Relationship Design for Socially-Aware Behavior in Static Games

Shenghui Chen¹, Yigit E. Bayiz¹, David Fridovich-Keil¹, Ufuk Topcu¹

¹University of Texas at Austin.

Contributing authors: shenghui.chen@utexas.edu; egebayiz@utexas.com; dfk@utexas.com; utopcu@utexas.com;

Abstract

Autonomous agents can adopt socially-aware behaviors to reduce social costs, mimicking the way animals interact in nature and humans in society. We present a new approach to model socially-aware decision-making that includes two key elements: bounded rationality and inter-agent relationships. We capture the inter-agent relationships by introducing a novel model called a relationship game and encode agents' bounded rationality using quantal response equilibria. For each relationship game, we define a social cost function and formulate a mechanism design problem to optimize weights for relationships that minimize social cost at the equilibrium. We address the multiplicity of equilibria by presenting the problem in two forms: Min-Max and Min-Min, aimed respectively at minimization of the highest and lowest social costs in the equilibria. We compute the quantal response equilibrium by solving a least-squares problem defined with its Karush-Kuhn-Tucker conditions, and propose two projected gradient descent algorithms to solve the mechanism design problems. Numerical results, including two-lane congestion and congestion with an ambulance, confirm that these algorithms consistently reach the equilibrium with the intended social costs.

Keywords: Game theory, mechanism design

1 Introduction

From the smallest insects to the largest mammals, natural agents demonstrate a remarkable capacity for socially-aware decision-making, resulting in behaviors ranging
from competition to altruism. For example, bees exhibit highly cooperative and altruistic behavior in colonies, from foraging for resources to prioritizing the bee queen, because they act in the interest of the hive’s collective needs instead of individual gains [1].

As autonomous agents become more pervasive, we expect them to be capable of co-existing with humans, creating a need for them to exhibit a similar kind of social awareness. Socially-aware decision-making has the added benefit of potentially improving the efficiency of the system. Consider a flow of traffic with autonomous vehicles navigating alongside human-operated ones, which may make unpredictable or less-than-optimal choices. Purely rational autonomous vehicles, lacking in social awareness, would struggle to reduce traffic congestion, as this task requires them to take into account the complex web of social relationships between each other and human-operated vehicles.

General-sum static games offer a mathematical formalism to capture the interaction among multiple agents. In these games, agents are typically modeled as choosing Nash equilibrium strategies where no agent can reduce their cost by changing their strategy unilaterally [2]. However, real-world scenarios often involve bounded rationality, where agents do not always choose optimal strategies. In such scenarios, instead of Nash equilibria, we model agent responses with quantal response equilibria, where strategies are probabilistically chosen based on potential costs [3].

Besides modeling the bounded rationality of agents, socially-aware decision-making also involves the consideration of social relationships. Humans are able to trust each other and behave cooperatively without much training, achieving higher efficiency because they are guided by ethical principles [4]. One important ethical principle is altruism—acting in a way that cares about and benefits others, but how do people calibrate how much they should care about other individuals in society? Kleiman-Weiner et al. [5] propose that the amount of care over specific people should be determined by abstract relationships. For example, people usually care more about others in their family than strangers. We hypothesize that autonomous agents can similarly attain socially-aware behaviors by introducing relationships among them. These social relationships, integral to this framework, can be effectively modeled as graphs. By representing agents as nodes and their relationships as edges, we can map out the possible social interactions among agents. This graph-based approach enables us to model how much each agent should take the costs of other agents into account.

In this paper, we present a novel approach to model socially-aware decision-making within a static game framework. Social scenarios often have inherent symmetries that dictate the range of relationships that can exist between agents. For example in a traffic setting, a regular car has a different relationship to an ambulance compared to another regular car. We associate each of these relationships with an adjacency matrix. Assigning weights to each relationship transforms the game’s structure, resulting in a game where an agent’s cost is influenced by the costs of others they are related to.

Our aim in this paper is to find the optimal weight assignment that obtains minimal social cost. Due to the multiplicity of quantal response equilibria, we formulate this goal via two bi-level optimization problems: Min-Max and Min-Min. We then propose
two projected gradient descent algorithms to solve these two problems and empirically validate the algorithms on two different congestion game scenarios.

2 Related Work

Altruism in Games: Cultivating altruistic behavior in static games is often challenging as agents face a dilemma where minimizing social costs does not align with optimizing their individual cost function. This dichotomy has led to the definitions of price of anarchy and price of stability [6]. The price of anarchy refers to the social cost ratio between taking optimal actions and the Nash equilibrium with maximal social cost. Conversely, the price of stability is the social cost ratio between optimal actions and the Nash equilibrium with minimal social cost. These metrics are by far the most commonly used ones for analyzing the cost of selfish behavior in the overall game performance. Several papers have proposed tight bounds on the price of anarchy in the context of atomic congestion games and cost-sharing games [7–9]. There are also works that investigate the price of anarchy [10, 11] and the price of stability [11] in weighted congestion games. In this paper, we focus on the problem of finding the optimal cost-sharing mechanism in static games, and propose parameter optimization methods to minimize the price of anarchy and the price of stability by encouraging cost-sharing between agents. We show the performance of these methods on weighted congestion games.

Differentiable Optimization: In this paper we optimize the game parameters based on the game solution they induce. Unfortunately, both Nash equilibria and quantal response equilibria of static games often are not expressable as closed-form formulae of the game parameters. Therefore, finding optimal parameters requires solving a bi-level optimization problem, which involves computing the directional derivatives of the equilibria with respect to game parameters by differentiating the nonlinear program that characterizes the game equilibria. Several papers have studied differentiation through the nonlinear program. Gould et al. [12] describe the general techniques of differentiating possibly nonlinear optimization problems, but these optimization problems do not allow inequality constraints. There are also implicit-differentiation-based methods that express the equilibrium as an implicit function of game parameters using KKT matrices and then use matrix calculus [13] to derive the gradients [14–16]. More recently, these differentiable optimization methods appear in the context of game theory as well, both for differentiating through Nash equilibria [17, 18], and quantal response equilibria [19].

3 Preliminaries

3.1 Static Games

A static game \( G = (N, S, c) \) is defined by three key elements. \( N = \{1, \ldots, n\} \) is a set of \( n \) players. For each player \( i \in N \), there is a finite set of pure strategies, or actions, \( S^i \) available. The set of all strategy profiles \( S = \prod_{i \in N} S^i \) is the Cartesian product of pure strategies for all players. Each player \( i \) has a cost function \( c^i : S \to \mathbb{R} \), which
assigns a real number as a cost to each strategy profile. The tuple of these functions for all players is represented as $c = (c_i)_{i \in N}$.

A mixed strategy for player $i$ is a probability distribution over their set of pure strategies $S_i$, denoted as $x_i \in \Delta_{|S_i| - 1}$, where $x_i(s)$ is the probability of choosing strategy $s \in S_i$. Assuming elements in $S_i$ are indexed from 1 to $|S_i|$, each $x_i$ can be represented as a non-negative vector in $\mathbb{R}^{|S_i|}$, where $1^\top x_i = 1$. A mixed strategy profile $x = (x^1, \ldots, x^n)$ is the tuple of mixed strategies of all $n$ players. For convenience, we introduce the notation $s^{-i} := (s^1, \ldots, s^{i-1}, s^{i+1}, \ldots s^n)$ as the indexed set of pure strategies of all players except player $i$. Similarly, $x^{-i} := (x^1, \ldots, x^{i-1}, x^{i+1}, \ldots, x^n)$ refers to the strategy profiles of all players other than player $i$, which allows us to represent the mixed strategy profile for player $i$ in the context of others as $x = (x^i, x^{-i})$. Given a mixed strategy $x$ and the $i$'th player’s cost function $c_i$, the expected cost of player $i$ can be written as

$$J(x, c_i) := \mathbb{E}_{s_i \sim x_i, \ldots, s_n \sim x_n} \left[ c_i(s^1, \ldots, s^n) \right]$$

$$= \sum_{s^1 \in S^1, \ldots, s^n \in S^n} \left( \prod_{i \in N} x_i(s^i) \right) c_i(s^1, \ldots, s^n)$$

This function is important as the objective functions of the optimization for both the forward game and the relationship design problems are in this form.

### 3.2 Nash Equilibrium and Quantal Response Equilibrium

The concept of Nash equilibrium (NE) refers to a set of strategies where no player can benefit by unilaterally changing their strategy, given the strategies of the other players. Formally, in a static game $G_c = (N, S, c)$, a mixed strategy profile $x = (x^i, x^{-i})$ is a Nash equilibrium if each player $i \in N$ chooses the optimal strategy $x_i$ given the strategies $x^{-i}$ of the other players, i.e., $x = \text{NE}(G_c)$ if and only if for all $i \in N$,

$$\min_{x^i} J((x^i, x^{-i}), c_i)$$

$$\text{s.t.} \quad 1^\top x_i = 1$$

$$x_i \geq 0$$

The quantal response equilibrium (QRE) concept builds on top of the optimality conditions for Nash in eq. (2) with an additional term of entropy for each player, i.e., $x = \text{QRE}(G_c)$ if and only if for all $i \in N$,

$$\min_{x^i} J(x, c_i) - \lambda H(x^i)$$

$$\text{s.t.} \quad 1^\top x_i = 1$$

Notice the non-negative constraints in eq. (2) are redundant since the logarithm function in the entropy term implies that $x^i \geq 0$. 

4
After rearranging we get for all $i \in N$,

$$
\begin{align*}
\min_{x^i} & \quad -\left(-\frac{1}{\lambda} J(x^{-i}, c^i)\right)^{\top} x^i - H(x^i) \\
\text{s.t.} & \quad \mathbb{1}^{\top} x^i = 1,
\end{align*}
$$

where we use the shorthand $J(x^{-i}, c^i)_a$ as a vector whose $a$’th entry is given by,

$$J(x^{-i}, c^i)_a = \mathbb{E}_{s \sim x^{-i}} \left[ c^i(s^{-i}) | s^{-i} = a \right].$$

Intuitively, $J(x^{-i}, c^i)$ denotes the vector of expected costs of player $i$ taking each possible action $a$ while all other players following the strategy profile $x^{-i}$.

Based on Theorem 4 in the thesis [15] by Amos, the solution for this optimization problem is in the form of a softmax function, where boundedly rational players choose strategies with probabilities proportional to their costs, i.e., $x = \text{QRE}(G_c)$ if and only if for all $i \in N$,

$$x^i = \text{softmax} \left( -\frac{1}{\lambda} J(x^{-i}, c^i) \right),$$

where

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}. (7)$$

Inspired by [19], we approximately compute the quantal response equilibrium by solving the following nonlinear least-squares problem subject to the constraint that players’ strategies need to be proper probability distributions.

$$\begin{align*}
\min_{x=(x^1, \ldots, x^n)} & \quad \sum_{i \in N} \left\| x^i - \text{softmax} \left( -\frac{1}{\lambda} J(x^{-i}, c^i) \right) \right\|^2 \\
\text{s.t.} & \quad \mathbb{1}^{\top} x^i = 1, \quad \forall i \in N
\end{align*}$$

Essentially this optimization identifies a joint strategy for all players that maximally satisfies the quantal response equilibrium condition in eq. (6).

### 3.3 Weighted Graphs

In the study of strategic interactions among multiple players, graphs can represent the complex existing relationships among players. Each graph is typically represented as an adjacency matrix $\phi \in \mathbb{R}^{n \times n}$, where $n$ is the number of players and each matrix entry $\phi_{ij} \in \{0, 1\}$ indicates the presence of relationships from player $i$ to player $j$. In scenarios involving multiple types of relationships, a superposition of such graphs effectively combines several adjacency matrices to create a comprehensive representation of all relationship dynamics. Assigning weights $w_r$ on each adjacency matrix $\phi_r$, we get the final combined network as $\sum_{r=1}^{k} w_r \phi_r \in \mathbb{R}^{n \times n}$.
4 Relationship Weight Design Problems

We introduce a novel model called a relationship game to capture the inter-agent relationships within a static game framework.

**Definition 1** (Relationship Game). A relationship game is an augmented static game \( G_c = (N, S, c, \Phi) \) that, in addition to the usual static game structure, contains an indexed set of relationships \( \Phi = (\phi_1, \phi_2, \ldots, \phi_k) \) where each relationship \( \phi_i \in \mathbb{R}^{n \times n} \) is an adjacency matrix representing a directed graph. A relationship game \( G_c \) together with a weight assignment \( w \in \mathbb{R} \) induces a static game \( \tilde{G}_c = (N, S, \tilde{c}) \), where the cost functions \( \tilde{c} = (\tilde{c}_i)_{i \in N} \) are altered by modification functions \( M(c, \Phi, w) \) as follows,

\[
\tilde{c}_i(s) = M(c, \Phi, w) = \sum_{j \in N} \left( \sum_{r=1}^{k} w_r \phi_r \right)_{ij} \forall s \in S.
\] (9)

Intuitively, \( \tilde{c}_i \) is the weighted superposition of all neighbors of player \( i \) in all possible relationship graphs. For convenience of reference, we write \( \tilde{c} = M(c, \Phi, w) \).

We specify the socially desirable behavior through a social cost function \( V : S \to \mathbb{R} \) that assigns real numbers to strategy profiles. Given a relationship game and a social cost function, the design problem aims to find a relationship weight vector \( w \) across relationship networks in \( \Phi \) such that the quantal response equilibrium \( x^* \) of the induced game \( \tilde{G}_c \) minimizes expected social cost \( J(x^*, V) \). This goal may not be well-defined if the game \( \tilde{G}_c \) has multiple quantal response equilibria, in which case there can be more than one possible value for \( J(x^*, V) \). Thus, to ensure the problem is well-defined, we require a method to specify the equilibrium \( x^* \) of \( \tilde{G}_c \) on which to carry out the minimization of \( J(x^*, V) \). We propose two methods to do this. The first is to minimize the maximal attainable cost across all quantal response equilibria of \( \tilde{G}_c \), and the second is to minimize the minimal attainable cost. Optimization problems (10) and (11) are the respective problems resulting from these two methods.

For both problems, the primary objective is to minimize a social cost function associated with a game equilibrium. However, as illustrated in Figure 2, multiple equilibria may emerge as the weight parameter \( w \) varies. The inner maximization/minimization thus selects an equilibrium, parameterized by \( \sigma \), that has the highest/lowest social cost. We define \( \sigma \) as a real number parameter between \([0, 1]\) and \( \text{QRE}(\tilde{G}_c, \sigma) \) is a surjective function that maps the entire range \([0, 1]\) onto the set of all quantal response equilibria of \( \tilde{G}_c \).

**Problem 1** (Relationship Weight Vector Design—Min-Max Form). Given a relationship game \( G_c = (N, S, c, \Phi) \) and a social cost function \( V \),

\[
\min_{w} \max_{\sigma} J(x, V) \quad \text{s.t.} \quad \tilde{c} = M(c, \Phi, w) \quad x = \text{QRE}(\tilde{G}_c, \sigma)
\] (10)
**Problem 2** (Relationship Weight Vector Design—Min-Min Form). Given a relationship game $G_c = (N, S, c, \Phi)$ and a social cost function $V$, \[ \min_{x,\sigma} J(x, V) \quad \text{s.t.} \quad \tilde{c} = M(c, \Phi, w) \quad x = \text{QRE}(G_{\tilde{c}}, \sigma) \] (11)

Here, Problem 1 minimizes the maximal cost across all equilibria (among the set of equilibria marked in orange in Figure 2). Conversely, Problem 2 minimizes the minimal cost across all equilibria which are marked in blue in the same figure. These two goals equate to minimizing the price of anarchy and the price of stability, respectively.

We explore the problems’ relevance to two distinct scenarios of socially desirable behavior in the context of congestion games:

1. **Two-lane Congestion**: This scenario involves a congestion game on a two-lane road. Each lane’s delay is proportional to its vehicle count. The social cost function, representing total congestion time, is the aggregate of individual delay times. The objective is to minimize overall congestion time.

2. **Congestion with Ambulance**: Similar to the two-lane case, but with an ambulance granted higher priority. The social cost function, a weighted sum of delays, prioritizes the ambulance. The goal is to facilitate ambulance passage, even if it increases delay for others.

![Descriptive illustrations for the two congestion game scenarios.](image)

**Fig. 1** Descriptive illustrations for the two congestion game scenarios.

![Multiplicity of equilibria in congestion with ambulance](image)

**Fig. 2** Multiplicity of equilibria in congestion with ambulance with $\lambda = 1.0$ and $w = [0.5, 0.5, w_3, 0.5]$, where $w_3$ is sampled in the range $0 \leq w_3 \leq 3$ with increments of 0.01. At each $w_3$ value, we test 300 different seeds with a tolerance threshold of 0.0001 to identify unique values.

### 5 Gradient-based Mechanism Design

We outline how to solve the mechanism design problems 1 and 2 with a gradient-based optimization approach, which first requires a game solver to compute the quantal response equilibrium. We implement the game solver by specifying the least-squares problem (8) in Julia [20] using the JuMP [21] interface and the COIN-OR IPOPT [22]
optimizer. Furthermore, we reroll seeds to explore the parameter space of \( \sigma \). Running with a different seed randomly samples a different value of parameter \( \sigma \), which sets a different random initial joint strategy vector \( x \), possibly leading to a different quantal response equilibrium. Since both the cost function \( \tilde{c} \) in the game and a parameter \( \sigma \) selected by a seed determines a game solution \( x \), we succinctly represent this process as \( x = \text{QRE}(G_{\tilde{c}}, \sigma) \).

We now describe two tailored projected gradient descent algorithms, Min-Max and Min-Min, to solve the mechanism design problems 1 and 2, respectively. Both algorithms share a common set of inputs: a relationship game \( G_c = (N, S, c, \Phi) \), a social cost function \( V \), a step size \( \alpha \), a convergence threshold \( \beta \), and a reroll number \( L \). Both aim to determine an optimal relationship weight vector \( w \), though with different objectives:

- Min-Max identifies a \( w \) that provides the lowest possible upper bound on \( J(x, V) \);
- Min-Min seeks the lowest possible value of \( J(x, V) \) associated with \( w \).

Gradient Descent. Recall from Problem 1 we define the objective function, \( J(x, V) \), in terms of the social cost for the modified game solution, where \( x = \text{QRE}(G_{\tilde{c}}, \sigma) \) and \( \tilde{c} = M(c, \Phi, w) \). Since \( c, \Phi, V \) are given parameters in this problem, the objective function is parameterized only by a relationship weight vector \( w \) and a seed-selected value \( \sigma \). Hence we simplify the notation to express the social cost as a function of \( w \) and \( \sigma \), denoted by \( J(w, \sigma) \). Given the step size \( \alpha \), we update the weight vector as

\[
    w = w - \alpha \nabla_w J(w, \sigma).
\]

We refer the readers to Appendix A for the derivation of \( \nabla_w J(w, \sigma) \), the gradient of \( J(w, \sigma) \) with respect to \( w \).

\( L_2 \) Projection. After the gradient step, we project the output \( \|w\| \) onto the unit sphere as follows.

\[
    \text{Proj}_{L_2}(w) = \frac{w}{\|w\|}.
\]

This projection enforces the constraint in eq. (3). Furthermore, we employ this projection for two reasons. Firstly, it allows us to constrain the range of \( w \) and prevent them from diverging to infinity, which allows the gradient descent to converge faster. Secondly, constraining \( \|w\| \) prevents \( J(w, \sigma) \) from taking arbitrarily small or large values, which can cause the bounded rationality term \( \lambda H(x^r) \) to either dominate the game objective or be negligible, defeating the purpose of modeling the agent behavior using a quantal response equilibrium.

Convergence condition. After computing the gradient in each iteration, we use the condition below to check for convergence:

\[
    \left\| \nabla_w J(w, \sigma) - \frac{\nabla_w J(w, \sigma)^T w}{\|w\|^2} \right\| < \beta.
\]

Intuitively, this condition ensures that the directional derivatives that lie tangentially to the unit sphere are smaller than a threshold parameter \( \beta \), implying that the weights \( w \) are near a critical point of \( J(w, \sigma) \) on the unit \( L_2 \) sphere.
Min-Max. Recall that we may have multiple equilibria in a game, and we can switch to different branches with different seeds in the game solver. Thus to only search within the top branch, we reroll seeds to find a seed that provides maximal social cost. Then the algorithm commits to this seed and performs the gradient computation.

Min-Min. The idea in this algorithm is to keep social cost non-increasing, and retry with a different seed otherwise. We take a projected gradient descent step and keep the new weight vector as a candidate $w$, then we only update $w$ when its social cost is smaller or equal to the previous social cost, else we increment the seed number and continue to the next iteration.

**Algorithm 1** Min-Max Projected Gradient Descent

Initialize $k = 1, w \sim Uniform(|\Phi|)$
while $k < k_{max}$ do
  $J = -\infty$
  for $\sigma_{test} \in [0, 1]$ do
    if $J(w, \sigma_{test}) > J$ then
      $\sigma \leftarrow \sigma_{test}$
      $J \leftarrow J(w, \sigma_{test})$
    end if
  end for
  Compute gradient $\nabla_w J(w, \sigma)$
  $w \leftarrow \text{Proj}_{L_2}(w - \alpha \nabla_w J(w, \sigma))$
  if $\|\nabla_w J(w, \sigma) - \frac{\nabla_w J(w, \sigma)^T w}{\|w\|^2} w\| < \beta$ then
    terminate with $w$.
  end if
  $k \leftarrow k + 1$
end while

**Algorithm 2** Min-Min Projected Gradient Descent

Initialize $k = 1, w \sim Uniform(|\Phi|)$
Set $\sigma = 0, J_{prev} = J(w, \sigma)$
while $k < k_{max}$ do
  Compute gradient $\nabla_w J(w, \sigma)$
  if $\|\nabla_w J(w, \sigma) - \frac{\nabla_w J(w, \sigma)^T w}{\|w\|^2} w\| < \beta$ then
    terminate with $w$.
  end if
  $w \leftarrow \text{Proj}_{L_2}(w - \alpha \nabla_w J(w, \sigma))$
  if $J(w, \sigma) > J_{prev}$ then
    $\sigma \leftarrow \sigma + \frac{1}{\alpha}$, continue.
  else
    $w \leftarrow w$
    $J_{prev} = J(w, \sigma)$
  end if
  $k \leftarrow k + 1$
end while

We implement the two algorithms in Julia, calling the game solver described in the beginning of this section. The code of both the game solver and the two projected gradient descent algorithms are publicly available at https://github.com/vivianchen98/relationship_game.

6 Numerical Examples

We test the proposed algorithms in the two congestion game examples described at the end of Section 4.

In two-lane congestion, we consider three players traveling on a road with two lanes, denoted as $a$ and $b$. The load, or the number of players choosing this lane, are $l_a$ and $l_b$. Each player chooses between these two lanes. The delay a player experiences in a lane is determined by that lane’s load: $l_a$ in lane $a$ and $2 \times l_b$ in lane $b$. The cost for each player is based on the delay they experience, influenced by their own lane.
choice as well as the choices of other players. The social cost in this example is the sum of all players’ costs: \( V(S) = \sum_{i=1}^{3} c^i(S) \). The indexed set of relationships includes an identity matrix to account for each player’s individual cost and a directed matrix capturing the relation from each player to the other two players.

\[
\Phi = \begin{cases} 
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{cases}, \quad \begin{cases} 
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{cases}
\]

(15)

In congestion with ambulance, we introduce an additional ambulance \( (A) \) into the two-lane congestion example of regular cars \( (R) \), with the same load functions. Here, the social cost is a weighted sum of the costs incurred by all players, with a higher weight on the ambulance to prioritize its movement:

\[
V(S) = 8 \cdot c^A(S) + \sum_{i \in R} c^i(S)
\]

(16)

The relationship basis includes an identity matrix for each player’s individual cost, along with matrices representing interactions among regular cars, from regular cars to the ambulance, and from the ambulance to regular cars.

\[
\Phi = \begin{cases} 
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{cases}, \quad \begin{cases} 
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
\end{cases}, \quad \begin{cases} 
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
\end{cases}
\]

(17)

We demonstrate the effects of applying Min-Max and Min-Min projected gradient descent on the two examples. Throughout, we let \( \alpha = 0.1 \), \( \beta = 0.0001 \), and the maximum number of iterations be 2000.

Figure 3 shows the convergence of both algorithms for the two examples, with the entropy weight vector \( \lambda \) varied across \( \{0.3, 0.5, 0.7\} \). For a \( \lambda \) value of 0.3 in the two-lane congestion scenario, the Min-Max algorithm coincides with the Min-Min algorithm, indicating the presence of a single equilibrium under this condition. As expected, the Min-Min algorithm consistently finds the lowest social cost, representing the best possible interaction outcome, while the Min-Max algorithm identifies the “upper bound” of social cost among all potential equilibria. This does not necessarily yield the most optimal interaction outcome, but it ensures the final game solution derived from the weight vector is at least as good as that particular social cost value. Due to random seed rerolls, Min-Max can occasionally fail to find the equilibrium with the highest social cost. This is reflected as the spikes in orange lines for \( \lambda = 0.3, 0.5 \) in the congestion with ambulance example. Setting the number of rerolls to be high alleviates this problem at the cost of runtime. As we increase the entropy weight, the
Min-Max — Min-Min

Two Lane Congestion

Congestion with Ambulance

Fig. 3 The convergence of the iterates in Algorithm 1 (Min-Max) and Algorithm 2 (Min-Min) on both congestion game examples. Termination points are highlighted with circles. The plots show the social cost at the current equilibrium $J(w, σ) = J(x, V)$ with respect to the iteration number.

Fig. 4 Superposed relationship graphs for selected iterations of the Min-Min optimization for the congestion with ambulance example, where $λ = 0.5$. Node colors represent the identity matrix for selfish costs, while undirected lines indicate bidirectional relationships between regular cars ($R$).

Min-Min algorithm converges towards higher social cost values because higher entropy introduces more randomness to the game solution and worsens the social cost. Min-Max terminates early in congestion with ambulance with $λ = 0.7$ as it quickly steps...
into an equilibrium which has a terminating condition value below the set threshold 0.0001, while the other trials do not meet the condition early on.

In the congestion with ambulance scenario, Figure 4 demonstrates the evolution of the weighted inter-agent relationships through network graphs. Initially, the algorithm assigns equal weights to all relationships. Over iterations, the regular cars adapt by increasingly prioritizing the ambulance (indicated by a gradually darkening blue line from $R$ to $A$), while competing among themselves (shown as red bidirectional lines between $Rs$). This visualization highlights how the evolved weight vector prompts regular cars to exhibit altruistic behavior towards the ambulance, mirroring the real-world social norm of yielding to emergency vehicles. The graph at iteration 200 reveals a counter-intuitive result: the proposed algorithm determines it is optimal for regular cars to actively impede each other by negatively incorporating each other’s costs, in order to prioritize the passage of the ambulance.

7 Conclusion

In this paper, we present a novel approach to model socially-aware decision-making in autonomous agents within a static game framework. We first define the concept of a relationship game, where an indexed set of player relationships augments a static game. We formulate two problems to optimize the weight vector corresponding to this set of relationships such that the modified game solution minimizes an objective function: the minimal social cost for Min-Min and the maximal social cost for Min-Max. We then propose two projected gradient descent algorithms tailored to solve each of these two problems and demonstrate their effectiveness numerically in two congestion game scenarios.

A limitation of our approach is the use of stochastic algorithms, which inherently require a balance between the number of seeds and runtime efficiency. Increasing the number of seeds can enhance the robustness and accuracy of the solution, but this comes at the expense of increased computational time. This trade-off can be particularly challenging in scenarios where rapid decision-making is critical, or when computational resources are limited. Future work could focus on developing more efficient stochastic methods that require fewer seeds to achieve a similar level of accuracy.

8 Acknowledgement

The authors would like to thank Yue Yu for his insights.

References

[1] Naeger, N.L., Peso, M., Even, N., Barron, A.B., Robinson, G.E.: Altruistic behavior by egg-laying worker honeybees. Current Biology (2013) https://doi.org/10.1016/j.cub.2013.06.045

[2] Başar, T., Olsder, G.J.: Dynamic Noncooperative Game Theory. SIAM, Philadelphia (1998). https://doi.org/10.1137/1.9781611971132
[3] McKelvey, R.D., Palfrey, T.R.: Quantal response equilibria for normal form games. Games and economic behavior (1995) https://doi.org/10.1006/game.1995.1023

[4] Kuipers, B.: Perspectives on Ethics of AI. In: The Oxford Handbook of Ethics of AI (2020). https://doi.org/10.1093/oxfordhb/9780190067397.001.0001

[5] Kleiman-Weiner, M., Saxe, R., Tenenbaum, J.B.: Learning a Commonsense Moral Theory. In: Journal of Cognition (2017). https://doi.org/10.1016/j.cognition.2017.03.005

[6] Roughgarden, T.: Algorithmic game theory. Communications of the ACM (2010) https://doi.org/10.1017/CBO9780511800481

[7] Chen, P.-A., Keijzer, B.D., Kempe, D., Schäfer, G.: Altruism and its impact on the price of anarchy. ACM Transactions on Economics and Computation (2014) https://doi.org/10.1145/2597893

[8] Caragiannis, I., Kaklamanis, C., Kanellopoulos, P., Kyropoulou, M., Papaioannou, E.: The impact of altruism on the efficiency of atomic congestion games. In: Trustworthy Global Computing: 5th International Symposium (2010). https://doi.org/10.1007/978-3-642-15640-3_12

[9] Gollapudi, S., Kollias, K., Panigrahi, D., Pliatsika, V.: Profit sharing and efficiency in utility games. In: 25th Annual European Symposium on Algorithms (2017). https://doi.org/10.4230/LIPIcs.ESA.2017.43

[10] Bhawalkar, K., Gairing, M., Roughgarden, T.: Weighted congestion games: the price of anarchy, universal worst-case examples, and tightness. ACM Transactions on Economics and Computation (2014) https://doi.org/10.1007/978-3-642-15781-3_2

[11] Gairing, M., Kollias, K., Kotsialou, G.: Existence and efficiency of equilibria for cost-sharing in generalized weighted congestion games. ACM Transactions on Economics and Computation (2020) https://doi.org/10.1145/3391434

[12] Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R.S., Guo, E.: On differentiating parameterized argmin and argmax problems with application to bi-level optimization. arXiv preprint:1607.05447 (2016)

[13] Magnus, J.R., Neudecker, H.: Matrix Differential Calculus with Applications in Statistics and Econometrics. John Wiley & Sons, New York (1988). https://doi.org/10.1002/9781119541219

[14] Amos, B., Kolter, J.Z.: Optnet: Differentiable optimization as a layer in neural networks. In: International Conference on Machine Learning (2017). https://proceedings.mlr.press/v70/amos17a.html
Appendix A  Gradient Derivation via Implicit Differentiation

Let $z$ be the flattened version of $x$, i.e.,

$$ z = f(x) = [x^1^T x^2^T \ldots x^n^T]^T. $$

$f$ induces a natural bijection between $x$ and $z$. Thus with a slight abuse of notation, we define

$$ J(z, V) := J(f^{-1}(z), V) = J(x, V). \quad (A1) $$

The gradient $\nabla_w J(x, V)$ we want can instead be computed by

$$ \nabla_w J(z, V) = \nabla_x J(z, V) \cdot \frac{\partial z}{\partial w}. \quad (A2) $$
The first gradient of expected cost with respect to $x$ is straightforward:

$$
\nabla_x J(z, V) = \nabla_{f(x)} J(f(x), V) = \left[ \frac{\partial J}{\partial x_j} \right]_{j \in N, \forall j \in S^i}
$$

(A3)

$$
\frac{\partial J}{\partial x_j} = \sum_{s^i = 0, \ldots, s^i \in S^i} \left( \sum_{t \neq i} (s^i) \prod_{t \neq j} x^i(s^t) \right).
$$

The partial derivative $\frac{\partial z}{\partial w}$ can be derived by applying the implicit function theorem on the KKT condition of Nash equilibrium.

We define $h^i(w, z) = h^i(w, f(x)) = \left[ \sum_{s^i = r, s^i \in S^i} \left( \tilde{c}^i(s) \prod_{t \neq i} x^i(s^t) \right) \right]_r$.

(A4)

$$
g^{i,j}(w, z) = g^{i,j}(w, f(x)) = \left[ \sum_{s^i = r, s^j = c, s^i \in S^i, s^j \in S^j} \left( \tilde{c}^i(s) \prod_{t \neq i, j} x^i(s^t) \right) \right]_{rc}.
$$

Then we let

$$
F(w, z) = z - s(w, z) = 0,
$$

where $s(w, z) = \left[ \text{softmax} \left( - \frac{1}{\lambda} h^i(w, z) \right) \right]_{i \in N}$,

and by implicit function theorem we have

$$
\frac{\partial z}{\partial w} = - \left( \frac{\partial F}{\partial z} \right)^{-1} \frac{\partial F}{\partial w}. \quad \text{(A5)}
$$

With $|z| = \sum_{i \in N} |S^i|$ and $J_{\text{softmax}}$ as the Jacobian of the softmax function, we have

$$
\frac{\partial F}{\partial z} = \frac{\partial z}{\partial w} - \frac{\partial s}{\partial z} = I_{|z|} + \frac{1}{\lambda} J_{\text{softmax}} \left( - \frac{1}{\lambda} h^i(w, z) \right) g^{i,j}(w, z),
$$

$$
\frac{\partial F}{\partial w} = - \frac{\partial s}{\partial w} = \frac{1}{\lambda} J_{\text{softmax}} \left( - \frac{1}{\lambda} h^i(w, z) \right) \frac{\partial h^i}{\partial w}. \quad \text{(A6)}
$$
where \( \frac{\partial h_i}{\partial w} \) is the Jacobian of \( h_i \) with respect to \( w \). That is,

\[
\frac{\partial h_i}{\partial w} = \left[ \sum_{s_i = r_i} \left( \sum_{j \in N} \phi_{ij}^c c_j(s) \right) \prod_{t \neq i} x^t(s') \right]_{rc}.
\] (A7)

In the implementation, Julia’s \ operator, performing QR factorization, computes eq. (A5) with better numerical stability.