Leadership Games and their Application in Super-Peer Networks

Thomas J. Walsh and **Javad Taheri** and **Jeremy B. Wright** and **Paul R. Cohen**
Department of Computer Science, University of Arizona, Tucson, AZ
{twalsh,taheri,wrightjb,cohen}@cs.arizona.edu

Abstract

This paper considers a setting where a single "leadership agent" intervenes in a multi-agent system through actions that (perhaps subtly) change the dynamics of the system. We describe a number of forms this intervention can take and compare these situations to settings in previous work. We identify two important effects of leadership: faster system convergence, and convergence to a better equilibrium. Empirically, we first explore these properties in leadership of algorithms engaged in classical 2-player games. We then apply this general framework to the leadership of a super-peer file-sharing network. In these experiments the network contains some agents that make locally greedy decisions that hamper the network as a whole. We show that a leader acting based on a more global criteria can push the system to a better equilibrium point as well as speeding up convergence. We also show how a mathematical approximation of such super-peer networks can be used to aid a leader in determining a minimum-cost intervention strategy.

Introduction

Within the multi-agent systems (MAS) community, the problems of manipulating agents by a self-interested player has been studied in numerous settings. However, less attention has been focused on the problem of an *external* agent, playing with different actions and different (perhaps more global) utilities, manipulating the system as a whole. Such situations seem to occur often in practice. For instance, consider a network of complicated trading agents exchanging resources. A "commissioner" agent in this situation may have to choose whether to block certain trades based on the cost of this intervention and its value of certain global properties (e.g. avoiding monopolies).

In this paper, we describe similar situations as *leadership games*. Leadership games involve a system of multiple (potentially learning) agents as well as an external agent called the *leader*. The leader will typically have a different set of available actions and a more global utility function than the other agents in the system. However, we will typically assume that the set of leadership actions is not enough to completely change the system (which would make this essentially a mechanism design problem). Instead, the leader may only be able to subtly prod the system by giving rewards or extra resources, or constricting the availability of resources or actions, all with respect to particular agents.

In general, this paper focuses on two important functions a leader can serve with respect to an agent system. First, a leader may improve the convergence *rate* of a system, that is decrease the amount of time needed for the behaviors of the agents to become stationary. Perhaps more importantly, we show that if the system has multiple possible convergence points (or *equilibria* of player policies), a leader can actually influence which one the system will converge to. Moreover, this can often be done without overwhelming the dynamics of the original game, through a limited-time intervention by the leader who can stop intervening once the system has reached the desired equilibrium.

While we investigate these properties briefly in Bimatrix games, the main application of our current work is in leading a type of peer-to-peer file sharing network called a *Super Peer Network*. These networks are comprised of two types of agents, *weak peers*, who make requests for files in the network and *super peers* who act as middlemen in the system and keep pointers to files in limited-size caches. We study a case where some of the super peers manage their caches greedily, maximizing their individual profit, but damages the system as a whole as many files become inaccessible and more altruistic super peers are starved. We show how a number of different leaders can affect such a system's convergence rate and asymptotic behavior through a limited intervention. Furthermore, we show that using a simplified mathematical model of the network, a leader can be constructed that attempts to minimize the cost of its intervention while still pushing the system to better performance.

Leadership Games

We now provide a general definition of the kinds of multi-agent systems that support leadership games and some prototypical action sets and utility functions that might guide a leader's behavior. We then describe how the leader's ability to influence various elements of the MAS can turn a leadership game into familiar scenarios from the literature.

General Definition

A leadership game $\mathcal{G}_{\mathcal{L}} = \langle \mathcal{G}, \mathcal{L} \rangle$ consists of a *base game* $\mathcal{G}$, as well as a leader $\mathcal{L}$. $\mathcal{G}$ is an MAS containing n play-

Copyright © 2011, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

ers, where each player i has an action set A_i and is guided by some utility function U_i. In general, it will be helpful to consider some typical game constructs such as attitudes and resources. Attitudes α_{ij} capture the disposition i towards j. The resources in the game R are a set of typed elements that each agent can acquire (or lose) through actions. For instance, in an auction setting, the resources may be money and the items to be purchased, and attitudes may be adjusted based on who is winning or losing in the different episodes. The utility function U_i is usually a function over the resources held by each agent, with weights or other constructs determining the relative allure of each resource to the given player. We will often refer to the empirical one-step utility as a *payout*. We consider all of the agents in the system to be attempting to maximize their utility over the lifetime of the game. Thus, agents may end up acting adversarially, negatively affecting the overall utility of the system. For instance, in an auction system, one rogue agent with many resources could simply outbid all the other impoverished agents, leading to low global measures of utility.

In a leadership game, the leader is usually tasked with making sure the system avoids such extremes or reaches a stable equilibrium that maximizes some level of global utility without costing the leader too much. More specifically, the leader exists externally to $\mathcal{G}$ in that the other players cannot directly affect his available actions or resources, but their behavior can affect his utility. The leader has its own action set $A_{\mathcal{L}}$ which it can use to influence the dynamics of $\mathcal{G}$. The leader takes actions in order to maximize its own utility function $U_{\mathcal{L}}$, which we generally assume is related to the overall performance of $\mathcal{G}$. We assume that the agents in $\mathcal{G}$ can view the leader's actions and therefore may have attitudes towards him (and vice versa) and the leader may have finite resources to use in the domain. Hence, the leader may need to make decisions about the most economical use of its manipulation abilities.

Typical Leadership Actions, Goals and Effects

We now describe a number of prototypical actions and utility functions that a leader may be equipped with. The actions for a leader will generally involve either changing the allocation of resources or constraining a player's available actions.

A leader that can change the allocation of resources may take on several forms. In simple two player Bimatrix games, where resources are only allocated as payouts (that immediately disappear), we will see that adjusting the payouts through reward shaping can change the resource allocation in $\mathcal{G}$ enough to change the behavior of the players. In larger games, changing the payouts on each turn can amount to bribery and changing the allocation of resources on each turn is similar to market design problems (as discussed below). A leader may also have the ability (as we use in our later experiments) to "block" certain agents from taking certain actions or at least limit the probability of the action's success.

In terms of utility functions, the leader may be driven by any of a diverse set of measures, including for instance, the average utility at each step for each of the n agents, or in more complex games, measures such as the total throughput of an information network or a measure of overall market performance. Whatever the motivating factor for the leader, we consider three important measures of its success:

1. The leader can affect the convergence rate of the system. That is, suppose $\mathcal{G}$ has a single equilibrium or distribution over equilibria. The leader may not have the resources or the capabilities to change the asymptotic behavior of the system, but he may be able to get it to converge faster by, for instance, making payouts larger or more deterministic.
2. The leader may change the equilibrium that $\mathcal{G}$ converges to. Specifically, when there are multiple potential equilibria in $\mathcal{G}$, the leader can potentially push the agents towards a more desirable (in terms of $U_{\mathcal{L}}$) asymptote. In situations where the desired equilibrium is stable without $\mathcal{L}$, the leader can potentially stop performing actions after the system has stabilized.
3. The leader may have constrained resources or costs of its own for intervening. Therefore criteria (1) and (2) may need to be balanced against the leader's budget.

Leadership in Related Work

We now describe how realizations of leadership games in prior work can be viewed through the general framework described above. First, we note that if the leader has full control over the dynamics of resources and payouts, and full knowledge of the agents in $\mathcal{G}$, the leadership game is essentially a mechanism design problem (Conitzer and Sandholm 2007; Guo and Conitzer 2010). This is a large field in MAS where the construction of voting, auction, and other systems have been studied in situations where agents may misrepresent their types. However, these works often assume the ability to change almost all of the parameters of the game. In leadership games, the leader's actions are often far more constrained and the leader needs to act dynamically against an MAS that may have several uncontrollable parameters. Similarly, the kind of problems often seen in the *Market Design* track of the Trading Agent Competition (http://tradingagents.org/) can be viewed as leadership games where multiple leaders are competing for the allegiance of the agents in $\mathcal{G}$. Strategies in this vein range from rule-based systems (Petric et al. 2008) to systems that dynamically profile agents in $\mathcal{G}$ (Gruman and Narayana 2008).

In situations where the leader can (with cost constraints) only affect the resources or payouts on each step, the problem is similar to efforts in multi-agent reward shaping in simple games and manipulation in more complex domains. Work in multi-agent reward shaping has shown the increased convergence rate (Babes, de Cote, and Littman 2008) described above and we show below that it can also be used to push agents towards different equilibria. In larger systems, a leader's ability to change the reward allocation or payouts is similar to studies of bribery in voting systems (Faliszewski, Hemaspaandra, and Hemaspaandra 2006; Yoram Bachrach and Faliszewski 2011). Indeed, recent work on voting manipulation has analyzed the complexity of minimizing the cost to the leader in different conditions (Bachrach et al. 2009). However, it is not clear if these lessons can be generalized beyond the specific systems studied.

In the opposite case, where the leader does not control resources but can only directly manipulate attitudes, leadership games look very much like the influence networks studied in social network theory and indeed the use of approximations for planning by a leader in such domains (Hung, Kolitz, and Ozdaglar 2011) mirror results we present in our super peer case study.

Finally, we note the connection to classical game theory and dynamical systems in several settings. For instance, if there are no learning agents in the MAS, then the leadership game is simply a two player interaction with a static opponent (the MAS). If the leader is able to force players to play actions from a distribution, the leader may be acting as an information source for a correlated equilibrium, though generally we do not consider leaders that can directly manipulate players in this way. In the case where the leader has the same action set as others in the MAS and also the same utility function, leadership games mirror the problem of leadership in multi-player games where agents can benefit by committing to a leader-follower pattern (Littman and Stone 2002).

Leadership in Bimatrix Games

We now provide some basic experimental results on leadership in Bimatrix games with two agents and 2 actions (cooperate and defect). In the base games, the only resources are the payouts on each turn and attitudes are captured indirectly by the action values. The leadership action we will consider in this setting is reward shaping (Ng, Harada, and Russell 1999), that is, a small amount of reward given (or subtracted) by the leader to one or both agents when they perform actions the leader supports (or dislikes). Prior work (Babes, de Cote, and Littman 2008) has studied the use of admissible shapes in such games, specifically when the shape at time t is determined by a difference of potentials $\gamma\phi(s_t) - \phi(s_{t-1})$ where the states s are determined by the representation of the individual learners. In such cases, shapes do not change the set of subgame-perfect equilibria for the particular players in $\mathcal{G}$, but instead act as a bias (which can be overcome through data) that can increase the convergence rate of the system. Here we will concentrate on how a leader that injects admissible or inadmissible shaping rewards into the system can change the equilibrium the system converges to.

First, we present a result in the Assurance game (Littman and Stone 2002) where payouts are 3 each for mutual cooperation, 1 each for mutual defection, and a cooperate/defect pairing yields 0 to the cooperator and 2 to the defector. There are two pure Nash equilibria in this game, either mutual cooperation or mutual defection. Figure 1 (left) shows the effect of leadership with an admissible reward shape in this game on two Q-learners (Watkins 1989) with 1 step of memory and using a learning rate of $\alpha = 0.1$ and non-decaying ϵ-greedy exploration with $\epsilon = 0.1$. Specifically, the leader, who covets the higher global utility from mutual cooperation, provides a reward shape based on $\rho(s) = -5$ for states where the agent just defected and otherwise $\rho(s) = 0$. Again, because the actual shape is calculated by $\gamma\phi(s_t) - \phi(s_{t-1})$, this only biases the agents and does not change the equilibria. The figure shows cooperation rates

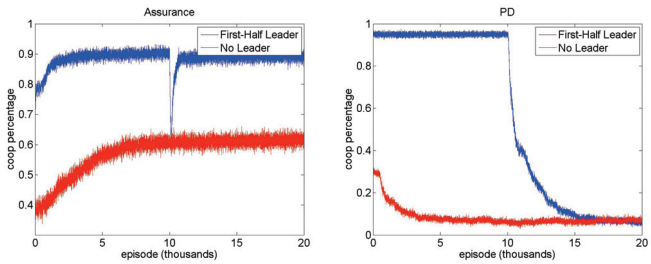

Figure 1: Percentage of cooperation (averaged over 500 runs) with and without leadership leadership in Assurance (left) and Prisoner's Dilemma (right). In both cases the leader stops shaping halfway through the run.

when a leader that injects such reward shapes over *only the first half of a trial*. Even this temporary leadership has the effect of biasing the learners towards convergence at the more globally appealing point, whereas without the leader, this best global equilibria is only reached approximately 60% of the time (mirroring the original results of (Littman and Stone 2002)). One should also notice the dip in cooperation just as the leader leaves the system, which occurs because the values for cooperation decrease when the shapes are removed, leading to a flirtation with defection, though the system quickly recovers. We have noticed this "leadership bump" often occurs when a leader starts or stops interaction with a system, and we will see this phenomenon again in our larger case study.

It is also possible for a leader to provide inadmissible shaping rewards, but the effect of these incentives is only temporary as they tend to briefly overwhelm the game before the leader steps out. Figure 1 (right) shows the percentage of cooperation among two single-state (memoryless) Q-learning agents in the classic Prisoner's Dilemma game with payouts of 3 for mutual cooperation, 1 for mutual defection and 5 to the defector (0 to the cooperator) when the actions differ. Here we see a leader that uses an inadmissible reward shape of -5 to any defectors, thereby overwhelming the game's structure, but once the leader withdraws, the agents largely fall back into mutual defection (the Nash equilibrium for $\mathcal{G}$). This effect is not always as immediate– we have seen that Q-learners with memory (multiple states) in this game will maintain a slightly higher percentage of cooperation (around 40% after the leader versus 30% with no leader at all), but this tendency does eventually decay (thousands of steps later).

We have now shown that leaders can affect the convergence of simple systems through reward shaping, and that admissible reward shaping in this setting allows the agent to provide these bias payments for only a short amount of time. Combined with the previous findings on reward shaping improving convergence rates, these results are encouraging, but our goal is to be able to influence a large MAS, not only simple Bimatrix games. To that end, we now present a case study of leadership in a super-peer network, a complex file sharing system, where we will see the end results mirror these lessons from Bimatrix games.

Super-Peer Networks: A Case Study

We now describe a case study with a much larger base game, specifically a super-peer file-sharing network. Our specific instantiation of this network contains two types of super peers (the proxy agents in the file sharing scheme), one that acts in a way that promotes global throughput and another that acts greedily but hurts the system as a whole. We show how different leaders can affect such a system, including situations where the leader needs to minimize its own costs.

Super-Peer Network Overview

In standard Peer-to-peer (P2P) systems, every machine (called a *peer*) hosts data and this decentralization results in desirable properties such as better reliability, improved scalability, and availability (Milojicic et al. 2002). The standard interaction in such a network is that peers submit queries for a file to some other peers in the network, and if any of these "neighboring" peers who either respond with a match or propagate the query.

In this work, we consider a slightly more structured P2P system called a *super-peer network* (Yang and Garcia-Molina 2003) as illustrated in Figure 2 (SPN). This network comprises two types of peers, *weak peers* (WP) that hold and trade all of the files and *super peers* (SP) which hold cached pointers to weak peer files and act as search proxies for the weak peers. Specifically, super peers can respond to *local* queries based on their own cache or *global* queries which they forward to the larger system.

We will use an implementation of super-peer networks close to the architectures described in the networking literature (Garbacki, Epema, and van Steen 2007). In this model, each weak peer can *pledge* itself to one or more super peers at each timestep, and those are the super peers it will send its queries to, on that step. When a super peer s receives a query from a weak peer, it first checks its own cache of files. If the file is found, then it responds with the file pointer. Otherwise, if the query is marked global, it forwards the query to its neighboring super peers. In this work, since our leaders will not be able to influence the topology of the super peers' interconnection, we assume that the super peers are strongly connected, so only one step of forwarding is needed. If the file pointer is in any super peer's cache, it provides super peer s with the pointer, which s then sends to the original weak peer. Notice that the super peers keep only pointers to files, not the files themselves (which the weak peers handle).

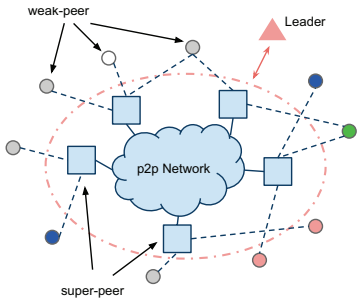

Figure 2: super-peer network structure

Agent Types, Utilities and Actions

We now describe the components of our super-peer network implementation in terms of the base-game parameters mentioned earlier. We will begin by describing the agent actions, utilities and types.

On every step, weak peers can choose up to 5 (in our experiments) super peers to pledge to, and if the weak peers are making a local query on this step (and don't have pending queries), they send it to those super peers. If they make a global query, it is sent to one of their super peers at random. On every step, a super peer can (1) respond to a weak peer's query if it has the file in its cache or forward a global search from a weak peer and (2) respond to a global search from another super peer. If a new file pointer is seen by the super peer as a result of a global search, it may also have to decide how to store this pointer in its limited-size cache. Super peers may have limited bandwidth (discussed in the next section) and so may have to decide which queries to answer.

The choices in each of these actions (who to pledge to, when to answer queries, how to update a cache) depend on each agent's own utility function. In our system, these utilities will be governed largely by the *subtype* (categories within the weak and super peer sets) of each agent. We consider each weak peer to be of a semantic type $\tau \in T$. The set of types corresponds to the set of different file types being shared in the system. In our experiments, weak peers will only request files of their own type, though in general a multinomial distribution could be used to model more diversified weak peers.

We also consider there to be two subtypes of super peers: *greedy* and *altruistic*. Greedy super peers attempt to maximize the number of hits from their client's requests in their cache of file pointers. Because their cache is of limited size, this often means deleting pointers to less popular (among their clients) pointers in favor of more popular ones. Although this greedy behaviour will boost the super peer's popularity locally, deleting these file pointers could be detrimental to the system's performance. By contrast, altruistic super peers seek to maximize the number of hits of their clients' requests but also guard against two possibly destructive cases: (1) They make sure file pointers they hold are not overwritten without a copy being made and (2) they do not store file pointers that exist elsewhere in the system. For the first case, when they are asked for a global search, they reject it if they do not have an empty slot in their cache. For the second rule, if they receive a global search query from a super peer that will copy the forwarded pointer, they delete their own copy. At a high level, greedy super peers base utility on the number of requests they manage to serve immediately while altruistic super peers have a utility that also factors in the performance of the system as a whole.

Resources and Attitudes

We now consider how the actions of different agents affect the resources in the system and attitudes of other agents. The set of resources (R) in an SPN are the files for the weak peers, the file pointers for each super peer and the amount of bandwidth for each super peer (how many requests it can

serve on each step). Each file has a type $\tau \in T$, such as movies, music, audio-book, etc. Based on the findings of (Cholvi, Felber, and Biersack 2004), we use a Zipfian distribution over the type of each weak peer and the type of each file in the system. This leads to a realistic skewing of the file and weak-peer types: file types that are the most requested are also the most prevalent. The file pointers are held by the super peers (initially randomly assigned) and updated based on their types (greedy or altruistic) as described earlier. As for bandwidth, we assume for now that each super peer has sufficient bandwidth to answer all its incoming queries, though we will later consider a leader that can "cap" this resource.

The decision by a weak peer i on which super peers to pledge to are governed by their attitudes towards each super peer j, α_{ij}. These attitudes are adjusted after each search query by i to j in the following manner:

1. If j answers a local search successfully then α_{ij} is increased by 1.
2. If j resolves a global search and is known to i successfully then α_{ij} is increased by 1, otherwise α_{ij} is newly initialized.
3. If i was pledged to j and j rejects the request (because of a bandwidth limitation), α_{ij} is decreased by 2.

The decisions by the super peers in responding to queries only need to be dealt with when bandwidth is limited and we assume here that the super peers randomly pick as many queries as they can serve within their bandwidth limits (that is they do not keep explicit attitudes towards weak peers). As for updating their cache of file pointers, again following the general outline of (Garbacki, Epema, and van Steen 2007) we assume that every file pointer considered by a super peer SP_i has a *score* β_{if} updated in the following manner.

1. When a weak peer does a local search for f, if f is already in the super peer's cache, β_{if} is increased by 1.
2. When a weak peer asks for a global search from SP_i, SP_i forwards the query to other super peers. Eventually another super peer SP_j responds to this query through SP_i. Then, if its cache is full, SP_i removes the item with the lowest β_i, and replaces it with f and sets β_{if} to $max_j\beta_{ij}$.

The interplay between actions, resources and attitudes can lead to two important phenomena in the system that our leadership agents will try to affect. First, the presence and potential popularity of greedy super peers can lead to *file loss* in the system in the sense that, if the only super peers with pointers to a certain file are greedy, and that file is not very popular, they all might overwrite that pointer in favor of more popular files, therefore rendering the file inaccessible in the system. A second complex system-wide feature is that of *specialization*. Because weak peers tend to pledge to super peers that often have the files they are looking for, and because super peers cache pointers that are popular among their pledges, the system tends to trend (or converge) to a specialized network where super peers cache mostly files of and have pledges mostly from a single type (see (Garbacki, Epema, and van Steen 2007)). In our experiments we noticed that greedy peers tend to specialize towards only the most popular file types, altruistic super peers often choose to act as niche stores, allowing weak peers of a less popular type to still find their files. Both of these phenomena can have a strong effect on global throughput measures and we will now see how a leader can improve such global measures by controlling these system-wide properties.

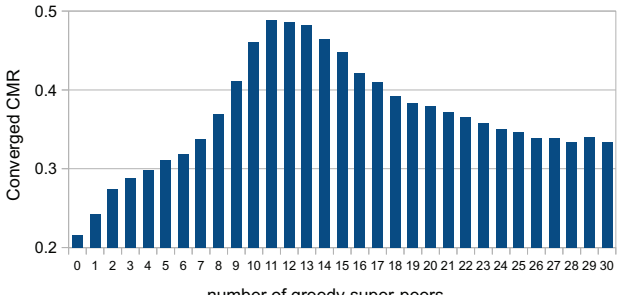

Figure 3: Converged CMR (10 runs) for different combinations of greedy and altruistic super peers (total 30).

Leadership in Super-Peer Networks

In general, our leaders will be trying to mitigate the effect of greedy super peers in the system, by limiting file loss, promoting specializations and otherwise trying to guide the system to better global throughput. To show the effect of greedy super peers without a leader's intervention, we have run experiments with different combinations of the altruistic and greedy super peers (always a total of 30). As shown in Figure 3, the system performs best when there are fewer greedy super peers in the system.

We will consider two different utilities $U_{\mathcal{L}}$ in our experiments, limiting file loss and also minimizing the *average cache miss ratio(CMR)* of the weak peers. The average CMR of the system is an important performance metric in super-peer networks, because a lower CMR means less average time for a weak peer to wait until its query is resolved. Note that if a query is missed, it has to be forwarded to all other super peers in the network, which will take much longer than if it hits in a pledged super peer's cache. In other words, a high CMR results in more global search, which means more packet forwarding and lower throughput.

We will also experiment with different leader action sets ($A_{\mathcal{L}}$) to improve the measures above. First, a leader can *cap* a super peer's bandwidth. This can be done either by restricting the percentage of requests that get through or the absolute number, both of which we experiment with. This type of action will be most helpful in changing the equilibrium the system converges to, in this case hopefully improving long term CMR or file loss statistics. For this type of action, the leader limits the bandwidth of the greedy super peers in the system for a limited period of time, hoping that this will lower the weak peers' attitudes toward the greedy super peers and give the weak peers time to develop an appropriate clientele.

We will also consider a leader that can *block* super peers from queries on all but one semantic type. That is, based on the cache contents or the traffic statistics of the super

peer, first determine that the super peer is more appropriate to serve only a semantic type τ_i. Then, it can block all the queries of the weak peers which are submitted to this super peer, but are not of type τ_i.

Super-Peer Experiments

We now describe a number of experiments using variants of the leadership strategies discussed above. Most of our experiments will show the effect of these actions by a leader that does not reason about its own cost for blocking or capping. But as a step towards performing such reasoning, we will also study the performance of a leader that uses a mathematically derived approximation of an SPN to find the best (lowest cost but still effective) bandwidth cap to use in a given time interval.

The number of super peers and weak peers that we used in our simulations are 30 and 300 respectively. We also defined 1800 files that are evenly and randomly distributed at the beginning. There are 10 semantic types defined for the files. The cache of super peers can accommodate up to 65 files, and the weak peers store attitudes for up to 10 super peers, pledging to the top 5. At the initialization, agent caches are populated with random items.

Bandwidth Capping for Better CMR

In this experiment, we show how a leader can affect the convergence of the system, leading to a better value for a global metric on the system. There are 30 super peers in the system, with 5 of them being greedy. Figure 4 (left) shows the effect of a bandwidth capping leader that caps the percentage of targeted super peers at 70% of the number of requests made to them, but *only* between steps 1000 and 2000. One can see a leadership "bump" in this graph as the leader intervenes (similar to what happened when the leader exited in our Bimatrix experiments) and the system struggles to adjust. However, the leader is able to push the system to a better equilibrium, even though it exits after only 1000 steps. This is because that intervention leads many of the niche weak peers (those with less popular semantic types) to develop bad attitudes towards the greedy super peers (who are unable to fulfill their requests). They then migrate to the altruistic super peers, who cache many of their files which are unpopular in general, but very popular within their type. This in turn limits the file loss in the system, and leads to a better CMR than if the system were run without a leader.

As empirical evidence that bandwidth capping by the leader is enforcing specialization, leading to the lower CMR, we define the *speciality* of a super peer as the fraction of files in its cache that are of the most prevalent type in the cache. We calculated the average specialty of both kinds of super peers at initialization and after convergence. For greedy super peers, specialty starts at .35 and increases to .63 without the leader but reaches .86 with the leader. A similar effect (.35 to .50 without the leader, .71 with the leader) is seen for altruistic super peers, though specialty is less pronounced there, presumably because these super peers can afford to specialize in two different niche file types. Note that the choice of the points where the leader steps in and out,

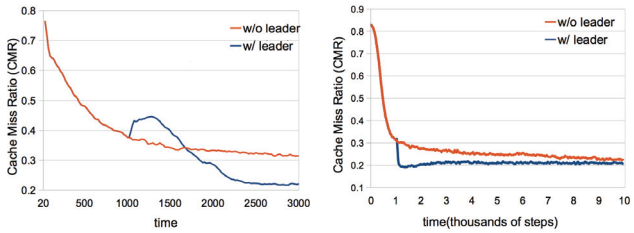

Figure 4: Left:A bandwidth capping leader's impact on the system CMR (10 runs). The capping is only done in the leader line between steps 1000 and 2000. Right: Performance of a blocking leader (stepping in at time 1000 for 80 steps) that improves convergence (10 runs) of a system with only altruistic super peers by enforcing specialization.

are arbitrary and we observed very similar results for different intervention start-points, though an intervention length of around 1000 steps seemed to be needed.

Forcing Specialization

Our next experiment shows how a leader can accelerate the convergence of the system, even if it doesn't improve the asymptotic performance. Again, this mirrors prior work in small-scale Bimatrix games. In this experiment, all the super peers are altruistic and the leader uses its "blocking" action as described earlier to force the all of the super peers to accept only requests for file types that form a plurality in their cache. Figure 4 (right) compares a system with such a leader versus a leaderless version and compares their CMRs. Again, the leader only intervenes for a short time– between steps 1000 and 1080. With this subtle intervention, there is a dramatic effect on the convergence rate– the system specializes almost immediately (and much faster than the leaderless system) and maintains a low CMR (which has a direct relation to super peers' specialization). We also explored leader interventions for shorter and longer intervals, and noticed that longer interventions did not particularly improve the convergence rate but, for interventions shorter than 80 steps, the system converges slower, but still faster than the case with no leader.

An Expectation Model for Managing Cost

In the experiments shown above, the actions of the leader were chosen by the experimenter, but ideally the leader would be an autonomous agent making decisions to maximize its own utility function. However the state space of that problem is very large due to the number of parameters in our model, and search could be prohibitively long. While one could use Monte Carlo simulation here (similar to (Hung, Kolitz, and Ozdaglar 2011)) we will instead attempt to decrease the parameter space by deriving a mathematical approximation of the SPN. This model will provide a way to choose actions *relative* to the parameterization of the system thereby decreasing the state space of the problem. We briefly describe this model below using the terms T, F, W and S for the sets of semantic types, files, weak peers and

super peers in the system and lower case like f_τ to signify, for instance, the set of files of type τ.

To consolidate terms in our equations and reach useful forms, the following assumptions were made: weak peers ask only for files of their type, super peers index only files of their type, there are equal numbers of each type of file, all files are equally popular, and attitudes are never decremented. Furthermore, in the empirical system weak peers send their request to each super peer in the top half of their cache in order of rank until the request is met or the list is exhausted. The expectation of what super peers will be in a weak peer's cache would be very difficult and inaccurate to calculate, as would their ranking. Instead, this model assumes a weak peer could ask *any* super peer at any time, and that the choice is probabilistic, such that the likelihood of any super peer being chosen is weighted by the weak peer's attitudes toward him. So we can denote the probability of weak peer i sending a request to super peer j as:

$$Pr_t(S_j|W_i) = \frac{\alpha_{ijt}}{\sum_k^S \alpha_{ikt}} \tag{1}$$

Note, this will still converge to some small set of super-peers receiving the majority of a weak peer's requests.

With these simplifications, we can derive some useful values such as the expected CMR at time t. Weak peers generate a request any time they aren't waiting on a previous request, or about every 2-6 timesteps. We will approximate this as a static requesting probability $Pr(\text{req})$. Therefore the expected number of requests each turn would be $Pr(\text{req}) \cdot |W|$. The expected number of cache misses is more involved, and depends on what file-types are being requested of whom. The end result is (after some $Pr(\text{req})$ terms cancelled):

$$\begin{aligned} CMR_t = \frac{1}{|W|} \cdot \sum_i^W \sum_\tau^T \Bigg[& Pr_t(\text{request } \tau|W_i) \\ & \cdot \Bigg(\sum_j^S Pr_t(S_j|W_i) \cdot Pr(\text{miss}|S_j, \tau) \Bigg)^{\frac{\omega}{2}} \Bigg] \end{aligned} \tag{2}$$

The summation term is the expected number of cache misses, and is essentially the sum of the likelihoods of each combination of file-type and super peer causing a miss, weighted by the expected number of times such a combination will be chosen. The summation over super peers is raised to the power $\omega/2$ where ω is the weak-peer cache size, because half the cache must fail to have the file before this is considered a cache miss. Two of the inner terms are very simple: $Pr_t(\text{request } \tau|W_i)$ is either 0 or 1 and $Pr(\text{miss}|S_j, \tau)$ is 1 or $\sigma/|f_\tau|$, dependant upon τ being of the same type as W_i or S_j respectively (where σ is the super-peer cache size).

The third inner term was defined in (1) but requires more explanation. $Pr_t(S_j|W_i)$ depends on W_i's attitudes, α_{it}, but α_{it} in turn depends on the previous probabilities of $Pr(S_j|W_i)$ because attitudes are only altered when a super peer serves a request:

$$\alpha_{ijt} = 1 + \sum_{s=0}^{t-1} \sum_\tau^T Pr_s(\text{req } \tau|W_i) \cdot Pr_s(S_j|W_i) \cdot Pr(\text{hit}|S_j, \tau) \tag{3}$$

where $Pr(\text{hit}|S_j, \tau) = 1 - Pr(\text{miss}|S_j, \tau)$. So $Pr(S_j|W_i)$ and α_{ij} are mutually recursive, and unfortunately cannot be solved for closed form. Further cancellations will give us two formulas, the probability that a weak peer requests from a super peer of its own type, abbreviated $Pr(\text{same})$:

$$Pr_t(\text{same}) = \frac{|s_\tau| + NumHits}{|S| + NumHits} \tag{4}$$

And the probability it will choose a different type:

$$Pr_t(\text{diff}) = \frac{|s_\tau|}{|S| + NumHits} \tag{5}$$

Where:

$$NumHits = \frac{B \cdot |S|}{|W| \cdot Pr(\text{req})} \cdot \frac{\sigma}{f_\tau} \cdot \sum_k^{t-1} Pr_k(\text{same}) \tag{6}$$

B here is a bandwidth cap, not shown in earlier derivations for simplicity. Equations (4) and (5) are very useful, as they can can be used to predict the number of requests a given super peer is likely to get, and how that will be affected by changing the bandwidth cap.

Bandwidth-Selecting Leader Results Now that our leader has a way to determine how much force to apply to cause a desired change in the system, we can test a bandwidth selecting leader against fixed policy leaders. We again experiment with a system containing 5 greedy super peers and a leader who caps bandwidth, this time using an absolute cap instead of a percentage. Additionally, such a bandwidth cap will be treated as incurring a cost to the leader equal to its difference from the expected number of request a super-peer would receive uncapped. In terms of utility, each leader wants to save at least 40 file pointers from being lost at minimum cost to itself. Because most file loss occurs at the beginning, the leader will intervene between times 1 and 200. We have empirically determined that approximately 40 file pointers can be saved in this time period if the requests to greedy peers are halved. We gave this as background knowledge to the leaders. Two fixed policy leaders are tested: a heavy-handed leader that and caps bandwidth at 1 query/timestep, and a light-handed leader that just minimizes its own cost by capping at 17 queries/timestep (where 18 is the expected number of requests to these super peers). The bandwidth selecting leader takes the background knowledge and uses $Pr_t(\text{same})$ and $Pr_t(\text{diff})$ to approximate the effects of applying different bandwidth caps to these peers, to determine the minimum cap that will halve greedy peer requests.

The results of this experiment show that the heavy-handed leader was indeed too forceful; it saved 62 file pointers from deletion on average, and additionally caused a large increase in CMR, all at a cost of $(18 - 1) \cdot 200 = 3400$. The light-handed leader only saved an average of 5 files, incurring a cost of $(18 - 17) \cdot 200 = 200$. The bandwidth selecting leader, meanwhile, chose a bandwidth cap of 5, thereby saving an average of 39 file pointers from deletion while only having a transitory effect on CMR and incurring a cost to himself of $(18 - 5) \cdot 200 = 2600$.

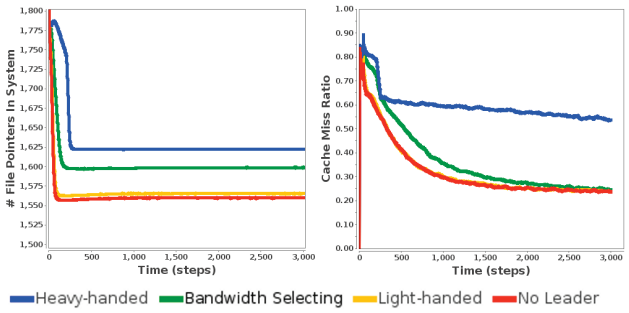

Figure 5: Unique file pointers in super-peer caches (left) and CMR (right). Averaged over 10 iterations.

One notable result of this experiment is the elevated CMR due to the heavy-handed leader's actions. This is an example of how careless or uninformed leadership can have adverse effects on the system. Our bandwidth-selecting leader used only a simple binary search and did not directly calculate the expected CMR, but with more sophisticated optimization techniques a leader could use a similar model to choose not only the bandwidth cap, but whom to cap, when to do it, and for how long.

Future Work

We have laid out a general formulation of leadership in manipulating an MAS and empirically demonstrated the potential of leaders in both toy domains and a complicated file sharing network, but there are many avenues of open research ahead. First, a better theoretical foundation in which to measure leadership is still needed, and general algorithms that work across domain types (e.g. auctions and voting systems) require further study. Empirically, our success in leading SPNs could be further studied in P2P networks and also in general *patron-client* interactions (Gellner and Waterbury 1977), which are a more general classification of the relationships in an SPN.

There are also a number of extensions to our leadership game definition that warrant further investigation. For instance, if multiple leaders are present (as is the case in many market-design settings) leadership becomes a "game on top of a game" between the two leaders for control of $\mathcal{G}$. Also, we have not explicitly reasoned about how the participants in $\mathcal{G}$ might more directly interact with or try to deceive the leader, though such actions are not prohibited by our general definition. Finally, in this work we have assumed that the leader is already aware of the intentions or at least the types of different agents in $\mathcal{G}$ (as with the greedy super peers). In general, however, players could be profiled (e.g. (Gruman and Narayana 2008)) based on statistics on their behavior, or even using some active exploration by the leader.

Conclusions

We developed a general formulation of leadership games where a leader tries to influence an MAS to reach either a better global optimum (with respect to the leader's utility function) or achieve convergence faster. We empirically demonstrated this approach in simple Bimatrix games and in a complicated P2P file sharing system where different leaders were able to induce better cache miss ratios and speed up specialization (and convergence). We also presented first results involving a leader that uses an approximation of the system to reason about the cost of its intervention.

Acknowledgements We thank Ron Breiger and Monica Vroman for helpful discussions. This work was supported by AFOSR FA9550-10-1-0569.

References

Babes, M.; de Cote, E. M.; and Littman, M. L. 2008. Social reward shaping in the prisoner's dilemma. In *AAMAS*.

Bachrach, Y.; Elkind, E.; Meir, R.; Pasechnik, D.; Zuckerman, M.; Rothe, J.; and Rosenschein, J. S. 2009. The cost of stability in coalitional games. In *International Symposium on Algorithmic Game Theory*.

Cholvi, V.; Felber, P.; and Biersack, E. 2004. Efficient search in unstructured peer-to-peer networks. *European transactions on telecommunications* 15(6):535–548.

Conitzer, V., and Sandholm, T. 2007. Incremental mechanism design. In *IJCAI*.

Faliszewski, P.; Hemaspaandra, E.; and Hemaspaandra, L. A. 2006. The complexity of bribery in elections. In *AAAI*.

Garbacki, P.; Epema, D.; and van Steen, M. 2007. Optimizing peer relationships in a super-peer network. In *Distributed Computing Systems*.

Gellner, E., and Waterbury, J. 1977. *Patrons and clients in Mediterranean societies*. Duckworth.

Gruman, M. L., and Narayana, M. 2008. Applications of classifying bidding strategies for the cat tournament. In *AAAI-08 Workshop on Trading Agent Design and Analysis*.

Guo, M., and Conitzer, V. 2010. Computationally feasible automated mechanism design: General approach and case studies. In *AAAI*.

Hung, B.; Kolitz, S.; and Ozdaglar, A. 2011. Optimization-based influencing of village social networks in a counterinsurgency. In *Social Computing, Behavioral-Cultural Modeling and Prediction*.

Littman, M. L., and Stone, P. 2002. Implicit negotiation in repeated games. In *8th International Workshop on Intelligent Agents*.

Milojicic, D.; Kalogeraki, V.; Lukose, R.; Nagaraja, K.; Pruyne, J.; Richard, B.; Rollins, S.; and Xu, Z. 2002. Peer-to-peer computing. Technical report, Hewlett Packard.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*.

Petric, A.; Podobnik, V.; Grguric, A.; and Zemljic, M. 2008. Designing an effective e-market: an overview of the cat agent. In *AAAI-08 Workshop on Trading Agent Design and Analysis*.

Watkins, C. J. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation, King's College, University of Cambridge, UK.

Yang, B., and Garcia-Molina, H. 2003. Designing a super-peer network. In *International Conference on Data Engineering*.

Yoram Bachrach, E. E., and Faliszewski, P. 2011. Coalitional voting manipulation: A game-theoretic perspective. In *IJCAI*.