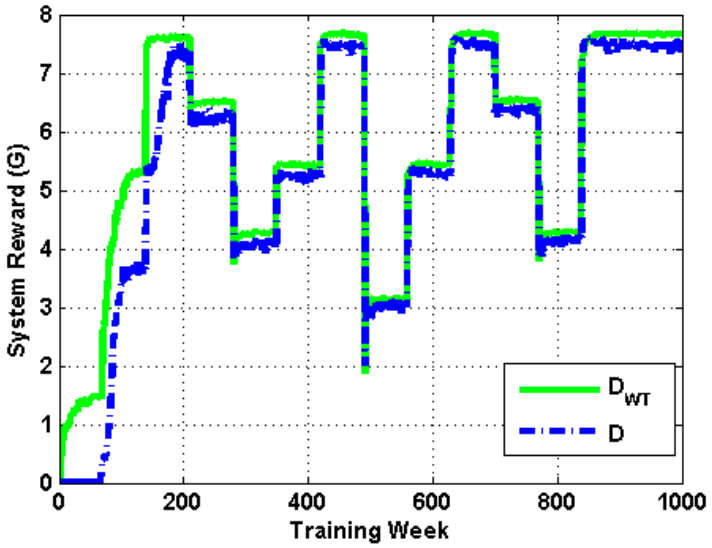

Figure 6.2: System performance when the capacity of the system changes from 3 to 7 and back randomly every 70 time steps for 4 actions and 120 agents.

6.2 Periodic Changes to the Environment

In this section we explore rapid periodic changes to the environment. Figure 6.3 shows the performance of D_{WT} versus difference reward D when the number of agents is changing rapidly. Unlike in the results of the previous section (Figure 6.1), D has a hard time tracking these changes. D_{WT} on the other hand converges to a good solution despite the fact that the number of agents in the system changes the optimal solution for each agent.

For this experiment, we modified the value update function to account for the

periodicity of the system, and allowed the value update to be:

$$V_k = (1 - \alpha) \cdot (\beta \cdot V_k^{t-1} + (1 - \beta) \cdot V_k^{t'}) + \alpha \cdot R \tag{6.1}$$

where t' corresponds to the last time in which the capacity was the same as currently. This value can be estimated in practice, though in this instance, in order to remove the impact of such estimation on the reward analysis, we applied it to both reward functions.

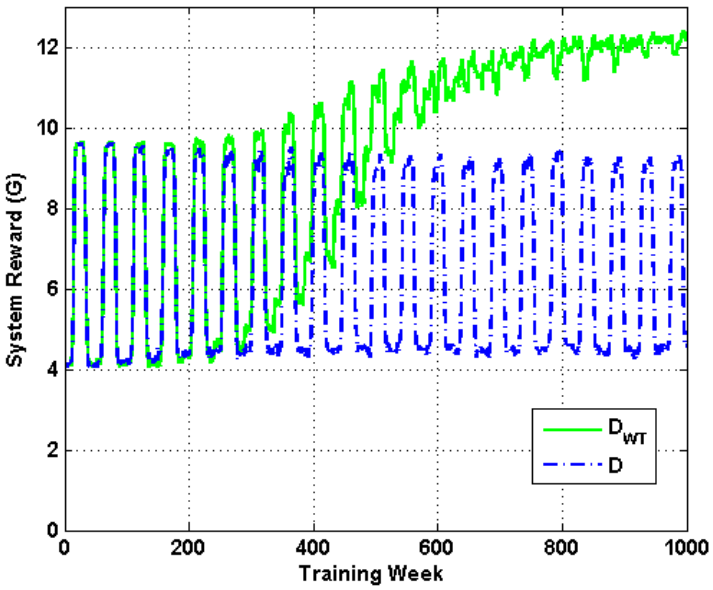

Figure 6.3: System performance versus training weeks. There were 8 actions with a capacity of 5. The standard difference reward D and D_{WT} are plotted with variations of 90 to 150 in the number of agents.

Finally, we explore the impact of rapid changes to the system capacity. Fig-

ure 6.4 shows the system performance when the capacity oscillates between 2 and 5. Unlike in Figure 6.2, D cannot track this continuous change as it does not get sufficient time to learn the system before the environment changes. D_{WT}, however, tracks the changes. Even though it has difficulties with the rapid changes, it reaches higher system level performance for both $C = 2$ and $C = 5$.

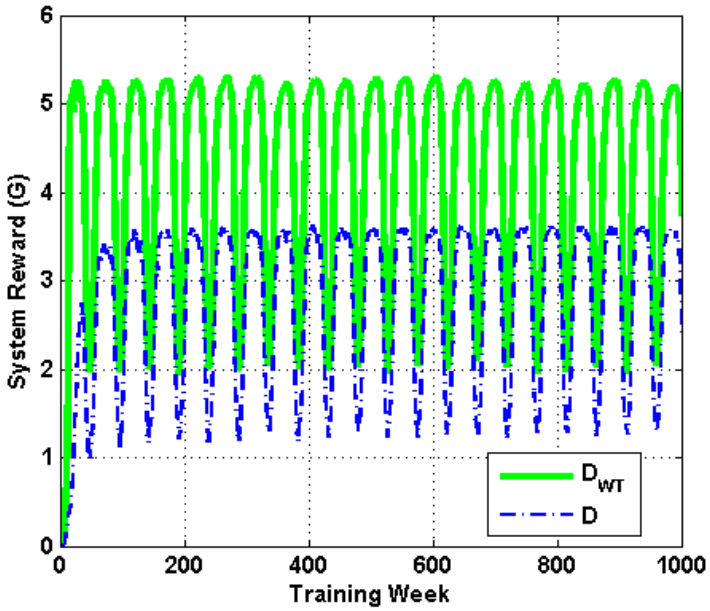

Figure 6.4: System performance when the number of agents changes periodically. There were 8 actions and 120 agents and capacity changed from 2 to 5 every 50 weeks.

Chapter 7 – Conclusion

Designing a controller for a large distributed system, containing many complex tasks to be accomplished, is a critical challenge. There are various types of engineering problems struggling with this issue. As the system is not determined involving restricted and local information, applying control methods in such a complicated system is not easy to achieve. The use of multiagent approaches is helpful in these domains where it provides an improved ability to better handle the complexity of large distributed systems.

7.1 Summary

In large complex and dynamic multiagent systems, the actions taken by the agents produce significant levels of noise to each agent trying to learn how its own actions affect the system objective. The agents are simultaneously acting in a shared environment while lacking information about what other agents are aiming to do. As a result, their actions must be combined with actions of all agents and the environment before leading to a reward. Although the reward provides each agent with feedback about a specific action, it also indirectly reflects other agents actions. As a consequence, an agent has a lengthy period where actions need to be sampled a large number of times to extract the "signal" from the "noise." Therefore defining

an adequate reward function plays an important role in learning performance. Traditional learning functions used for providing action rewards and subsequent selection values, such as using only locally available information or features of the entire system break down in large systems. The difference reward was shown to signicantly outperform these local and system objectives. The use of a difference reward was shown to successfully recover the performance of agent learning in large complex systems by providing more useful and distinguished information about the reaction of the environment to an action through the benefit of utilizing the direct effect of agents individual actions on the system.

For very large systems however, a standard difference reward function, while effective, is still based on an agent taking an action and receiving the subsequent reward to maintain alignment with the system objective as a whole. In this thesis, we present a modification to previously utilized difference reward functions, called the Action Not Taken reward. It provides the agents with rewards on actions that were not taken by the agent. To demonstrate the result of this new approach we chose an abstract of congestion problems known as the"Bar Problem," in which agents need to collaborate with each other to reach the desired system objective.

In order to improve the performance of the basic ANT reward, we investigated two possible solutions: First, taking advantage of the ANT reward early in the process and then using previous difference reward for the rest of the learning procedure. The second proposed method is to only consider a subset of actions for each agent, to be updated at each time step to develop a model of team formation.

Applying these modifications provides moderate improvement on the basic

ANT reward. In addition, we provided modified versions of the ANT reward that combined the two mentioned theories, early stopping and team structures, which provided further improvements in both the learning speed and the quality of the solution reached. We also provided theoretical results showing that utilization of this modified difference function significantly improves the learning speed of the agents. This increase in speed is based on the agent receiving a counterfactual reward that estimates the reward an agent would have received had it taken a particular action.

The use of team rewards proved to be beneficial for learning. However, it evaluates the extra information received from team members in the same way. We present a last modification to the ANT reward in which the reward received from team members varies based on the number of team members performing the same action. The improvements due to the proposed method are significantly more pronounced in the presence of a large number of agents.

In multiagent learning, particularly in dynamic environments, each agents action indirectly affects the reward and indeed the behavior of other agents. One can think of the existence of other agents as the internal (or external) actuator that results in a changing environment. In this thesis, we explore the new condition where, beside the fact that all the agents are in some way changing the environment, the environment itself is changing. We show that the performance improvements are significantly more pronounced in dynamic environments where the conditions change either randomly or with high periodicity. In both cases, the rapid learning allows the agents to track a highly dynamic environment.

7.2 Future Work

Though the result of applying the ANT reward is encouraging, there are multiple areas for future exploration within this domain. For example, additional improvement may be obtained by investigating different methods where we consider other agent's rewards within the system. Specifically, investigation of the effect of utilizing an estimated, rather than directly calculated, difference reward function. Moreover, modifying the way in which agents estimate their ANT rewards can lead to substantial computational gains in addition to the already achieved increase in speed in the number of iterations required for convergence.

In addition, communication and observation requirements of the agents can be explicitly explored and connected to system performance. Techniques which provide agents with more intelligent reaction to changing of environment as well as more suitably choosing the team members can be beneficial. In such a method, agents can even take advantage of dynamic selection of other agents.

Finally, having agents adopt particular roles within a team can potentially provide further improvement in the learning speed. Defining an innovative relationship between team members or even distinctive teams with the intention of being more aligned with a system objective may lead to significant progress in the learning process.

Efficient learning in multiagent systems is an intricate and challenging problem. As this thesis suggests, there are successful methods for addressing the challenges in order to improve the system performance in addition to the speed of learning,

as are there many areas for further research.

Bibliography

[1] A. Agogino and K. Tumer. Unifying temporal and structural credit assignment problems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, New York, NY, July 2004.

[2] A. Agogino and K. Tumer. Distributed evaluation functions for fault tolerant multi rover systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Seattle, WA, July 2006.

[3] A. K. Agogino and K. Tumer. Handling communication restrictions and team formation in congestion games. *Journal of Autonomous Agents and Multi Agent Systems*, 13(1):97–115, 2006.

[4] A. K. Agogino and K. Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic environments. *Journal of Autonomous Agents and Multi Agent Systems*, 17(2):320–338, 2008.

[5] R. Akella and P.R. Kumar. Optimal control of production rate in a failure-prone manufacturing systems. In *IEEE Transactions on Automatic Control*, pages 116–126, 1986.

[6] E. Altman and N. Shimkin. Individual equilibrium and learning in processor sharing systems. In *Operations Research*, pages 776–784, 1998.

[7] S. Arai, K. Sycara, and T. Payne. Multi-agent reinforcement learning for planning and scheduling multiple goals. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, pages 359–360, July 2000.

[8] W. B. Arthur. Complexity in economic theory: Inductive reasoning and bounded rationality. *The American Economic Review*, 84(2):406–411, May 1994.

[9] A. L. Bazzan, J. Wahle, and F. Klügl. Agents in traffic modelling – from reactive to social behaviour. In *KI – Kunstliche Intelligenz*, pages 303–306, 1999.

[10] Givan R. Immerman N. Bernstein, D. S. and S. Zilberstein. The complexity of decentralized control of markov decision processes. In *Mathematics of Operations Research*, pages 819–840, 2002.

[11] D. P. Bertsekas. Dynamic programming and optimal control. In *Athena Scientic*, 2000.

[12] D. P. Bertsekas and J. N. Tsitsiklis. Neuro-dynamic programming. In *Athena Scientic*, 1996.

[13] C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*, Holland, 1996.

[14] J. A. Boyan and M. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems - 6*, pages 671–678. Morgan Kaufman, 1994.

[15] S. Hanks C. Boutilier, T. Dean. Decision-theoretic planning:structural assumptions and computational leverage. In *Journal of Articial Intelligence Research*, pages 1–94, 1999.

[16] R. Parr S. Venkataraman C. Guestrin, D. Koller. Efficient solution algorithms for factored mdps. In *Journal of Articial Intelligence Research*, pages 399–468, 2003.

[17] G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: A bayesian approach. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, pages 709–716, Melbourne, Australia, July 2003.

[18] D. Challet and Y. C. Zhang. On the minority game: Analytical and numerical studies. *Physica A*, 256:514–532, 1998.

[19] C. Claus and C. Boutilier. The dynamics of reinforcement learning cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752, Madison, WI, June 1998.

[20] B. Clement and E. Durfee. Theory for coordinating concurrent hierarchical planning agents. In *Proceedings of the National Conference on Artificial Intelligence*, pages 495–502, 1999.

[21] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems - 8*, pages 1017–1023. MIT Press, 1996.

[22] A. T. Campbell C.Y. Wan, S. B. Eisenman. Coda: congestion detection and avoidance in sensor networks. In *Conference On Embedded Networked Sensor Systems*, pages 05–07, Nov. 2003.

[23] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. In *Journal of Computational Intelligence*, pages 142–150, 1989.

[24] T. G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence*, 13:227–303, 2000.

[25] Jennings N. R. Dutta, P. S. and L. Moreau. Cooperative information sharing to improve distributed learning in multi-agent systems. In *Journal of Articial Intelligence Research*, pages 407–463, 2005.

[26] R. El Azouzi T. Jimenez E. Altman, T. Boulogne and L. Wynter. A survey on networking games in telecommunications. In *Computers and Operations Research*, pages 286–311, 2005.

[27] S. G. Ritchie F. Logi. A multi-agent architecture for cooperative inter-jurisdictional traffic congestion management. In *Transportation Research Part C: Emerging Technologies*, pages 507–527, Elsevier, 2002.

[28] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. In *IEEE Trans. on Automatic Control*, Sep. 2004.

[29] G. Fortino G. D. Fatta. A customizable multiagent system for distributed data mining. In *Symposium on Applied Computing*, pages 42–47, NY, USA, 2007.

[30] S. B. Gershwin. *Manufacturing Systems Engineering.* Prentice-Hall, May 1994.

[31] C. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. In *Journal of Articial Intelligence Research*, pages 143–174, Computer Science Department, Stanford University, 2004.

[32] C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *n Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 137–144, ACM Press, New York, NY, USA, 2003.

[33] C. Guestrin. *Planning under uncertainty in complex structured environments*. PhD thesis, Stanford University, 2003.

[34] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning*, pages 227–234, 2002.

[35] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250, June 1998.

[36] J. Hu and M. P. Wellman. Online learning about other agents in a dynamic multiagent system. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 239–246, May 1998.

[37] F.B. Bastani I.L. Yen. Robust coordination in distributed multi-server systems. In *Advances in Parallel and Distributed Systems*, pages 133–138, Oct 1993.

[38] P. Jefferies, M. L. Hart, and N. F. Johnson. Deterministic dynamics in the minority game. *Physical Review E*, 65 (016105), 2002.

[39] N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1:7–38, 1998.

[40] Giles C.L Jim, K. How communication can improve the performance of multiagent systems. In *5th International Conference on Autonomous Agents*, 2001.

[41] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[42] N. Khani and K. Tumer. Fast multiagent learning: Cashing in on team knowledge. In *Artificial Neural Networks in Engineering*, pages 3–10, St. Louis, 2008. ASME. **1st runner up for best theoretical paper award**.

[43] S. Kirkpatrick, C. D. Jr Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

[44] Asada M. Kuniyoshi Y. Noda I. Kitano, H. and E. Osawa. Robocup: The robot world cup initiative. In *In Proceedings of the IJCAI-95 Workshop on En-tertainment and AI/Alife*, 1995.

[45] F. Klügl, A. Bazzan, and S. Ossowski, editors. *Applications of Agent Technology in Traffic and Transportation*. Springer, 2005.

[46] M. Klusch. In *Intelligent information agents: agent-based information discovery and management on the Internet*, Springer-Verlag, New York, 1999.

[47] Spaan M. T. J. Kok, J. R. and N. Vlassis. Non-communicative multi-robot coordination in dynamic environments. In *Robotics and Autonomous Systems*, pages 99–114, 2005.

[48] P.R. Kumar. Re-entrant lines. queueing systems: Theory and applications. In *IIE Transactions*, May 1993.

[49] R. J. La and V. Anantharam. Optimal routing control: Game theoretic approach. *submitted to IEEE transactions on Automatic Control*, 10(2):272–286, April 1999.

[50] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Machine Learning Conference*, pages 535–542. Morgan Kauffman, 2000.

[51] C.T. Lawson. Citizen participation: freight community-style. In *Journal of the Transportation Research Forum*, pages 131–152, 2003.

[52] Ortiz C. Lesser, V. and M. Tambe. Distributed sensor nets: A multiagent perspective. In *Advances in Neural Information Processing Systems (NIPS)*, Kluwer academic publishers, 2003.

[53] M. L. Littman. Value-function reinforcement learning in markov games. In *Journal of Cognitive Systems Research*, pages 55–66.

[54] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.

[55] M. L. Littman and C. Szepesvari. A generalized reinforcement learning model: convergence and applications. In *in Proceedings of the Thirteenth International Conference on ing*, pages 310–318, Bari, Italy, July 1996.

[56] B. Raney M. Balmer, K. Nagel. Large-scale multi-agent simulations for transportation applications. In *Journal of Intelligent Transportation Systems*, 2004.

[57] H. Chen M. Huang D. Hendriawan M. Chau, D. Zeng. Design and evaluation of a multiagent collaborative web mining system. In *Decision support systems*, pages 167–183, April 2003.

[58] M. J. Mataric. Coordination and learning in multi-robot systems. In *IEEE Intelligent Systems*, pages 6–8, March 1998.

[59] Mary McGlohon and Sandip Sen. Learning to cooperate in multi-agent systems by combining Q-learning and evolutionary strategy. *International Journal on Lateral Computing*, 1(2):58–64, 2005.

[60] Shen W.-M. Tambe M. Modi, P. J. and M Yokoo. Asynchronous distributed constraint optimization with quality guarantees. In *Articial Intelligence*, pages 149–180, 2005.

[61] R.A. Murphey. Target-based weapon target assignment problems. In *Nonlinear Assignment Problems: Algorithms and Applications*, pages 39–53, Kluwer Academic Publishers, 1999.

[62] Kim H. J. Jordan M. Ng, A. Y. and S. Sastry. Autonomous helicopter flight via reinforcement learning. In *In Advances in Neural Information Processing Systems (NIPS)*, 2004.

[63] A. Orda, R. Rom, and N. Shimkin. Competitive routing in multiuse communication networks. *IEEE/ACM Transactions on Networking*, 1(5):510–521, 1993.

[64] Liviu Panait, Karl Tuyls, and Sean Luke. Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. *Journal of Machine Learning Research*, 9:423–457, 2008.

[65] D. Parkes. On learnable mechanism design. In *Collectives and the Design of Complex Systems*. Springer, 2004.

[66] M. L. Puterman. Markov decision processes: Discrete stochastic dynamic programming. In *In Advances in Neural Information Processing Systems (NIPS)*, Wiley, New York, 1994.

[67] D. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[68] M. Rovatsos R F. Fischer and G. Weiss. Hierarchical reinforcement learning in communication-mediated multiagent coordination. In *in Proceedings 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1334–1335, New York, US, August 2004.

[69] K. Jha R. K. Ahuja, A. Kumar and J. B. Orlin. Exact and heuristic methods for the weapon-target assignment problem. In *Technical Report, MIT, Sloan School of Management Working Paper*, 2003.

[70] M. A. Goodrich R. W. Beard, T. W. McLain and E. P. Anderson. Coordinated target assignment and intercept for unmanned air vehicles. In *IEEE Transactions on Robotics and Automation*, pages 911–922, 2002.

[71] T. Roughgarden. *Selsh Routing and the Price of Anarchy*. MIT Press, 2005.

[72] N. Yang S. Chen. Congestion avoidance based on lightweight buffer management in sensor networks. In *Parallel and Distributed Systems, IEEE Transactions*, pages 934–946, Sept. 2006.

[73] I. Kroo S. R. Bieniawski and D. Wolpert. Flight control with distributed effectors. In *AIAA Guidance, Navigation, and Control Conference*, San Francisco, CA, August 15-18, 2005.

[74] A. Sherstov and P. Stone. Three automated stock-trading agents: A comparative study. In *Agent Mediated Electronic Commerce VI: Theories for and Engineering of Distributed Mechanisms and Systems (AMEC 2004), Lecture Notes in Artificial Intelligence*, pages 173–187. Springer Verlag, Berlin, 2005.

[75] P. Stone. *Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer*. MIT Press, Cambridge, MA, 2000.

[76] P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 2005.

[77] T. Sugawara and V. Lesser. On-line learning of coordinated plans. In *In-Working Papers of the 12th International Workshop on Distributed Articial Intelligence*, pages 335–355, 1993.

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

[79] K. Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, 1998.

[80] C. Szepesvari and M. L. Littman. A unified analysis of value-function-based reinforcement learning algorithms. In *Neural Computing*, pages 8–11, 1999.

[81] M. Hallenbeck C. Boon R. Margiotta T. Lomax, S. Turner. Traffic congestion and travel reliability. September 2001.

[82] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[83] Matthew E. Taylor, Shimon Whiteson, and Peter Stone. Comparing evolutionary and temporal difference methods for reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1321–1328, Seattle, WA, July 2006. **Best Paper Award**.

[84] G. Tesauro. Practical issues in temporal difference learning. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems - 4*, pages 259–266. Morgan Kaufmann, 1992.

[85] G. Tesauro. Temporal difference learning and td-gammon. In *Communications of the ACM*, pages 58–67, March 1995.

[86] K. Tumer and A. Agogino. Robust coordination of a large set of simple rovers. In *Proceedings of the IEEE Aerospace Conference*, Big Sky, MO, March 2006.

[87] K. Tumer and A. Agogino. Distributed agent-based air traffic flow management. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 330–337, Honolulu,HI, May 2007. **Best Paper Award**.

[88] K. Tumer, Z. T. Welch, and A. Agogino. Aligning social welfare and agent preferences to alleviate traffic congestion. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Estoril,Portugal, May 2008.

[89] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.

[90] Karl Tuyls and Simon Parsons. What evolutionary game theory tells us about multiagent learning. *Artificial Inelligence*, 171(7):406–416, 2007.

[91] Katja Verbeeck, Ann Nowe, and Karl Tuyls. Coordinated exploration in multi-agent reinforcement learning: An application to load balancing. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Utrecht, The Netherlands, 2005.

[92] Katja Verbeeck, Maarten Peeters, Ann Nowe, and Karl Tuyls. Reinforcement learning in stochastic single and multi-stage games. In *Adaptive Agents and Multi-Agent Systems II, Lecture Notes in Artificial Intelligence*, pages 275–294. Springer Verlag, Berlin, 2005.

[93] J. M. Vidal and E. H. Durfee. The moving target function problem in multi-agent learning. In *Proceedings of the Third International Conference on Multi-Agent Systems*, pages 317–324. AAAI/MIT press, July 1998.

[94] Jose M. Vidal. Multiagent coordination using a distributed combinatorial auction. In *AAAI Workshop on Auction Mechanism for Robot Coordination*, July 2006.

[95] N. Vlassis. A concise introduction to multiagent systems and distributed ai. In *Informatics Institute, University of Amsterdam*, 2003.

[96] E.M. Atkins W. Ren, R.W. Beard. A survey of consensus problems in multi-agent coordination. In *American Control Conference*, pages 1859– 1864, June 2005.

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

[98] C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, Cambridge University, 1989.

[99] G. Weiss. Multiagent systems: A modern approach to distributed articial intelligence. In *MIT Press*, 1999.

[100] M. P. Wellman, S.-F. Cheng, D. M. Reeves, and K. M. Lochne. Trading agents competing: Performance, progress, and market effectiveness. *IEEE Intelligent Systems*, 18(6):48–53, November/December 2003.

[101] Shimon Whiteson, Matthew E. Taylor, and Peter Stone. Empirical studies in action selection for reinforcement learning. *Adaptive Behavior*, 15(1), 2007.

[102] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.