

of anarchy, we define the price of monarchy, in order to quantify the practical cost of cooperation in terms of communication cost. The quantification of these two criteria provides an eloquent means for trade off analysis for coordination in MARS.

Aspiring to reduce these two costs, we then introduce the A Few Good Agents (AFGA) approach. The key idea of the proposed approach comes from reciprocal altruism. Informally, an agent will cooperate if and only if the agent believes that it would receive higher expected payoffs by taking cooperative actions than by not doing so, and the mutual dependence among agents generates such an incentive for persistent cooperation.

The AFGA approach is driven by a synergy between two types of agents: *leaders* and *voters*. A leader learns to make good predictions about the changes in the environment, and selects actions based on its predictions. When the entire population is composed of agents of a type leader, then the system pays the full price of anarchy.

A voter, alternatively, utilizes *social learning* - learning from other agents - to learn indirectly about the environment from a set of leaders, and selects actions based on its leader's predictions. The intuition for voter agents is similar to that of classical *ensemble learning* methods [10] in that a voter's decisions depend on some set of other learning agents that may use various learning algorithms. The learning algorithms of leaders are transparent to voters, hence, voters' criteria for subscribing to a leader's prediction is determined by the actual performance of the leader's past predictions.

In addition, the AFGA approach presents a beneficial supplementary property. From the perspective of multi-agent systems (MAS), it is redundant to have multiple agents learning the same information simultaneously. We design AFGA agents to utilize social learning to maximize the learning utility of the MAS. This is intuitive if social learning is relatively easier than the learning of the actual task. The name "a few good agents" refers to the subset of agents that are adept learners, namely the leaders, while the other agents - voters - take full advantage of the leaders by only needing to solve a simpler learning problem.

The use of social learning additionally provides robust performance in the case of a dynamically changing population, e.g., cars in traffic routing problems. Most existing MAL algorithms assume a finite set of agents learning at the same speed. Since learning agents tend to explore more at the beginning of the learning phase, the performance of learning algorithms degrades whenever a new set of agents is introduced to the system. The AFGA approach is better insulated from this potential learning degradation caused by dynamic changes in population because new agents can utilize social learning instead of exploring the environment themselves.

We have carried out a preliminary set of experiments in the El Farol bar problem [1], a simple example of MARS, to evaluate the efficacy of our approach. The results show that the AFGA agents significantly reduce the price of anarchy compared to a Nash equilibrium which is a targeted solution of most existing MAL algorithms. At the same time, the AFGA approach has also been shown to bound the price of monarchy significantly lower than that of a centrally administered system.

More notably, the results also demonstrate that the AFGA approach is still effective under two types of uncertainty: 1) when agents have only limited observation of the state of resources, 2) when the members of the agent population in

MARS change dynamically, e.g., traffic flow in an automobile routing problem.

2. DEFINITIONS

This section provides a formal definition of MARS, followed by preliminary definitions that are used in our discussion throughout the paper.

2.1 Multi-agent Resource Selection Problem

We formally define MARS as a quadruple of $(N, \Gamma, \vec{A}, \vec{R})$. MARS is a single state repeated game which is repeated infinitely.

- N is a set of agents. $N = \{1, 2, \dots, n\}$.
- $\Gamma = \{r_1, \dots, r_m\}$ denotes a set of resources available for agents in N .
- $\vec{A}_t = a_1 \times \dots \times a_n$ denotes the resource choices of the agents at time t where $a_i \in \Gamma, \forall i \in N$.
- $\vec{R}_{t+1} : \Gamma \times \vec{A}_t \rightarrow \mathfrak{R}$ is a delayed reward function

Specifically, a reward associated with using a resource is defined as a function of the number of concurrent users of the resource, and all users using the same resource share the same reward. Thus, MARS is a class of congestion games [20].

2.2 Selfish Equilibria

One of the main criteria of evaluating a MAL algorithm in MARS is convergence. The following two equilibria are often discussed as the target of convergence.

- I. Nash equilibrium (NE) [18]: A joint strategy profile π is in a NE iff no player wants to deviate from the current choice of action given the opponents' actions are fixed.
- II. Correlated equilibrium (CE) [2]: Given a joint strategy profile π , let π_i, π_{-i} denote the action prescribed for agent i by the strategy π , and the vector of actions prescribed by the strategy π for agents $j \in N, j \neq i$. When a common prior assumption holds, i.e., when all agents have access to the joint distribution of actions of all agents, a correlated equilibrium is a strategy profile s s.t. all players are Bayes rational, i.e., $\forall i \in N, E[r_i(\pi_i, \pi_{-i})] \geq E[r_i(a_i, \pi_{-i})] \forall a_i \in A_i$ where A_i denote a set of available actions of agent i .

In fact, a NE is a special class of a correlated equilibrium where agents' decision making (or the information on which an agent's decision depend) is independent from one another.

2.3 Centrally Administered System (CAS)

A centrally administered system (CAS) is a model where a global administrator has access to complete information. An administrator also communicates with all agents in the system, optimizing the performance of the entire population.

3. CRITERIA FOR COOPERATIVE MAL

We propose the use of two criteria to evaluate cooperative MAL algorithms. The quantification of these two values provides important criteria for coordination decisions. Knowingly, agents should coordinate with others if and only if the coordinated actions reduce the price of anarchy while bounding the price of monarchy by a proper level.

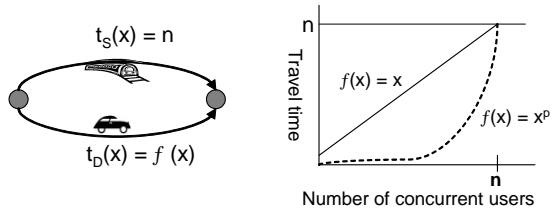


Figure 1: Driving versus Metro

3.1 Price of Anarchy

The price of anarchy, which was first introduced in [16], measures the inefficiency of a selfish equilibrium. Let φ_a and SE denote the objective function value of a target problem using an algorithm a , and a set of selfish equilibria, respectively. In the price of anarchy literature, it is conventionally assumed that the objective is to *minimize* the cost function φ . Let φ_s and φ_{opt} denote the objective function value of a selfish equilibrium s , $s \in SE$, and that of the optimal solution, respectively. The price of anarchy, $\$_{worst}^A$, is defined as $\max_{s \in SE} (\frac{\varphi_s}{\varphi_{opt}})$. $\$_{best}^A$ is defined similarly. In the original work, the price is computed using the worst NE. Similar work on correlated equilibria can be found in [7].

In this paper, we generalize the definition such that the price of anarchy measures the inefficiency of a MAL algorithm. Thus, the price of anarchy of a learning algorithm l is

$$\$1^A = \frac{\varphi_l}{\varphi_{opt}}. \quad (1)$$

3.2 Price of monarchy

Analogous to the price of anarchy, we introduce a new measure, price of monarchy. Whereas the price of anarchy measures potential quality loss due to selfish decisions, the price of monarchy estimates the practical cost of installing cooperation in MAS. We mainly discuss time delay during execution, thus we define the price of monarchy in terms of communication cost.

Thus, the lower bound of the price of monarchy is found in non-communicating systems. In general, the price of monarchy depends on how a coordination mechanism is implemented, e.g., vigorous negotiation may require iterative communication processes. We set the upper bound of the price of monarchy to that of a CAS, and disregard algorithms for which the communication cost exceeds this upper bound.

Let ς_l and ς_{-c} denote a communication cost function of a learning algorithm l , and that of a non-cooperative system, respectively. The price of monarchy, $\$1^M$, is

$$\$1^M = \frac{\varsigma_l}{\varsigma_{-c}}. \quad (2)$$

4. SOME BASIC MARS PROBLEMS

In this section, we use two illustrative examples to describe the main issues raised in MARS, namely, 1) the price of anarchy in selfish routing, and 2) the rationality paradox.

4.1 Driving versus Metro

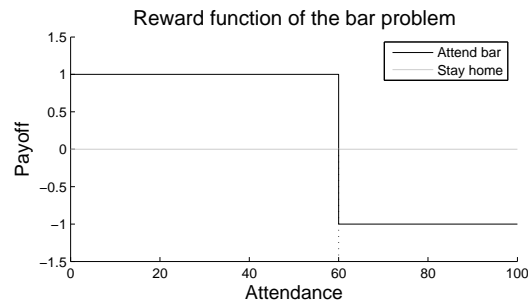


Figure 2: Reward function of EFBP ($n = 100, \tau = 60$)

Let us first illustrate selfish routing using a simple example of MARS [19, 22]. Suppose there exist n self-interested agents that are deciding between two actions of taking a metro or driving to work, denoted by M and D , respectively. Figure 1 illustrates such an example. The natural objective of this problem is to minimize the average travel time of all agents.

Let x_a denote the number of agents that selected action a , where $a \in \{M, D\}$, and let $t_a(x_a)$ be the travel time of taking action a . Note that the travel time is a function of x_a , and the agents that selected the same action all experience the same travel time. Let us assume that $t_M(x_M) = n$ where n is the number of agents. On the other hand, let the travel time of driving be a linearly increasing function, $t_D(x_D) = x_D$ such that $t_D(n) = n$. That is, taking a metro takes a constant travel time that is always slower than driving except when the traffic is fully congested.

In this case, self-interested agents converge to an equilibrium in which agents will always choose to drive even when the road is fully congested, resulting the average travel time of n . This is a NE since no one is motivated to deviate from their current choice of actions given the choices of other agents are fixed.

If we assume that there exists a CAS that selects a small number of agents, ϵ , ($\epsilon < n$), and forces them to take the metro, then the travel time of the selected agents is still not any slower than that of their original decisions, i.e., n . This enforcement, however, enables the remaining drivers to travel faster, reducing the travel time of a driving agent to $n - \epsilon$.

In this case, the average travel time of all agents is a convex function, $\frac{1}{n} \times \{\epsilon n + (n - \epsilon)^2\}$. By taking the derivative of the convex function, an optimal value is trivially found, e.g., $\epsilon = \frac{n}{2}$, reducing the average travel time down to $\frac{3n}{4}$.

Suppose instead that the travel time function is non-linear (dotted line in Figure 1), e.g., an exponential function, $t_D(x_D) = x^P$, for some P . Then, in the limit, the travel time of driving becomes ignorable, i.e., $\lim_{P \rightarrow \infty} (n - \epsilon)^P = \lim_{P \rightarrow \infty} \{n^P (1 - \frac{\epsilon}{n})^P\} = 0$. Thus, the average travel time of agents in this case is reduced to ϵ .

In this example, the price of anarchy is $\frac{4}{3}$ and $\frac{n}{\epsilon}$ for the linear travel time function and the exponential function, respectively. This example provides an interesting observation: although the price of anarchy can be very high, e.g., exponential cost functions, the price can be significantly reduced by a small number of altruistic agents.

4.2 The El Farol Bar Problem (EFBP)

The El Farol bar problem (EFBP) introduced in [1] is

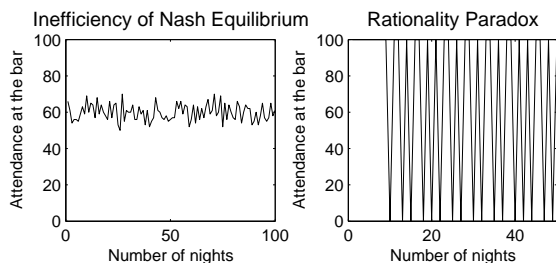


Figure 3: Selfish Equilibria and Rationality Paradox

another example of MARS. EFBP is defined as follows. A set of n agents repeatedly make decisions of whether to attend a bar or not on certain nights. The only observations available to the agents are the past history of attendance at the bar. In the original problem, it was assumed that agents have access to a complete history of attendance.

The payoff of attending a bar is high only if the number of attendees at the bar on the night is less than a certain threshold, τ . However, the agent receives the worst payoff if the bar is over crowded. Thus, an agent is better off staying home if it believes that the bar would be crowded on the night. An example of the reward function for $n = 100, \tau = 60$ is shown in Figure 2.

Despite the simplicity, EFBP clearly exhibits the two important issues of MARS: the *rationality paradox* and *selfish equilibria*. The rationality paradox refers to the fact that a rational agent fails to learn the best action based on its expected reward. Since all agents are simultaneously learning the same information, agents reason in the same manner.

For instance, when an agent predicts the attendance at the bar is lower than τ then the agent decides to attend the bar. Since the other agents also reason the same manner, however, the entire population decides to attend the bar, receiving the worst payoffs. Figure 3 (right) depicts such a result. Agents face contradicting outcomes by making decisions based on their rationality.

Existing studies on EFBP have been mostly focused on the issue of the rationality paradox, seeking algorithms that converge to selfish equilibria. For instance, agents using an inductive reasoning algorithm converge to a mixed strategy NE [1]. Figure 3 (left) depicts a mixed strategy NE of EFBP. Alternatively, agents using a regret-based learning algorithm converge to a set of correlated equilibria [12].

5. A FEW GOOD AGENTS MODEL

In general, some agents can learn better than others at certain times either because they are exposed to different parts of information in the environment, or because they simply have better learning algorithms. Based on this observation, we propose a multi-agent model in which a set of such privileged agents learn directly from the environment while the rest utilize social learning, i.e., they learn rather indirectly from those privileged agents.

We first define two types of agents: *leader* and *voter*. A *leader* agent learns to choose actions for a group of one (itself) or more agents. Instead of learning a policy for itself, an agent may depend on the strategies of other agents. If an agent is following the strategy of another agent it is a *voter*. Thus, a voter agent tries to learn whose strategy yields the highest payoffs.

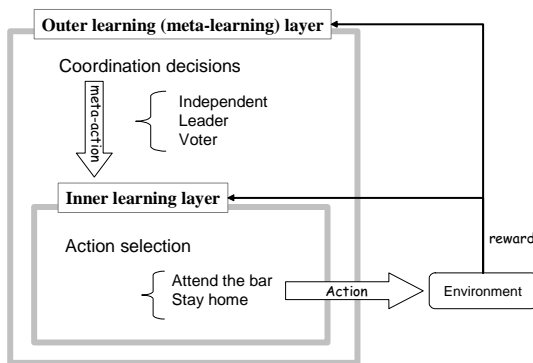


Figure 4: Hierarchical learning layers

An agent can change its type between leader and voter. When an agent is changing its type, it is said that the agent *mutates*. The mutation succeeds with some probability.

Let $type(i)$ denote the type of agent i , $i \in N$. In this context, we hypothesize the existence of a set of leaders at time t , α_t , that can lead the population to perform better than any other set of leaders. Formally,

Let N be a set of self-interested agents competing for scarce resources, then there exists a set of leaders, $\alpha_t, \alpha'_t \subseteq N$ at time t , s.t. $\varphi_t(N, \alpha_t) \leq \varphi_t(N, \alpha'), \forall \alpha' \subseteq N$, where $\varphi_t(N, \alpha')$ denotes the quality loss (in resource utilization) at time t , and $\forall j \in N, j \in \alpha'$ iff $type(j) = L$.

That being said, we aim to design learning algorithms for the agent population of MARS to learn to elect the ideal set of leaders themselves.

The crux of the AFGA approach lies in mutual dependence between leaders and voters. On one hand, a leader needs its voters to execute collective actions to achieve a higher payoff, thus it is motivated to be truthful to the voters in order to retain its voting bloc. Since there are multiple leaders, a voter can switch to a new leader if the performance of the current leader is no longer satisfying.

On the other hand, a voter elects to listen to the leader because it believes that acting in the group will produce higher rewards than acting individually. Thus, the AFGA approach is based on reciprocal altruism, which is indeed a class of a selfish model.

The AFGA model is composed of two hierarchical layers as depicted in Figure 4. In general, the goal of reinforcement learning is to learn a policy that maps a state onto an action to maximize the expected future reward over time. The learning of a policy to determine an action, e.g., whether to attend a bar or stay home, is done in the *inner* learning layer. The *outer* layer is where agents learn to choose a coordination action that maximizes the reward. The set of decision choices in the *outer* layer are called *meta-actions*.

As shown in Figure 4, an agent makes decisions in both layers each night, and both the selected meta-action and the ultimate action (attend the bar or stay home) are evaluated according to the same reward received from the environment.

In what follows, we will describe the learning occurring in each layer in detail. Although we will use the bar problem

as an example to illustrate the algorithm, the algorithm is designed for a more general class of a MARS problem.

5.1 Outer learning (meta-learning) layer

In multi-agent systems, an agent needs to decide whether it should act independently or coordinate its actions with other agents in an environment to maximize its performance. The most innovative part of the AFGA approach is representing such coordination decisions as a *meta-learning* layer in the agent’s reasoning model.

For instance, an agent learns to make a better decision on whether it should be a *leader* or a *voter* in the outer layer. Alternatively, an agent may still decide to act independently.

The meta-actions representing such choices are denoted by a set: $\{L, I, \alpha_1, \dots, \alpha_k\}$. Meta-action L means that the agent is a leader itself whereas meta-action I indicates that the agent is an *independent* voter. An independent voter uses its own policy to select actions. Although a leader without subscribers can be also viewed as an independent agent, it is not included in the set of meta-actions because a leader itself cannot *choose* to become an independent leader in the current model.

Other meta-actions, e.g., α_i , denote candidate leaders. For example, if an agent’s meta-action is α_i then the agent is a voter following the strategy of another agent α_i . The maximum number of candidate leaders that one voter can keep in its memory is parameterized by k (default value for k is 4 in the experiments below). Thus, each agent has $k + 2$ meta-actions.

Agents learn a policy in the outer layer using a Q-learning algorithm that was simplified for a stateless repeated game [8]. After each night, an agent updates a Q^M -value for its current meta-action. Suppose α_i is the current meta-action. Let r_t be the reward at time t after following the prescription of the current leader α_i . Note that the reward is only defined for the actual action that was taken, e.g., attend the bar, as opposed to meta-actions. The Q^M -value of α_i is then updated as follows.

$$Q_{t+1}^M(\alpha_i) = (1 - \eta)Q_t^M(\alpha_i) + \eta r_t \quad (3)$$

where η is the learning rate such that $0 < \eta \leq 1$.

An agent then chooses a meta-action using an ϵ -greedy method, i.e., an agent selects the best meta-action based on the Q^M -values most of time, but it explores other options with a small probability, ϵ .

For instance, a leader agent mutates to a voter if Q^M -value of meta-action L , $Q^M(L)$, is no longer the highest. Conversely, a voter agent mutates to a leader if its $Q^M(L)$ is the highest. With a small probability ϵ , an agent randomly chooses a meta-action rather than the action with the maximum Q^M -value.

5.2 Inner learning layer

In the inner layer, an agent decides the actual action, e.g., whether to attend the bar or not. In fact, every agent has its own stochastic policy π for choosing an action. Initially, the policy of an agent is a random choice among all available actions. The policy is updated when the agent is a leader, and the policy remains stationary when it is a voter.

Depending on its current meta-action choice, an agent may follow its own policy or that of another agent. The strategy of a voter following a leader is straightforward. A voter queries its leader, and the leader uses its policy to prescribe an action for the voter. The voter then simply follows the action that its leader has just prescribed. When

a voter is independent, on the other hand, it uses its own (stationary) policy to make a decision.

A more interesting question is how a leader updates its policy. A typical reinforcement learner in the bar problem may try alternative actions and update the value of the two actions - attending the bar or staying home - in order to learn a policy, e.g., *attend* the bar with some probability p .

Instead of updating the value of an individual action of attending the bar, a leader agent updates the expected attendance at the bar. Because the reward in congestion games is defined as a function of the number of the agents that have selected the same action (resource), the expected attendance is in fact the prediction of *joint* actions of the agent population.

Let n denote the number of agents that are deciding to attend the bar, and let τ be the maximum attendance such that agents receive the worst payoffs if the attendance at the bar exceeds τ . The goal of a leader is to regulate the joint actions of its subscribers lest the total attendance exceed the threshold τ assuming the rest of the population is acting selfishly.

Let $l_i(t)$, and $\bar{l}_i(t)$ denote agent i ’s observation of the actual attendance of the bar on the t^{th} night, and agent i ’s prediction for the expected attendance on the t^{th} night, respectively. Then the update function is

$$\bar{l}_i(t + 1) = (1 - \eta)\bar{l}_i(t) + \eta l_i(t) \quad (4)$$

where η is the learning rate such that $0 < \eta \leq 1$.

Based on the expected attendance, a leader further estimates an admissible number of agents among its subscribers that can attend the bar. Since the expected attendance represents the joint actions of the agent population on the next night, the predicted attendance can be interpreted as the sum of two numbers: 1) the number of agents that the leader i is going to send among its subscribers (voters), denoted by $c_{i,t}$, and 2) the number of agents that decide to attend the bar independent of the leader i , denoted by $c_{-i,t}$.

Let $\Omega_{i,t}$ denote a set of voters that subscribe to the leader i ’s strategy at time t . The length of Ω_i is at least 1 since Ω_i includes leader i itself. An admissible number of agents that can attend on the following night, $c_{i,t+1}$, is estimated by subtracting the load that would be generated by non-subscribers from the predicted attendance $\bar{l}(t)$ as follows.

$$c_{-i,t+1} = \max\{\tau, \bar{l}(t)\} \times \left(1 - \frac{|\Omega_{i,t}|}{n}\right) \quad (5)$$

$$c_{i,t+1} = \max(0, \tau - c_{-i,t+1}) \quad (6)$$

Finally, the policy π is updated as follows. Let $p_{i,t}$ denote the probability that a subscriber of leader i at time t attends the bar.

$$p_{i,t} = \frac{c_{i,t+1}}{|\Omega_{i,t}|} \quad (7)$$

$$\pi_i = (1 - \eta)\pi_i + \eta p_{i,t}$$

In other words, policy π_i is the probability that leader i prescribes action *attend* for its subscribers. Since the decision is randomized by policy π , all subscribers receive a fair chance to attend the bar. A leader also counts the number of subscribers for which it has prescribed action *attend* so that the number of attendees among its subscribers at time t is at most $c_{i,t}$ - the admissible number of attendees.

6. EXPERIMENTS

