Coordinating actions in congestion games: Impact of top-down and bottom-up utilities[*]

Kagan Tumer
Oregon State University
kagan.tumer@oregonstate.edu

Scott Proper
Oregon State University
proper@eecs.oregonstate.edu

Abstract

Congestion games offer a perfect environment in which to study the impact of local decisions on global utilities in multiagent systems. What is particularly interesting in such problems is that no individual action is intrinsically "good" or "bad" but that combinations of actions lead to desirable or undesirable outcomes. As a consequence, agents need to learn how to coordinate their actions with those of other agents, rather than learn a particular set of "good" actions. A congestion game can be studied from two different perspectives: (i) from the top down, where a global utility (e.g., a system-centric view of congestion) specifies the task to be achieved; or (ii) from the bottom up, where each agent has its own intrinsic utility it wants to maximize. In many cases, these two approaches are at odds with one another, where agents aiming to maximize their intrinsic utilities lead to poor values of a system level utility. In this paper we extend results on difference utilities, a form of shaped utility that enables multiagent learning in congested, noisy conditions, to study the global behavior that arises from the agents' choices in two types of congestion games. Our key result is that agents that aim to maximize a modified version of their own intrinsic utilities not only perform well in terms of the global utility, but also, on average perform better with respect to their own original utilities. In addition, we show that difference utilities are robust to agents "defecting" and using their own intrinsic utilities, and that performance degrades gracefully with the number of defectors.

Keywords: Multiagent, Reinforcement learning, Coordination, Congestion games.

1 Introduction

Congestion games are characterized by having the system performance depend on the number of agents that select a particular action, rather than on the intrinsic value of particular actions. Examples of such problems include many traffic problems (like lane/route selection) [9, 13, 14, 30, 34], path selection in data routing [32], and side selection in the minority game [15, 28]. In those problems, the desirability of lanes, paths or sides depends solely on the number of agents having selected them. As a consequence, agents (or players) that take independent actions need to coordinate their actions in order to achieve their own goals.

In this paper we apply multiagent coordination to two distinct formulations of a congestion game (Figure 1). First we investigate this problem from "system-centric" view with a well defined system utility function measuring the performance of the full system. In this problem, different actions have different values based on their potential use and their impact on the

[*]Appears in Autonomous Agents and Multi-Agent Systems, 27(3), 419-443. DOIi:10.1007/s10458-012-9211-z.

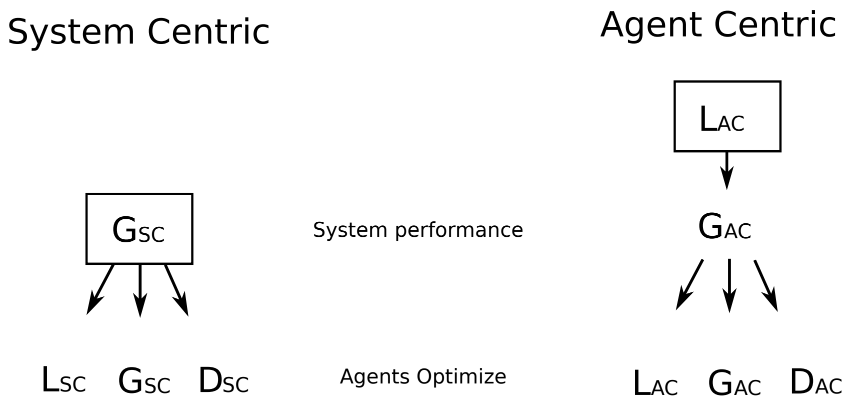


Figure 1: The two different views of systems in congestion games. The system-centric view (left) starts with a global utility function provided for the task. Agents then receive utility functions directly derived from this utility. The agent-centric view (right) starts from the agents' intrinsic utilities, then builds a global performance metric from them to measure the entire system. From that global utility we derive utility functions for the agents to pursue (this is more suited to competitive domains). In each case, we consider three utility types: "L" for local utility, "G" for global or system utility, and "D" for difference utility (Section 3.2).

full system. In this case, the agents need to learn to select actions to not produce "spikes" for certain actions, with deleterious effects on the system utility. Then we investigate this problem from the perspective of the agents who want to maximize their own intrinsic utility functions, which better accommodates a competitive setting such as congestion games. Based on those agent-centric functions, we build a social welfare function to measure the system performance. Both formulations share the same underlying property that agents greedily pursuing their best interests cause congestion to worsen for everyone in the system, including themselves. However, the solution to alleviating congestion takes on a different form in each formulation, as the interaction between agent-centric and system-centric utilities have different characteristics in their respective formulations.

In this setting, we aim to study the system behavior that results from the agents aiming to maximize the utilities they are given. One direct approach to this problem would be to enumerate the agents' actions and compute the equilibrium point that ensues. However, this is impractical not only because of the sheer number of agents and actions, but also because of the information needed by each agent [27]. Another approach would be to have each agent use a randomized policy based on a set of actions determined to optimize their utilities (like a "softmax" approach as described in [43]). However, this approach requires the agents to know the utility value for each of their actions. Because in this setting the resulting system utility depends on the agents' joint action and because the agents do not know what actions were taken by the other agents in the system, they operate with (very) limited information. The only feedback they receive is their own utility values. Therefore, to test the efficiency of each agent utility, we focus on having the agents learn the set of actions that optimize their utilities. This is similar to agents playing a non-stationary "n-armed" bandit game [43] where the distribution of each arm depend on the actions of other agents.

In this work, each agent aims to perform an "action-to-utility" mapping based on a simple reinforcement learning algorithm [33, 43]. Because the focus of this work is the impact of the utilities themselves and the learning algorithm is simply a tool to maximize those utilities, we focus on simple learning algorithms. Indeed, the key issue here is to ensure that the agents have utilities that promote good system level behavior. To that end, it is imperative that the agent utilities: (i) be aligned with the system utility[1], ensuring that when agents aim to maximize their own utility they also aim to maximize system utility; and (ii) be sensitive to the actions of the agent taking that action, so that each agent can determine the proper actions to select (i.e., they need to limit the impact of other agents in the utility functions of a particular agent).

The difficulty in agent utility selection stems from the fact that typically these two properties provide conflicting requirements. A utility that is aligned with the system utility usually accounts for the actions of other agents, and thus is likely to not be sensitive to the actions of one agent; on the other hand, a utility that is sensitive to the actions of one agent is likely not to be aligned with system utility. This issue is central to achieving coordination in a congestion game and has been investigated in various fields such as computational economics, mechanism design, computational ecologies and game theory [10, 29, 41, 26, 38, 42, 49]. In particular, the difference utility, which aims to directly estimate the impact of an agent on the global utility has been shown to provide coordinated behavior in multiple domains where a well-defined system level utility function was provided, including multi-robot coordination and air traffic control [1, 2, 3, 46, 49, 52]. In this work, we extend this concept to domains defined by agent utilities.

We experiment with two congestion domains in this paper: a "single-peak" domain with few actions, only one of which is "desired" or targeted by agents [48, 47]; and a "multi-peak" domain with many actions, several of which may be desired by agents. These domains correspond to various kinds of real-life problems; for example the single-peak domain loosely corresponds to a rush-hour traffic problem in which many drivers desire to arrive at home immediately after leaving work at 5 PM. The multi-peak congestion domain loosely corresponds to an airport scheduling problem with many airplanes wishing to leave the airport at several different peak times. The key differences between these problems is that in the first one there are few actions with large capacities (e.g., a handful of 15-minute windows to depart but the road can take many cars in that time slice) whereas in the second problem there are many actions with small capacity (e.g., many departure times for aircraft, but only a few can depart at any one time).

1.1 Contributions

In this paper, we build the agent utilities around the concept of the difference utility extensively used in problems defined on the left side of Figure 1 [1, 2, 3, 46, 49, 52].

This paper has three key contributions, all based on the right hand formulation of Figure 1:

1. We demonstrate that the difference utility can be derived even if the problem formulation does not have a system level utility.

2. We then show that agents using the difference utility not only perform well in terms of a global metric, but *on average* perform better on their own intrinsic utilities. From Figure 1, this means that agents pursuing D_{AC} achieves higher values of L_{AC} than if they were aiming to directly maximize L_{AC}.

3. We then show that agents using the difference utility derived from an agent centric setting also perform well in terms of a system-centric utility. From Figure 1, this means that

[1]We call the function rating the performance of the full system, "system utility" throughout this paper. We will specify "system-centric utility" or "agent-centric utility" to distinguish between the two main system performance criteria.

agents pursuing D_{AC} achieve good values of G_{SC}, whereas agents pursuing L_{AC} or G_{AC} do not.

This paper is organized as follows: Section 2 presents the system dynamics and mathematical congestion model used in this paper. Section 3 discusses the agent learning and the agent utilities that will be explored, including the difference utility. Section 4 presents a system viewed from top-down from a system-centric perspective, starting from a well defined system utility and using that to derive agent utilities. Section 5 presents the system from a bottom-up, agent-centric perspective, with each agent starting from an intrinsic utility, and the system building a social welfare function to rate the performance of the full system. Section 6 presents experimental results and shows that not only do difference utilities provide good values for the system-centric utility, but having agents use a modified version of their intrinsic utilities rather than aim to maximize them directly leads them to achieve higher values of their own intrinsic utilities. Finally Section 7 discusses the implication of these results and provides insight into the reasons for some of the behavior observed in these experiments.

1.2 Related Work

Congestion games [40] – where system performance depends on the number of agents taking a particular action – are central to work in AI, game theory, operations research, and economics. The congestion game studied in this paper is similar to the problem experienced by drivers choosing a time at which to leave work. Similar traffic problems are well-studied in previous literature and provide an interesting domain to study the behavior of cooperative multiagent systems. This type of congestion game is ubiquitous in routing domains (e.g., on a highway, a particular lane is not preferable to any other lane, what matters is how many others are using it) [31, 48]. Agents in such a system often require sophisticated coordination in order to avoid a "tragedy of the commons" where selfish agents reduce the reward gained by everyone, including themselves. An example of such an approach is reservation based intersection management where agents are cars [18]. Larger, strategic solutions to such problems have also been explored [9]. The impact of reinforcement learning agents (at the level of individual traffic lights) has been studied [51, 35].

The social impact of local decisions has also been studied as variants of the El Farol bar problem (also called the minority game) which can be viewed as a one shot congestion game [16, 15, 52]. The multi-night bar problem is an abstraction of congestion games which has been extensively studied [4, 28]. In such versions of the congestion game, each agent has to determine which day of 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, which requires agents to remain in constant coordination. The congestion game we study in this work is similar, but due to the domain and shape of the utility functions, only high attendance is a problem (too few agents selecting an action is not an issue), but that problem is compounded by the cascading congestion in other slots.

Typical approaches to solving congestion games have often studied the extremely challenging [36] problem of finding a Nash equilibrium for general games [17, ?]. In general, equilibria are hard to find and there may be an exponential number of them, particularly when the number of agents and the number of agent actions increases. Indeed, describing strategic interactions in normal form requires space exponential in the number of players, and is thus difficult even without computational issues [27].

In addition, direct computation of incentives to lead to particular solutions can provide solutions to congestion games. For example, the Vickrey-Clarke-Groves (VCG) mechanism [19] can provide a solution and has been used in congestion games [39]. However, the computation of the equilibrium points, allocations and utilities can be difficult in larger problems. An extension dubbed "learnable mechanism design" aims to overcome this problem and has been successful in some domains [37].

In cases where direct computation of equilibrium and agent actions is not practical, reinforcement learning has been used to coordinate multiagent systems. For example, Win or Learn Fast [11] is one such approach that learns faster when losing, and more slowly when winning, which allows agents to converge to good policies in multiagent games. Alternately, agents may rely on knowledge about the actions of other agents [25], or require communication between the agents [44]. Computing equilibrium points is of course critical in such problems and efficient methods for their computation have been proposed [12]. In addition, N-armed bandit problems have been explored by reinforcement learning to study the convergence of agents [43]. The congestion game studied in this work can be viewed as a variant of such problems where the distribution of each "arm" depends on the actions of other agents in the system.

In general, coordinating large numbers of agents using a learning approach has been shown not to perform well in some domains, particularly when agents may work at cross-purposes [45]. In fact, the congestion domains we study in this paper are instances of such problems in that without external mechanisms, agents pursuing their own utilities lead to poor system behavior. One approach to overcome this is the use of agent specific utilities such as difference utilities that aim to capture the impact of an agent on the full system [1, 48, 50, 52]. Such utilities have been applied to a multitude of domains such as multi-robot coordination, air traffic control and sensor coordination, all domains that had a well defined system level utility (as described on the left-hand side of Figure 1) [3, 46, 47], and have not been extended to problem defined by the agent's intrinsic utilities (left hand side of Figure 1).

2 System Dynamics and Congestion Model

In this work, we investigate a congestion model where there is a fixed set of agents whose actions are to select one or more desired resources, referred to in this paper as "slots" or "time slots". Rather than model traffic directly [5, 8, 7, 20, 22, 23, 24, 34], we aim to capture "the agent action to utility" mapping using an abstract model of congestion, but still retain the complexity of traffic flow.

This model abstracts many different real-world congestion games, such as drivers competing for the use of a particular stretch of road or a particular lane of a highway, patrons visiting a bar on different days, airplanes competing to depart an airport at particular profitable times, or even packets on a network competing for particular routes [6]. If the number of agents is below the capacity of the slot, the agents reach their destination at the expected times. However, if the number of agents exceeds the slots capacity, the agents remain in the system and effectively reduce its capacity for all following agents. In this paper, we use a nonlinear model for the agents "cascade" from one time slot to another, which introduces both a time-dependent aspect and a nonlinear congestion increase on to the more traditional bar problem (minority game) [4, 16, 15, 28, 52].

Let the number of agents k_{a_i}, in slot a_i be given by:

$$k_{a_i} = k_{a_i - 1} - exit_{a_i}(k_{a_i - 1}) + \sum_j I(a_i, j) \ , \tag{1}$$

where k_{a_i-1} is the number of agents in the preceding slot $(a_i - 1)$ and $I(a_i, j)$ is the indicator function that returns one if agent j selected slot a_i and zero otherwise.

Given a system parameter c that provides the capacity of a resource, Equation 2 provides the number of agents leaving the system at time slot i:

$$exit(k) = \begin{cases} k & \text{if } k \leq c \\ ce^{-\frac{k-c}{c}} & \text{otherwise} \end{cases} \tag{2}$$

This equation states that up to c agents may leave the system at slot i, and the number of agents that may leave decreases exponentially once that capacity has been reached. The agents that do not leave remain in the system. The net effect of this congestion model is in decreasing the effective capacity of a time slot based on the number of agents who selected that slot. Congestion is modeled as a queue. New agents enter the back of the queue. Agents leaving the system are taken from the front of the queue. Thus, the actions of an agent have influence over multiple time slots, starting with the time slot in which they entered the system.

For example, if 300 agents are on a given time slot with capacity 250, the capacity is reduced by 18%, resulting in an effective capacity of 205 and causing 95 agents to be cascaded to the next time slot. In contrast, in a linear model, only 50 agents would have been cascaded to the next time slot. This effect becomes more pronounced as congestion increases (for example for 500 agents in the system, the capacity is reduced by 63%, to an effective capacity of only 91 and 409 agents are cascaded to the next time slot).

3 Agent Coordination

In this paper, the joint agent's environment is modeled as a finite set of actions $a \in A$ available to the agent [2] and reward model $R : A \rightarrow \mathbb{R}$ which returns the reward $R(a)$ after taking action a. We will be referring to rewards as "utilities for the rest of the paper.

Each agent i, tries to maximize its own utility function $g_i(a)$, and the system performance is measured by a system utility function $G(a)$ (for example, this is the "system-centric utility" discussed in Section 4 or the social welfare function developed in Section 5). In addition, to isolate the action of agent i when necessary, we use the notation a_{-i} to refer to the situation where agent i's action has been removed.

3.1 Agent Learning

The focus of this paper is studying the impact of the utility functions that promote desired system behavior, assuming the agents have the capabilities to maximize their utilities. To that end, each agent will use a simple reinforcement learning algorithm (though alternatives such as evolving neuro-controllers are also effective [3]).

For complex delayed-reward problems, relatively sophisticated reinforcement learning systems such as temporal difference may have to be used. The congestion domain modeled in

[2]In past work, we referred to this as "full system state" z. In this paper we use a to emphasize the dependence of the utilities on the agents' joint action.

this paper only needs to utilize immediate utilities, and thus we will not use sophisticated, delayed-reward learning algorithms such as temporal difference learning. Instead, each agent will use a value table that will provide estimates for the utility values of specific actions.

In addition, to allow exploration of the action space, the agents will select the action deemed best at that point with probability $1 - \epsilon$ and take a random action with probability ϵ. This reinforcement learner can be described as an ϵ-greedy learner with a discount parameter of 0. At every episode an agent takes an action and then receives a reward (value of the immediate utility) evaluating that action. After taking action a and receiving reward R an agent updates its value table as follows:

$$V(a) \leftarrow (1 - \alpha)V(a) + \alpha \cdot R \ , \tag{3}$$

where α is the learning rate. In the experiments described in this paper, α is equal to 0.5 and ϵ is equal to 0.05. The parameters were chosen experimentally, though system performance was not overly sensitive to these parameters.

3.2 Difference Utility Functions

A key question now is to determine what immediate reward each agent will receive for their actions, and how the selection of that reward impacts system level behavior. In addition to an agent's intrinsic utility (assuming it exists) and the system utility (G) in this work we will explore the **difference** utility [49] for shaping agent behavior. For a given system level function G, the difference utility for an agent i is given by:

$$D_i(a) \equiv G(a) - G(a_{-i}) \ , \tag{4}$$

where a_{-i} is the joint action vector without agent i. In other words, the action of i is removed (i.e., replaced with a domain specific "null" vector). This type of agent utility offers two advantages. First, D_i and G are aligned, meaning that any action that improves D_i also improves G. This is because the two utilities have the same partial derivative with respect to the actions of agent i, as the second term of D_i does not depend on the actions of that agent [49]. Second, D_i usually has far better signal-to-noise properties than does G, because the second term of D removes some of the effect of other agents (i.e., noise) from i's utility function.

The difference utility can be applied to any linear or non-linear system utility function. However, its effectiveness is dependent on the domain and the interaction among the agent utility functions. At best, it fully cancels the effect of all other agents. At worst, it reduces to the system utility function, unable to remove any terms. Still, the difference utility often requires less computation than the system utility function [50]. Indeed, for the problems presented in this paper, agent i can compute D_i using less information than required for G (see details in Section 4 and Section 5).

4 Measuring System Performance Top Down

The first step in this investigation is determining well defined performance metrics that rate the different outcomes resulting from the agents' actions. Though for some domains, there may be a single logical utility function (e.g., robots maximizing an exploration based utility), in the congestion domain, this step is far from straight-forward. In this study, we will focus on two distinct perspectives and provide results and insight for both.

First, one can use a *top-down* approach where a "system-centric utility" rates the performance of the system from an average congestion perspective, with little to no regard for individual agents' preferences. Second, one can use a *bottom-up* approach where the agents' intrinsic preferences are used to build a social welfare function and rate the performance of the system in terms of that social welfare function, directly reflecting how satisfied the agents are with the outcomes. Though one may reasonably expect that the system-centric utility and the social welfare function based on agent preferences will aim to lower congestion, there is no reason to assume that they will have the same optima, or promote similar behavior. Let us first discuss the case where the system performance is measured by a global utility representing the system perspective. This *system-centric utility* is given by:

$$G_{SC}(a) = \sum_{a_i} w_{a_i} exit(k_{a_i}) \ , \tag{5}$$

where w_{a_i} are weights that model scenarios where different time slots a_i have different desirabilities, and $exit(k_{a_i})$ measures the number of agents that exit the system in time slot a_i.

In this problem formulation, the task is to have the agents choose time slots that maximize G_{SC}, the system-centric utility. To that end, agents have to balance the benefit of going at preferred time slots against possible congestion in those time slots.

While the goal is to maximize G_{SC}, the distributed nature of the problem means that each individual agent will try to maximize an agent-specific utility. The agents will maximize their utilities through reinforcement learning (as discussed in Section 3.1).

A key decision then is in determining the type of utility that each agent will receive. In this formulation, we focus on the following three agent utilities:

- The first utility is simply the system utility G_{SC}, where each agent tries to maximize the system utility directly.

- The second utility is a local utility, L_{SC_i} where each agent tries to maximize a utility based on the time slot it selected:

  $$L_{SC_i}(a) = w_{a_i} exit(k_{a_i}) \ , \tag{6}$$

  where k_{a_i} is the number of agents in the time slot chosen by agent i.

- The third utility is the difference utility, D_{SC}:

  $$\begin{aligned} D_{SC_i}(a) &= G_{SC}(a) - G_{SC}(a_{-i}) \\ &= \sum_j w_j exit(k_j) - \sum_j w_j exit(k_{-i_j}) \ , \end{aligned} \tag{7}$$

  where k_{-i_j} is the number of agents there would have been in time slot j had agent i not been in the system (for the slots in which the agent is in the system, this is $k_j - 1$).

  The computation of Equation (7) is difficult due to the requirement to receive information from every time slot. Restricting this equation to between the time slots when an agent enters and exits the system captures the key contribution of an agent, with two simple assumptions: (i) what happened before agent i took its action has no impact on agent i's utility; and (ii) what happens after agent i arrives has no impact on agent i's utility. The first assumption always holds, and the second one is a good approximation as the impact of the two terms of Equation (7) are very similar for when the actions of agent i have been removed (see Section 7 for further examination of this assumption). With these assumptions, we can use the following approximation:

$$D_{SC_i}(a) \simeq \sum_{j=a_i}^{\tau_i} w_j exit(k_j) - \sum_{j=a_i}^{\tau_i} w_j exit(k_j - 1) \ , \tag{8}$$

where a_i is the selected slot and τ_i is the slot in which the agent exited the system.

5 Measuring System Performance Bottom Up

In the discussion above, the system-centric utility measured the health of the whole system, providing a global perspective from which to derive utilities for the individual agents. Though appealing in its formulation, this approach is less representative of congestion than a model in which the agents' intrinsic preferences are the key factors in shaping system behavior. In this section, we focus on such an agent-centric approach, and introduce individual agent preferences that directly capture the desires of the agents.

In this approach, the system performance is gauged by a social welfare function based on the agent preferences. With this function the individual agent utilities measure the agents' success at attaining personal arrival preference. We selected an exponential decay function to represent the agents' satisfaction with their time slots, resulting in the following *agent-centric*, L_{AC_i}, functions (which again only depends on the joint action a):

$$L_{AC_i}(a) = e^{\frac{-(\tau_i - t_i)^2}{b}} \ , \tag{9}$$

where τ_i is time in which agent i leaves the system, t_i is the desired or target slot for leaving the system for agent i, and b is a parameter that determines the steepness of the decline in utility as a function of the time gap between desired and actual arrival times.

After an agent picks some slot to depart, depending on the choices of other agents, it may find itself stuck in congestion, leaving the system at a later time (τ_i) than intended. The agent actually wants to leave the system at its target time (t_i), so L_{AC_i} peaks when the agent leaves the system at its target time.

Unlike in the previous section where we started from a system wide perspective (system-centric utility), in this perspective, we start from intrinsic agent preferences. Using the agent-centric functions, we construct a social welfare function (Equation 10) and use that to measure the system performance. The interesting question we address in this section is what utilities should the agent aim to maximize to also maximize the social welfare function. In this study, we focus on the following three agent utilities:

- The agent-centric utility given by L_{AC_i}, where each agent tries to directly maximize its own local utility.

- The social welfare function based on the agent-centric functions:

$$\begin{aligned} G_{AC}(a) &= \sum_i L_{AC_i}(a) \\ &= \sum_i e^{\frac{-(\tau_i - t_i)^2}{b}} \end{aligned} \tag{10}$$

This function measures the performance of the system as the sum of the individual local agent utilities, so the system does best when all of its agents arrive in their desired target slots.

- The difference utility which in this formulation is derived from the social welfare function computed above is given by:

$$D_{AC_i}(a) \quad = \quad G_{AC}(a) - G_{AC}(a_{-i}) \tag{11}$$

This utility computes an agents impact on the system by estimating the gain to the system by the removal of that agent. The net effect of removing agent i from the system is in allowing other agents to potentially leave the system rather than remain in a congested system. When the time slot chosen by agent i is not congested, agent i's removal does not impact any other agents. Using this reasoning, Equation (11) provides the difference between agent i's own utility and the utility the agents who remain in the system would have received had agent i not been in the system. In terms of the original agent-centric utilities, this means that agent i's utility is penalized by the contributions of agents that it caused to be delayed.

The computation of Equation (11) is problematic in practice due to the coupling between the agents actions and their impact of future time slots. To effectively measure this value, an estimate of each time slot following the agent's selected time slot would need to be computed. A more readily computable estimate for the difference utility may be derived by substituting Equation (10) into Equation (11) as follows:

$$D_{AC_i}(a) = \sum_j e^{\frac{-(\tau_j - t_j)^2}{b}} - \sum_{j \neq i} e^{\frac{-(\tau_j^{'} - t_j)^2}{b}} \ , \tag{12}$$

where τ and $\tau^{'}$ signify the time slot that the agents leave the system with the agent i in the system, and without agent i in the system respectively.

We now make the simplifying assumption that the only significantly affected slot is the one the agent attended, a_i. By doing this, we ignore the direct impact that agent i has on following slots, and any rewards agents delayed into following slots might receive. As well-behaving agents should never be delayed, we believe this is a reasonable assumption, especially once the policy begins to converge:

$$D_{AC_i}(a) \simeq \sum_j I(\tau_j = a_i) e^{\frac{-(a_i - t_j)^2}{b}} - \sum_{j \neq i} I(\tau_j' = a_i) e^{\frac{-(a_i - t_j)^2}{b}} \ , \tag{13}$$

where $I()$ is an identity function that returns 1 if it's argument is true, 0 otherwise. Intuitively, the first term counts the number of agents in slot a_i that exit it with agent i in the system, multiplied by a weight for that slot, and the second term counts the number of agents in slot a_i that exit it with agent i removed from the system, likewise multiplied by a weight. Simplifying:

$$\begin{aligned} D_{AC_i}(a) \simeq & I(\tau_i = a_i) e^{\frac{-(a_i - t_i)^2}{b}} + \\ & \sum_{j \neq i} \left(I(\tau_j = a_i) e^{\frac{-(a_i - t_j)^2}{b}} - I(\tau_j' = a_i) e^{\frac{-(a_i - t_j)^2}{b}} \right) \\ = & I(\tau_i = a_i) e^{\frac{-(a_i - t_i)^2}{b}} + \sum_{j \neq i} \left(e^{\frac{-(a_i - t_j)^2}{b}} \left(I(\tau_j = a_i) - I(\tau_j' = a_i) \right) \right) \end{aligned} \tag{14}$$

If we enumerate the outcomes of $I(\tau_j = a_i) - I(\tau'_j = a_i)$ and remove one impossible outcome in which removing an agent from the system increases congestion, we note that $I(\tau_j = a_i) - I(\tau'_j = a_i) = -I(\tau'_j = a_i)I(\tau_j > a_i)$ and can make the corresponding substitution:

$$D_{AC_i}(a) \simeq I(\tau_i = a_i)e^{\frac{-(a_i-t_i)^2}{b}} + \sum_{j \neq i} \left(-e^{\frac{-(a_i-t_j)^2}{b}} I(\tau'_j = a_i)I(\tau_j > a_i) \right) \tag{15}$$

For the single-peak domain described in Section 6.2, all agents share the same desired slot, i.e. $t_j = t_i$. If agents do not share the same desired slot, as in the multi-peak domain in Section 6.3, we assume that agents taking the same action desire the same slot, i.e. $t_j \simeq t_i$. Thus we replace t_j with t_i and factor out $-e^{\frac{-(a_i-t_i)^2}{b}}$ from the sum:

$$D_{AC_i}(a) \simeq I(\tau_i = a_i)e^{\frac{-(a_i-t_i)^2}{b}} - e^{\frac{-(a_i-t_i)^2}{b}} \sum_{j \neq i} \left(I(\tau'_j = a_i)I(\tau_j > a_i) \right) \tag{16}$$

Intuitively, $\sum_{j \neq i} \left(I(\tau'_j = a_i)I(\tau_j > a_i) \right)$ counts the number of other agents in slot a_i that the action of agent i causes to be delayed. From the system dynamics (Equation (2)), we know this is one agent unless congestion is very high. In other words, if agent i is delayed, it will cause (at most) one other agent in the system to also be delayed. Thus we can approximate Equation (16) as follows:

$$D_{AC_i}(a) \simeq I(\tau_i = a_i)e^{\frac{-(a_i-t_i)^2}{b}} - e^{\frac{-(a_i-t_i)^2}{b}} I(\tau_i > a_i) \tag{17}$$

or:

$$D_{AC_i}(a) \simeq \begin{cases} e^{\frac{-(a_i-t_i)^2}{b}} & : \tau_i = a_i \\ -e^{\frac{-(a_i-t_i)^2}{b}} & : \tau_i > a_i \end{cases} \tag{18}$$

where the condition $\tau_i > a_i$ determines whether agent i caused any congestion. Note that Equation (18) may also be understood intuitively: the first term indicates that agent i receives its local utility directly, due to lack of congestion. The second term indicates that agent i should not only not be rewarded by its local utility, but also penalized an equal amount, which in fact compensates for the loss of the local utility of another agent (the agent which was delayed due to this agent's action).

6 Experimental Results

We perform experiments on two variations of the congestion domain. The "single-peak" domain has only 9 slots (actions) and a single "peak" or desired time for agents to schedule. The "multi-peak" domain has 89 slots or actions, and 11 evenly-scattered peak times which agents are randomly assigned to. Despite the multi-peak domain having more slots, it is equally congested as the capacity of each slot has been reduced so as to match the overall capacity of the single-peak domain.

For the single-peak congestion domain, we first discuss the implementation and then test the effectiveness of the different agent utilities in promoting desirable system behavior by performing experiments for both the top down and bottom up approaches. We then show how agents learning on the bottom-up utility leads to intriguing behavior, where agents that adopt the bottom-up utility result in better global behavior as measured by the top-down utility than agents learning on the actual top-down utility. We show an “ideal” situation where cooperative agents agree to follow a new personal utility. Finally, we then show several experiments where we either scale the number of agents, or scale the percentage of agents using difference utilities (instead of local utilities).

We follow the single-peak experiments with those for the multi-peak congestion domain. After discussing the implementation, we show results demonstrating the effectiveness of the top-down and bottom-up learning approaches.

6.1 Summary of Agent Utility Functions and Experiments

In all the results reported in the following sections, regardless of what utilities the agents aimed to maximize, *the global system performance* is measured by the appropriate system utility (system centric or social welfare function, based on how the problem was defined as described in Figure 1), and normalized to the optimal performance. Below, we summarize the terminology of the different utility functions discussed in preceding sections:

- G_{SC} : Global utility for system centric formulation measuring full system performance.
- L_{SC} : Local agent utility derived from G_{SC}, which can be viewed as a simple decomposition of G_{SC}.
- D_{SC} : Difference utility derived from G_{SC}.
- G_{AC} : Social welfare function based on L_{AC}.
- L_{AC} : Agent centric or intrinsic utilities of each agent.
- D_{AC} : Difference utility derived from G_{AC}, which itself is based on L_{AC}. Thus, D_{AC} is an indirect shaping of L_{AC}.

In the rest of the section, we perform the following experiments:

1. Single peak congestion domain
   (a) Bottom up results to show the performance of different agent utilities including agent action profiles and scaling results
   (b) Top down results to show agent performance and scaling properties.
   (c) Results measuring the impact of using bottom up derived utilities for system centric utility.
   (d) Results measuring the impact of defectors (agents that use their intrinsic utilities).
2. Multi peak congestion domain
   (a) Bottom up results to show the performance of different agent utilities.
   (b) Top down results to show agent performance.
   (c) Results measuring the impact of using bottom up derived utilities for system centric utility.

6.2 Single-Peak Congestion Domain Implementation

The single-peak congestion domain contains nine possible actions corresponding to time slots 0-8. For our experiments, each of the agents received higher rewards for being assigned to the center time slot ($a_i = 4$). However, a capacity of 250 agents/slot prevents all agents from successfully taking this action. Most experiments used 1000 agents, however we also experimented with scaling the number of agents between 400 and 1400. This domain corresponds to some real-world domains in which a single time or location is desired by all the agents in a system. For example, a rush-hour traffic domain in which all agents desire to arrive home immediately after leaving work at 5pm could be abstractly modeled as a single-peak congestion domain.

Our experiments ran for 400 episodes, or time steps as each episode only lasts for one action. We averaged our results over 30 runs. We further divided each run into 20 segments of 20 episodes each, averaging over the data points in each segment in order to plot a progression over time. In addition to plotting the mean for each run, we graphed the differences in each mean as error bars using the standard deviation of the mean/$\sqrt{n}$, where n is the number of runs. We used values of $\alpha = .5$, $\epsilon = .05$, and $b = 1.0$ (see Equation (9)) for our experiments. In all experiments in this paper, agent value functions were optimistically initialized. As our experiments compare results between many different measures of utility, we could not directly compare average utility values. Instead, we compared averages of the utility as a percentage of a near-optimal value of the global utility. This value was found by creating hand-coded spreads of assignments of agents to time slots. For example, a near-optimal assignment of 1000 agents to 9 time slots would be:

$$[0\ 0\ 125\ 250\ 250\ 250\ 125\ 0\ 0]^T\ . \tag{19}$$

This spread was created by maximizing the capacity of the slot for the most desirable slots, and dividing the remaining agents between the remaining most desirable slots, preserving symmetry. Such a spread can only be "near-optimal" because the various utilities used optimize different things, and could allow other spreads to have slightly better utility. For all utilities used in this paper, this spread (or a similar one, for our experiments scaling the number of agents) allows a fair comparison.

6.2.1 Results of Bottom-Up Approach

Let us first focus on the bottom up perspective of having the desirability of time slots be directly derived from the intrinsic utilities of the agents. In these experiments we explore the performance of agents greedily pursuing their utilities (L_{AC}) with agents pursuing a modified version of their utilities (D_{AC}), and investigating the impact of the agents' choices in both cases on the slot capacities and consequently on their own utilities.

Figure 2 shows the performance of the local agent-centric, social welfare, and difference utilities in this model. All performance is measured by the social welfare function. Agents using the social welfare function directly were able to improve their performance only slightly. This is because the social welfare function has low signal-to-noise with respect to the agents' actions (an agent's action is masked by the "noise" of the other agents' actions). Even if an agent were to take a system wide coordinated action, it is likely that some of the 999 other agents would take uncoordinated actions at the same time, lowering the value of the global utility. An agent using the system utility typically does not get proper credit assignment for its actions, since the utility is dominated by other agents. The local agents performed very poorly in their overall utility, while agents using the difference utility reach near-optimal performance.

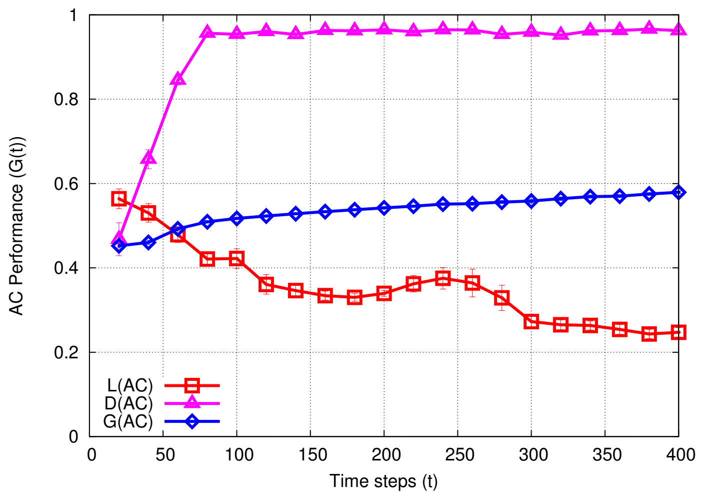


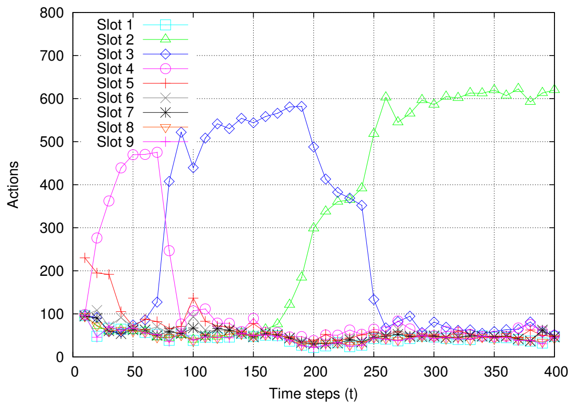
Figure 2: Performance on agent-centric social welfare function. Agents using their intrinsic preferences perform very poorly (.25), but agents using difference utilities achieve excellent performance (0.9). Performance was measured as a percentage of the optimum found by a hand-coded solution.

In part, the results of Figure 2 can be explained by examining the Nash equilibria various utilities are attempting to converge to. D_{AC} and G_{AC} are attempting to converge on a different equilibrium that L_{AC}; this explains the poor performance of L_{AC}. G_{AC} is unfortunately a very noisy signal and thus does not converge in the available time. One of the benefits that D_{AC} provides an equilibrium that is globally desirable, as shown in [3, 52]. See Section 7 for further discussion.

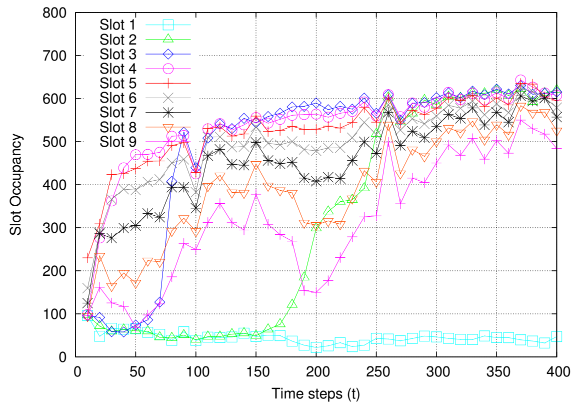
The counterintuitive result is that average agent utility improves when agents do not follow their local agent-centric utilities. Because of the non-linear interactions between slot congestion and arrival rates, agents that selfishly pursue their own utility cause congestion while those agents that consider their impact on others leave the system closer to the desired time slot more frequently. Figure 3 shows the attendance of agents in each slot when using either agent-centric local or difference utilities. Agents using the agent-centric difference utility manage to keep the critical slots remarkably free of congestion, excepting noise from ongoing ϵ-greedy selection. Agents using local utility greedily attempt to select earlier and earlier slots in order to leave the system in the desired slot, which only causes increased congestion.

Figure 4 graphs the results of experiments with the three utility functions so far discussed, two random policies, and different numbers of agents. As the number of agents are increased, the capacity of slots remains at 250/slot, so as may be seen here, local utilities perform well at first when sharing slots is easy. However as congestion increases, performance of the agents using local utility drops off sharply. Agents using difference utility continue to perform well as the number of agents increase, though this utility does suffer somewhat due to the approximate nature of the calculation.

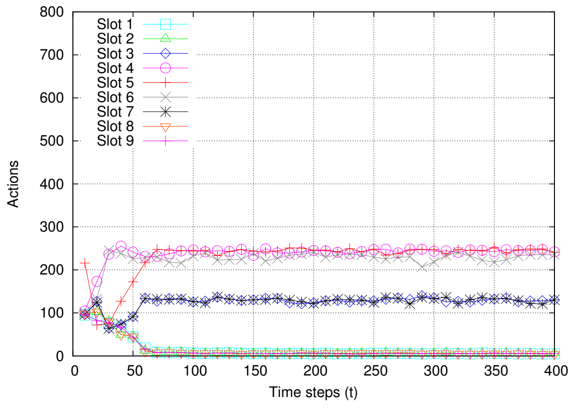
As may be expected, the performance of agents using the social welfare utility calculation improves as potential congestion increases. This is because agents using these utilities are spread somewhat randomly over the available slots. While congestion is low, this is a poor policy, however as congestion increases to the point where most slots will need to be full, a wide distribution of agents to slots is desirable.



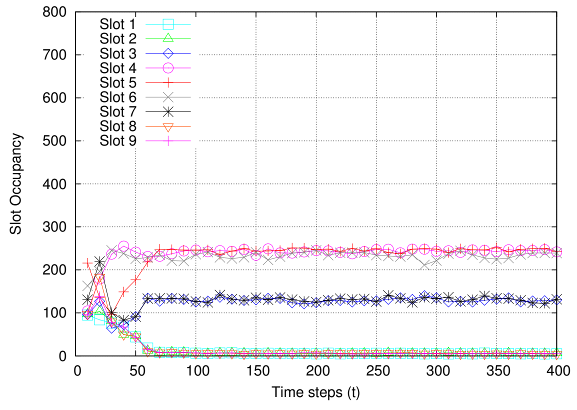
(a) Action histogram for L_{AC}



(b) Slot occupancy for L_{AC}



(c) Action histogram for D_{AC}



(d) Slot occupancy for D_{AC}

Figure 3: Action histograms and slot occupancy counts over time for L_{AC} (top) and D_{AC} (bottom). Agents using difference utility (based on social welfare function) form a distribution that is close to optimal. Agents using local utilities attempt to account for congestion by taking an earlier time slot, but fail to coordinate with other similarly-minded agents.

In Figure 4 we also included the performance of two hand-tuned policies. The "unweighted random" policy selects slots for agents with equal probability for each time slot. As may be expected, the results for this policy scale similarly to that of agents using social welfare utility. We also included results for a "weighted random" policy, where the probability of each agent selecting a slot was weighted according to the weight vector w (Equation (20)) which determines system-centric utility. As might be expected, performance using this policy improved as the number of agents increased, but dropped off as soon as congestion increased and the distribution of the weight vector no longer closely matched the optimal spread of agents. Note though, that this algorithm has access to centralized information (the weight vector) that individual agents do not.

In Figure 5, we conducted similar experiments where we also scaled the capacity of the slots in proportion to the number of agents using them. For example, 400 agents would have to share slots with a capacity of 100 agents/slot. These results show that performance is similar across all numbers of agents, demonstrating that the utilities used scale almost equally well in this situation. Social welfare utility scales slightly worse than other utilities due to the

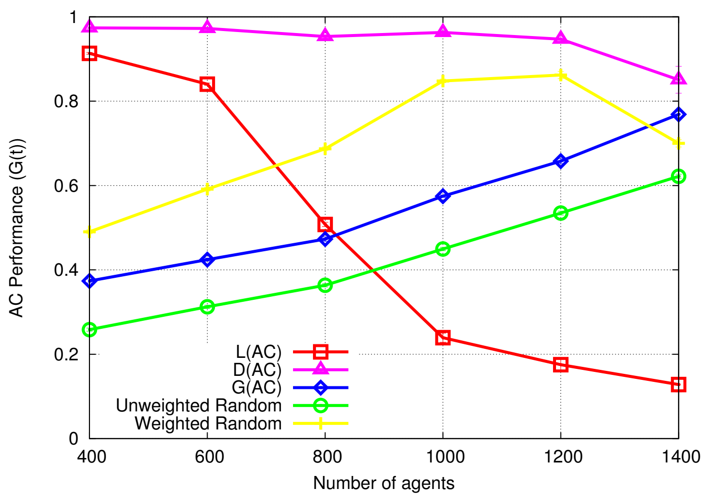


Figure 4: Performance of various utility functions and numbers of agents measured at the end of learning against agent-centric utilities. Local utilities scale poorly, while difference utilities handle various numbers of agents successfully.

decreased ability of agents to learn from the reward signal. For this particular 4:1 ratio of agents to capacity of slots, the performance of the weighted random policy performs nearly as well as the difference utility, however this would decrease a great deal as the ratio changes.

6.2.2 Results of Top-Down Approach

In this set of experiments we used various system-centric ("top down") utilities to learn a policy for the agents. As seen in Section 4, we require a weight vector to be defined that shapes the congestion to a particular configuration. The weighting vector we used for our experiments was shaped so its peak value was placed in the most desirable (center) time slot:

$$w = [0\ 5\ 10\ 15\ 20\ 15\ 10\ 5\ 0]^T\ . \tag{20}$$

The desirability of a time slot decreases linearly for earlier and later times. This experiment investigates the critical case where a congestion clears slowly, with agents exceeding capacity causing exponential harm to the system. Coordination becomes more critical in this case as even a small congestion can cascade into causing significant delays, a pattern that closely matches real-life congestion patterns.

This results shows that agents using the difference utility are able to quickly improve system performance (see Figure 6), though the performance still does not approach that of bottom-up utilities. In contrast, agents that try to directly maximize the system utility learn slowly and never achieve good performance during the time-frame of the experiment for the reasons discussed in Section 5.

Figure 7 shows the impact of scaling the number of agents using local, difference, and social welfare system-centric utilities. These results are similar to those found using bottom-up utility, however there are some important differences. Difference utilities do not perform as well here, likely due to the approximation used. Calculations using local utility actually start out better than difference utilities, but again rapidly decrease for the same reasons as

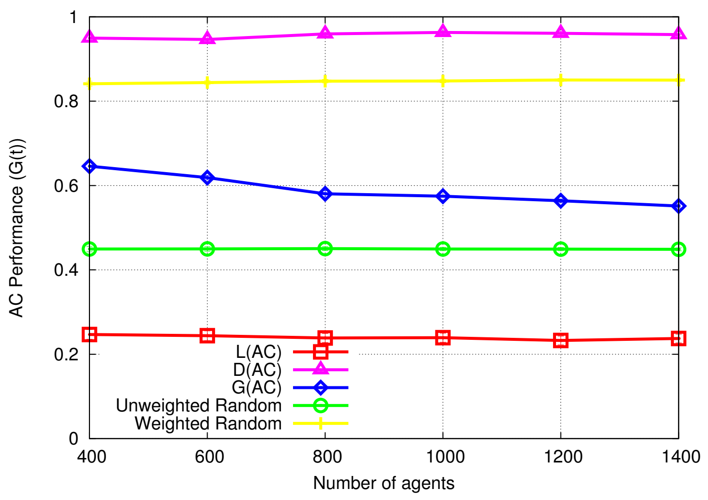


Figure 5: Performance of various utility functions and numbers of agents measured at the end of learning against agent-centric utilities. Capacity is scaled with the number of agents. The weighted random using the centralized knowledge about the weight vector policy performs nearly as well as using the difference utility.

in the bottom-up utility. Surprisingly, for 1000 and 1200 agents, the weighted random policy actually outperforms any other, but again, it is important to note that this policy has access to information not available to the learning approaches.

6.2.3 Agent/System Utility Mismatch

We now investigate the performance of agents using the utilities reported in Section 5 but measuring the system performance in terms of the top down utility. This situation is akin to agents using their own criteria for actions, but having a city manager measure system performance based on an overall congestion criteria. It is crucial to emphasize that none of the three utilities used by the agents during training aimed to directly maximize the utility by which we measure the system performance (system-centric utility). We are investigating this utility mismatch as it is critically relevant to many applications where - though there may be a system-wide metric (e.g., system-centric utility) - the individuals in the system use different criteria to choose their actions (e.g., the time slot in which they leave the system).

Figure 8 shows the performance of the agent-centric utilities, social welfare functions and difference utilities based on the social welfare function on the *system-centric utility*. Agents using the social welfare function directly are again unable to improve their performance very much, and agents using the their own intrinsic utilities harm the system-centric utility. Their performance can be explained by revisiting the arrival profile for the local agent-centric utility in Figure 3; most agents attempt to simultaneously account for congestion and select a slot close to their target. However, local utilities provide insufficient coordination and so heavy congestion is created in certain slots.

Agents using the difference utility learn to perform well from the system-centric perspective, and the agent arrival profile in Figure 3 demonstrates how this performance is achieved. The agents are able to learn a better estimate of how their actions impact the system, resulting in less congestion and overall better system-centric performance.

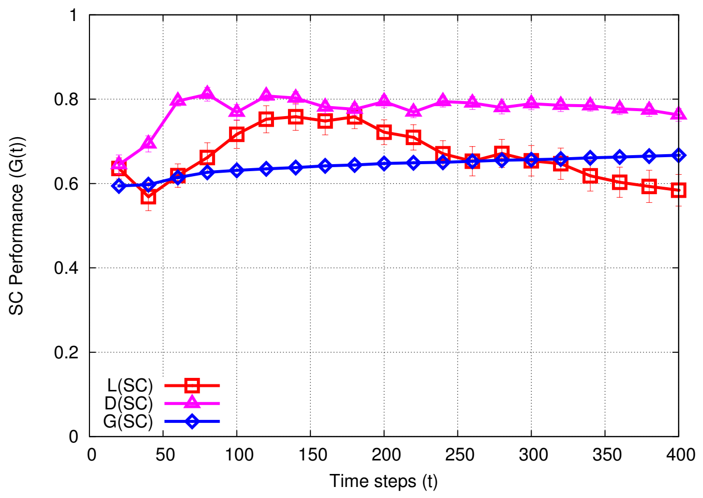


Figure 6: Performance on system-centric utility. Agents using different utilities outperform those using either local or social welfare utility functions.

Figure 9 shows how the performance of the various bottom-up utilities scales as the number of agents increase, as measured against system-centric utility. Results are similar to that of Figure 4, however the two random polices perform even better when measured against system-centric utilities. Again, though, difference utilities scale far better than social welfare or local utilities.

6.2.4 Changing Participation in Use of Difference Utility

In this section, we investigate the critical question of how many agents are required to adopt the difference utility in order for the effect to be felt by all. In other words, what if some agents "defect" and use their own intrinsic utilities. Will that cause the system performance to crash, or is system behavior robust to this type of behavior? Figure 10 shows the results for this experiment. This figure shows results for 1000 agents, some of which are using local utilities and some are using difference utilities. The percentage of agents using difference utilities was increased from left to right, with correspondingly increasing performance. As may be seen in this graph, significant improvements in social welfare utility were seen by the time 25% of the agents were using difference utilities. Results were measured against both forms of global utilities. Performance continues to improve as more and more agents use difference utilities, until about the 80% mark, where improvements in performance start to flatten out.

This result has two positive points: First, improvements start almost immediately with agents adopting the difference utility. Second, if we start with all agents using the difference utility, having a few agents defect will not dramatically impact system performance, preventing a mass-defection.

6.3 Multi-Peak Congestion

The multi-peak congestion domain has many more actions: a total of 89, corresponding to 89 time slots 0...88. Instead of a single time slot desired by all the agents, each agent is randomly assigned one of 11 time slots evenly spread through the action space at times 4, 12, 20, 28, 36, 44, 52, 60, 68, 76, and 84. Each agent receives higher reward for being assigned to its

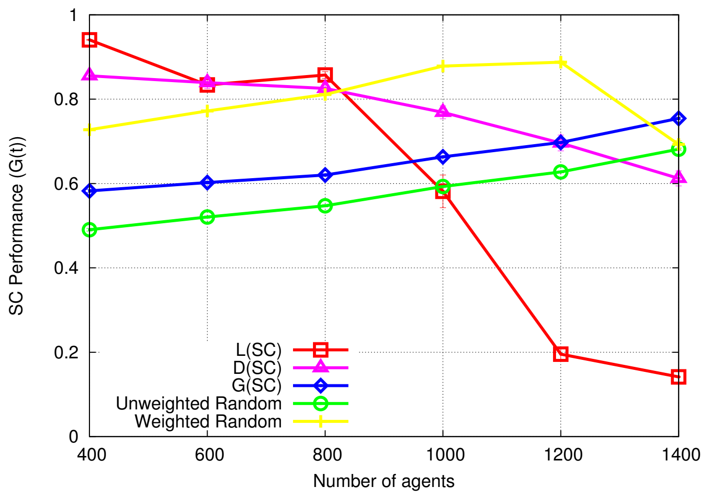


Figure 7: Performance of various utility functions and numbers of agents measured at the end of learning against system-centric utility. Local utilities scale poorly, while difference utilities handle various numbers of agents successfully.

desired slot. Our experiments all involve 1000 agents. We reduce the capacity of each slot to 25 in order to maintain a similar level of congestion to our single-peak domain experiments. This domain corresponds to some real-world domains in which multiple times or locations are desired by particular subsets of the agents in a system. For example, certain departure times at an airport might be more desired than others by the commercial airline industry. This kind of real-world problem could be abstractly modeled as a kind of multi-peak congestion domain.

Unless otherwise noted, experiments in the multi-peak domain used similar parameters as the single-peak domain. Due to the larger number of actions, we run our experiments for 1000 episodes. As with the single-peak domain experiments, we created a hand-coded spread of assignments of agents to time slots in order to measure the results against a near-optimal baseline. The spread was created by repeating the following pattern 11 times:

$$[0\ 0\ 8\ 25\ 25\ 25\ 8\ 0]\ . \tag{21}$$

Where the center "25" value corresponds to one of the peaks given above. Agents placed in this spread are assumed to prefer the closest peak time. This spread maximizes the capacity of the most desirable slots, and divides the remaining agents between the remaining most desirable slots, preserving symmetry. This spread can only be "near-optimal" because the various utilities used optimize different things, and thus other spreads might have slightly better utility.

Figure 11 shows the performance of the local agent-centric, social welfare, and difference utilities in this model. All performance is measured by the social welfare function. Agents using the social welfare function directly performed very poorly. Due to the large number of actions, the low signal-to-noise ratio of the reward signal here makes learning a good action extremely difficult. The local agents initially did well by finding their desired slots, but then performance dropped as those slots became more and more congested. Agents using the difference utility do comparatively very well, though the larger number of actions in this problem do not permit a solution as optimal as that found by similar results for the single-peak domain.

For the experiments shown in Figure 12, we used various system-centric ("top down")

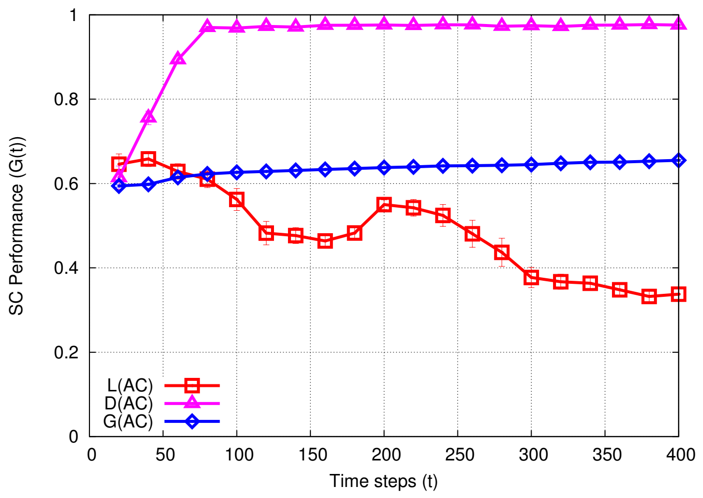


Figure 8: Performance with respect to system-centric utility of agents trained using agent-centric (local, difference and social welfare) utilities. Though none of the agents were aiming to maximize the system-centric utility (directly or indirectly), the system behavior promoted by the difference utility provided high values of the system-centric utility. Agents pursuing their own preferences lead the system to perform worse than random, as agents all attempting to leave the system at the same time slot frustrate each other and cause disastrous congestion.

utilities to learn a policy for the agents. As seen in Section 4, we require a weight vector to be defined that describes the desired distribution of agents. The weighting vector we used for our experiments was generated using a pattern, which we repeat 11 times:

$$[0\ 5\ 10\ 15\ 20\ 15\ 10\ 5]\ . \tag{22}$$

A trailing "0" completes the weight vector. This pattern places a "20" weight at each desired slot, with decreasing weight given to surrounding slots.

Because individual agents are no longer attached to particular desired slots, exploration is easier and thus the social welfare utility performs better than using the bottom-up reward. Using local utilities still performs poorly, as agents must keep trying different strategies as they greedily attempt to get the greatest local reward. Difference utilities outperform both these strategies, however, as they avoid the selfish exploration of local utilities while having a better signal-to-noise ratio than the social welfare function.

In the experiment shown in Figure 13, we measure the performance of the utilities reported in Section 5 against the system-centric utility. This experiment is similar to those performed in Section 6.2.3. The result shows the performance of the agent-centric utilities, social welfare functions and difference utilities based on the social welfare function on the *system-centric utility*. As with the experiments on the single-peak domain, agents using the social welfare function directly do not improve performance very much, and agents using the their own intrinsic utilities actually harm the system-centric utility.

Agents using the difference utility learn to perform well from the system-centric perspective, and even outperform agents that actually learn using the system-centric reward (Figure 12)! The agents are able to learn a better estimate of how their actions impact the system, resulting in less congestion and overall better performance from the system-centric perspective.

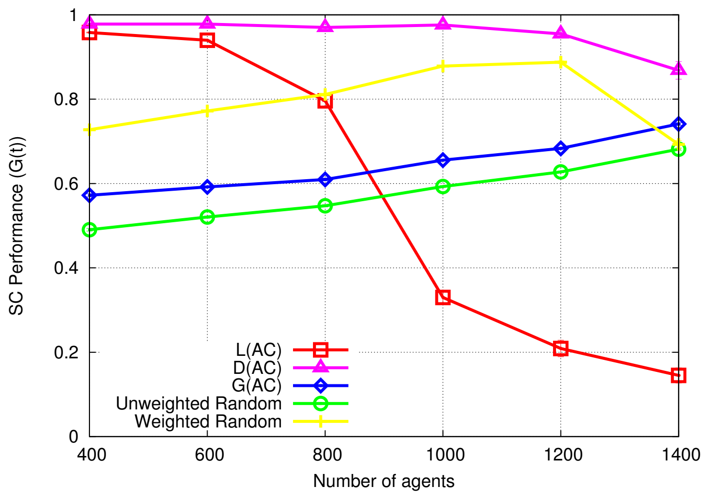


Figure 9: Performance of various bottom-up utility functions and numbers of agents measured at the end of learning against system-centric utility.

7 Discussion

This paper presented an agent coordination approach in two different types of congestion games. For each problem, we presented a *top-down* method by which agents can coordinate actions in order to alleviate spiking at peak congestion times. Then we presented a *bottom-up* method for improving social welfare when agents use an estimated difference utility instead of their own local agent-centric utility. This is an interesting result that states that agents *on average* reach higher values of their own intrinsic utilities if they do not aim to maximize it directly, but rather aim to maximize a modified version of that utility.

These results are based on agents receiving utilities that are both aligned with the system utility and are as sensitive as possible to changes in the utility of each agent. In these experiments, agents using difference utilities produced near optimal performance (about 97% of optimal). Agents using system utilities (55-60%) performed somewhat better than random action selection (45%), and agents using local utilities (25%) provided performance ranging from mediocre to much worse than random in some instances, when their own interests did not align with the social welfare function. The speed of convergence provided by the difference utilities (Figures 2, 6, and 8) is critical to these results as each agent that converges to a policy reduces the uncertainty present in other agents' learning. As more agents converge, all agents have a less "noisy" environment in which to learn which allows the agents to select good policies. These results extend earlier results showing the benefits of difference utilities to domains defined by intrinsic agent utilities rather than a system centric global utility as was used in previous work [1, 3, 46, 49].

In part, the excellent performance of difference rewards – and the success of the assumptions made in deriving them – can be explained by examining the "factoredness" of the various reward functions used. Factoredness [2] measures how much the shaped utility is aligned with

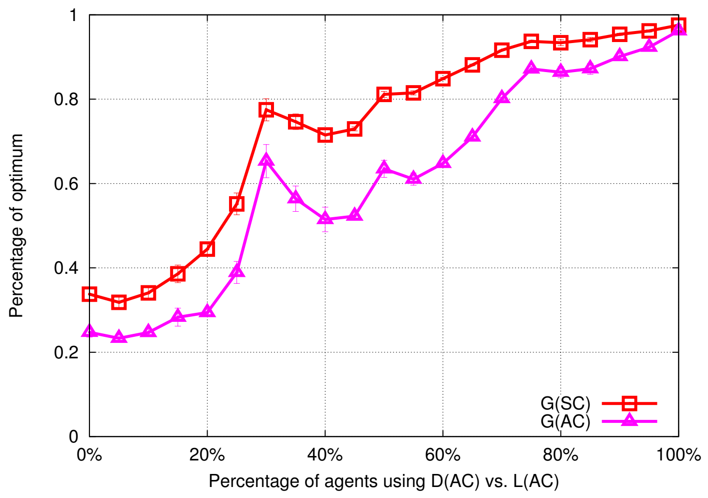


Figure 10: Changing performance of experiments with various percentages of agents using difference utilities over local utilities, as measured against both system-centric and agent-centric social welfare utility. The labels G(SC) and G(AC) indicate the utility that learning was measured against, not the utilities used to learn.

the system utility. Formally it is defined as:

$$F_{g_i} = \frac{\sum_{a}\sum_{a'} u\left[(g_i(a) - g_i(a'))(G(a) - G(a'))\right]}{\sum_{a}\sum_{a'} 1} \tag{23}$$

where the joint actions a and a' only differ in the action of agent i, and $u[x]$ is the unit step function, equal to 1 if $x > 0$. This definition keeps track of the cases where the change in the individual utility $g_i(a) - g_i(a')$ and the system utility $G(a) - G(a')$ have the same sign.

We use Equation 23 to estimate how factored various agent utility functions are compared to the top-down and bottom-up system utilities. As it is not possible to examine each of the huge number of possible combinations of actions of the agents using this equation, this forces us to use sampling techniques and generalize from them. We cannot pick samples from a uniformly random distribution of agent actions, because this would heavily bias the samples towards "flat" action histograms. To overcome this problem, we approximate the factoredness of a utility by collecting samples using the policy which we are calculating factoredness for. For each of the 400 episodes of a run, the rewards received by an agent in each slot was tested against the rewards it would have received for choosing any other slot. As we had 9 slots, we thus collected $9 \times (9 - 1) \times 400 = 28,800$ samples.

Table 1a shows that difference utilities are much more factored with the global utilities than local agent utilities. this is particularly striking for the bottom up case where the system utility is a straight sum of the agents' local utilities. But this result shows that agents that aim to maximize their own utilities harm the system performance. This is akin to the tragedy of the commons and has been observed in many problems of this type [21]. The difference utility does not suffer from this problem and this is one of the main reasons behind the performance of the difference utility in this domain. We can also now empirically justify the assumptions made in deriving each difference utility. If the assumptions did not hold, the difference utility

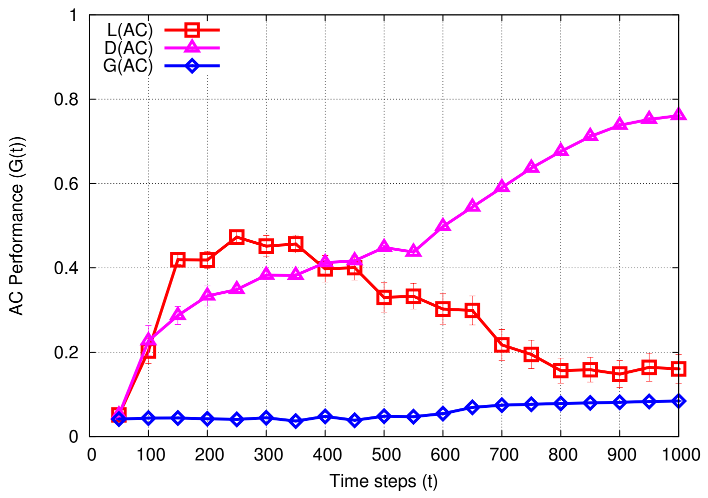


Figure 11: Performance on agent-centric social welfare function. Agents using their intrinsic preferences perform very poorly (.15), but agents using difference utilities achieve good performance (0.75). Performance was measured as a percentage of the optimum found by a hand-coded solution.

would not be factored with the global utility, and performance would decline.

Table 1b also partially explains the very good performance found when training on agent-centric utilities, but measuring against system-centric utility. Note that the factoredness of D_{AC} on G_{SC} is very high (.91) – though not higher than D_{SC} on G_{SC} (.98). This high factoredness explains why D_{AC} is suitable for learning when measured against G_{SC}, but because D_{AC} is not more factored than D_{SC} for this measurement, also shows that there is something else contributing to its good performance. In this case, D_{AC} is more sensitive to the actions of an agent, i.e. the utility signal is less noisy.

In addition to their good performance, difference utilities also provided consistency and stability to the system. Regardless of the system utility on which it is based, the difference utility aims to remain aligned with that utility, and promotes beneficial system-wide behavior in general. In the congestion domain, the system-centric utility and the social welfare function

Table 1: Comparative factoredness of difference and local utilities as measured against corresponding global utilities. Factoredness is measured as a percentage, so 1.0 is perfectly factored. Difference utilities are much mored factored than local utilities, leading to much better learning performance.

(a)

	G_{AC}
D_{AC}	0.88
L_{AC}	0.14

(b)

	G_{SC}
D_{SC}	0.98
L_{SC}	0.48
D_{AC}	0.91
L_{AC}	0.15
G_{AC}	1.00

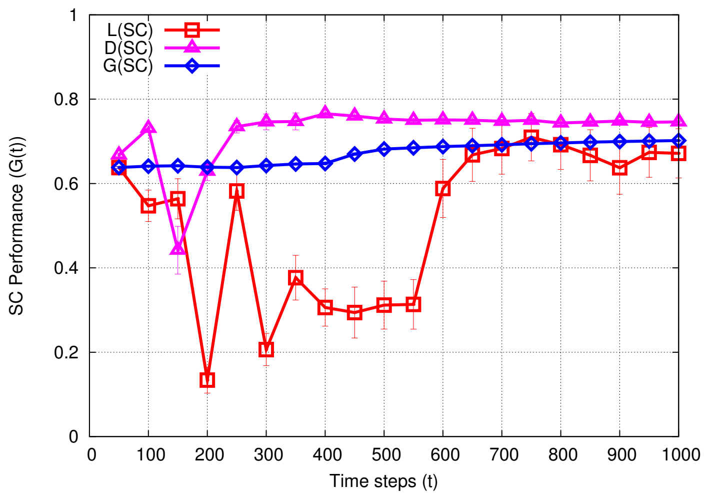


Figure 12: Performance on system-centric utility. Agents using different utilities outperform those using either local or social welfare utility functions.

based on the agent-centric utility were aiming to promote the concept of a "good congestion pattern". It is therefore not totally surprising that maximizing one utility allows the agents to perform well on the other. However, it is worth nothing that neither the agents directly aiming to maximize the social welfare function, nor the agents aiming to maximize their own preferences achieved this result. An interesting note here is that the system-centric utility is more concerned with congestion, whereas the agents are more concerned with delays (leaving the system at the target time slot).

One issue that arises in some congestion games that does not arise in many other domains (e.g., rover coordination) is in ensuring that agents comply with the utility function they are given. In this work, we did not address this issue, as our purpose was to show that solutions to the difficult congestion game can be addressed in a distributed adaptive manner using intelligent agents (However, in Section 6.2.4 we provided brief results on the impact of agents defecting.) Ensuring that agents follow their given utility is a fundamentally different problem. On one hand, agents will notice that maximizing the difference utility does lead to better values of their own intrinsic utilities. On the other hand, there is no mechanism to ensure that they do so. Indeed, this is an area where the proposed method and mechanism design are complementary, in that from a computational perspective, it might be more fruitful to provide incentives for agents to follow their difference utilities than to provide incentives on their intrinsic utilities that lead to globally desirable equilibria.

Acknowledgements: We gratefully acknowledge Zachary Welch for performing early versions of these experiments. This work was partially supported by NSF grant 0931591.

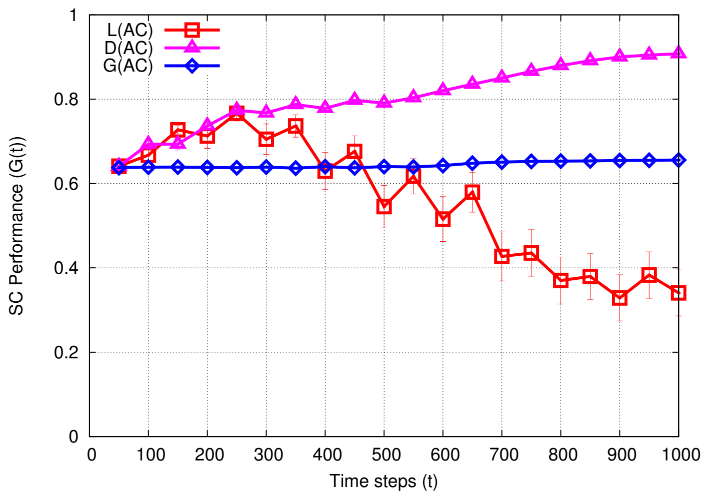


Figure 13: Performance with respect to system-centric utility of agents trained using agent-centric (local, difference and social welfare) utilities. Though none of the agents were aiming to maximize the system-centric utility (directly or indirectly), the system behavior promoted by the difference utility provided high values of the system-centric utility. Agents pursuing their own preferences lead the system to perform worse than random, as agents aiming to arrive one time frustrate each other and cause disastrous congestion.

References

[1] A. Agogino and K. Tumer. Regulating air traffic flow with coupled agents. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Estoril,Portugal, May 2008.

[2] 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.

[3] A. K. Agogino and K. Tumer. Efficient evaluation functions for evolving coordination. *Evolutionary Computation*, 16(2):257–288, 2008.

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

[5] M. Balmer, N. Cetin, K. Nagel, and B. Raney. Towards truly agent-based traffic and mobility simulations. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 60–67, New York, NY, July 2004.

[6] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama. Dynamical model of traffic congestion and numerical simulation. *Physical Review E*, 51(2):1035–1042, 1995.

[7] A. Bazzan and F. Kluegl. *Multiagent Architectures for Traffic and Transportation Engineering*. Springer, 2009.

[8] A. L. Bazzan and F. Klügl. Case studies on the Braess paradox: simulating route recommendation and learning in abstract and microscopic models. *Transportation Research C*, 13(4):299–319, 2005.

[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] 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.

[11] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, April 2002.

[12] Ronen I Brafman and Moshe Tennenholtz. Efficient learning equilibrium. *Artificial Intelligence*, 159(1-2):27–47, 2004.

[13] B. Burmeister, A. Haddadi, and G. Matylis. Application of multi-agent systems in traffic and transportation. *IEEE Proceedings in Software Engineering*, 144(1):51–60, 1997.

[14] C. R. Carter and N. R. Jennings. Social responsibility and supply chain relationships. *Transportation Research Part E*, 38:37–52, 2002.

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

[16] J. Cheng. The mixed strategy equilibria and adaptive dynamics in the bar problem. Technical report, Santa Fe Institute Computational Economics Workshop, 1997.

[17] Vincent Conitzer and Tüomas Sandholm. Complexity results about nash equilibria. In *Proceedings of the 18th international joint conference on Artificial intelligence*, IJCAI'03, pages 765–771, 2003.

[18] K. Dresner and P. Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 530–537, New York, NY, July 2004.

[19] T. Groves. Incentives in teams. *Econometrica: Journal of the Econometric Society*, 41:617–631, 1973.

[20] S. Hall and B. C. Draa. Collaborative driving system using teamwork for platoon formations. In *The third workshop on Agents in Traffic and Transportation*, 2004.

[21] G. Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.

[22] D. Helbing. Structure and instability of high-density equations for traffic flow. *Physical Review E*, 57(5):6176–6179, 1998.

[23] D. Helbing. Traffic and related self-driven many-particle systems. *Reviews of Modern Physics*, 73:1067–1141, 2001.

[24] D. Helbing and B. Tilch. Generalized force model traffic dynamics. *Physical Review E*, 58(1):133–138, 1998.

[25] 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.

[26] B. A. Huberman and T. Hogg. The behavior of computational ecologies. In *The Ecology of Computation*, pages 77–115. North-Holland, 1988.

[27] Samuel Ieong, Robert McGrew, Eugene Nudelman, Yoav Shoham, and Qixiang Sun. Fast and compact: a simple class of congestion games. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 2*, AAAI'05, pages 489–494, 2005.

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

[29] 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.

[30] B. S. Kerner and H. Rehborn. Experimental properties of complexity in traffic flow. *Physical Review E*, 53(5):R4275–4278, 1996.

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

[32] A. A. Lazar, A. Orda, and D. E. Pendarakis. Capacity allocation under noncooperative routing. *IEEE Transactions on Networking*, 5(6):861–871, 1997.

[33] 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.

[34] K. Nagel. Multi-modal traffic in TRANSIMS. In *Pedestrian and Evacuation Dynamics*, pages 161–172. Springer, Berlin, 2001.

[35] D. Oliveira, A. L. C. Bazzan, B. C. Silva, E. W. Basso, L. Nunes, R. J. F. Rossetti, E. C. Oliveira, R. Silva, and L. C. Lamb. Reinforcement learning based control of traffic lights in non-stationary environments: a case study in a microscopic simulator. In Barbara D. Keplicz, Andrea Omicini, and Julian Padget, editors, *Proceedings of the 4th European Workshop on Multi-Agent Systems, (EUMAS06)*, pages 31–42, December 2006.

[36] Christos Papadimitriou. Algorithms, games, and the internet. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, STOC '01, pages 749–753, 2001.

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

[38] D. C. Parkes. *Iterative Combinatorial Auctions: Theory and Practice.* PhD thesis, University of Pennsylvania, 2001.

[39] David C. Parkes and Jeffrey Shneidman. Distributed implementations of Vickrey-Clarke-Groves mechanisms. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '04, pages 261–268. IEEE Computer Society, 2004.

[40] Robert W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[41] T. Sandholm and R. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37:147–166, 1995.

[42] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, July 2000.

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

[44] Moshe Tennenholtz and Aviv Zohar. Learning equilibria in repeated congestion games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pages 233–240, 2009.

[45] Gerald Tesauro and Jeffrey O. Kephart. Pricing in agent economies using multi-agent q-learning. *Autonomous Agents and Multi-Agent Systems*, 5:289–304, 2002.

[46] 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.

[47] K. Tumer, A. K. Agogino, and Z. Welch. Traffic congestion management as a learning agent coordination problem. In A. Bazzan and F. Kluegl, editors, *Multiagent Architectures for Traffic and Transportation Engineering.* Springer, 2009.

[48] 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.

[49] K. Tumer and D. Wolpert. A survey of collectives. In *Collectives and the Design of Complex Systems*, pages 1,42. Springer, 2004.

[50] K. Tumer and D. H. Wolpert. Collective intelligence and Braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, Austin, TX, 2000.

[51] Marco Wiering. Multi-agent reinforcement leraning for traffic light control. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 1151–1158, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

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