D3C: Reducing the Price of Anarchy in Multi-Agent Learning

Ian Gemp
DeepMind
London, United Kingdom
imgemp@deepmind.com

Kevin R. McKee
DeepMind
London, United Kingdom
kevinmckee@deepmind.com

Richard Everett
DeepMind
London, United Kingdom
reverett@deepmind.com

Edgar Duéñez-Guzmán
DeepMind
London, United Kingdom
duenez@deepmind.com

Yoram Bachrach
DeepMind
London, United Kingdom
yorambac@deepmind.com

David Balduzzi
XTX Markets
London, United Kingdom
dbalduzzi@gmail.com

Andrea Tacchetti
DeepMind
London, United Kingdom
atacchet@deepmind.com

ABSTRACT

In multiagent systems, the complex interaction of fixed incentives can lead to outcomes that are poor (inefficient) not only for the group, but also for each individual. Price of anarchy is a technical, game-theoretic definition that quantifies the inefficiency arising in these scenarios—it compares the welfare that can be achieved through perfect coordination against that achieved by self-interested agents at a Nash equilibrium. We derive a differentiable, upper bound on a price of anarchy that agents can cheaply estimate during learning. Equipped with this estimator, agents can adjust their incentives in a way that improves the efficiency incurred at a Nash equilibrium. Agents do so by learning to mix their reward (equiv. negative loss) with that of other agents by following the gradient of our derived upper bound. We refer to this approach as D3C. In the case where agent incentives are differentiable, D3C resembles the celebrated Win-Stay, Lose-Shift strategy from behavioral game theory, thereby establishing a connection between the global goal of maximum welfare and an established agent-centric learning rule. In the non-differentiable setting, as is common in multiagent reinforcement learning, we show the upper bound can be reduced via evolutionary strategies, until a compromise is reached in a distributed fashion. We demonstrate that D3C improves outcomes for each agent and the group as a whole on several social dilemmas including a traffic network exhibiting Braess’s paradox, a prisoner’s dilemma, and several multiagent domains.

KEYWORDS

Price of Anarchy; Nash; Reward Sharing; Win-Stay Lose-Shift; Collective Intelligence; Multiagent Reinforcement Learning

1 INTRODUCTION

We consider a setting consisting of many interacting artificially intelligent agents, each with specific individual incentives. It is well known that the interactions between individual agent goals can lead to inefficiencies at the group level, for example, in environments exhibiting social dilemmas [6, 15, 21]. In order to resolve these fundamental inefficiencies, agents must reach a compromise.

Any arbitration mechanism with a central coordinator1 faces challenges when scaling to large populations. The coordinator’s task becomes intractable as it must both query preferences from a larger population and make decisions accounting for the exponential growth of agent interactions. If agents are permitted to modify their incentives over time, the coordinator must collect all this information again, exacerbating the computational burden. In addition, a central coordinator represents a single point of failure for the system whereas successful multiagent systems identified in nature (e.g., market economies, ant colonies, etc.) are often robust to node failures [10]. Therefore, we focus on decentralized approaches.

Design Criteria: The celebrated Myerson-Satterthwaite theorem [3, 14, 27, 38] states that no mechanism can simultaneously achieve optimal efficiency (welfare-maximizing behavior), budget-balance (no taxing agents, burning side-payments, or hallucinating rewards), appeal to rational individuals (individuals want to opt-in to the mechanism), and be incentive compatible (resulting behavior is a Nash equilibrium). While this impossibility result precludes a mechanism that satisfies the above criteria perfectly, it says nothing about a mechanism that satisfies them approximately, which is our aim here. In addition, the mechanism should be decentralized, extensible to large populations, and adapt to learning agents with evolving incentives in possibly non-stationary environments.

Design: We formulate compromise as agents mixing their incentives (rewards or losses) with others. In other words, an agent may become incentivized to minimize a mixture of their loss and other agents’ losses. We design a decentralized meta-algorithm that allows agents to search over the space of these possible mixtures.

---

1For example, the Vickrey-Clarke-Groves (VCG) mechanism [7].
2 DYNAMICALLY CHANGING THE GAME

In our approach, agents may consider slight re-definitions of their original losses, thereby changing the definition of the original game. Critically, this is done in a way that conserves the original sum of losses (budget-balanced) so that the original group loss can still be measured. In this section, we derive our approach to minimizing the price of anarchy in several steps. First we formulate minimizing the price of anarchy via compromise as an optimization problem. Second we specifically consider compromise as the linear mixing of agent incentives. Next, we define a local price of anarchy and derive an upper bound that agents can differentiate. Then, we decompose this bound into a set of differentiable objectives, one for each agent. Finally, we develop a gradient estimator to minimize the agent objectives in settings with bandit feedback (e.g., RL) that enables scalable decentralization.

2.1 Notation and Transformed Losses

Let agent $i$’s loss be $f_i(x) : x \in X \rightarrow \mathbb{R}$, where $x$ is the joint strategy of all agents. Let $f_i^A(x)$ denote agent $i$’s transformed loss which mixes losses among agents. Let $f(x) = [f_1(x), \ldots, f_n(x)]^T$ and $f^A(x) = [f_1^A(x), \ldots, f_n^A(x)]^T$ where $n \in \mathbb{Z}$ denotes the number of agents. In general, we require $f_i^A(x) > 0$ and $\sum_j f_j^A(x) = \sum_i f_i(x)$ so that total loss is conserved. Under these constraints, the agents will simply explore the space of possible non-negative group loss decompositions. We consider transformations of the form $f_i^A(x) = A^T f_i(x)$ (note the transpose) where each agent $i$ controls row $i$ of $A$ with each row constrained to the simplex, i.e., $A_i \in \mathcal{A}^{n-1}$. For example, agent 1’s loss is mixed according to the first column of $A$ which may not sum to 1, and not the first row, which it controls:

$$[A_{11}, A_{21}, A_{31}]$$

$$f^A_i(x) = \{[0.9, 0.3, 0.5], f_1(x), f_2(x), f_3(x)\}.$$  (1)

Lastly, $[a; b] = [a^T, b^T]^T$ signifies row stacking of vectors, and $X^*$ denotes the set of Nash equilibria.

2.2 Price of Anarchy

Nisan et al. [29] define price of anarchy as the worst value of an equilibrium divided by the best value in the game. Here, value means sum of player losses, best means lowest, and Nash is the chosen equilibrium concept. It is well known that Nash can be arbitrarily bad from both an individual agent and group perspective; Appx. B presents a simple example and demonstrates how opponent shaping [12, 23] is not a balm for these issues. With the above notation, the price of anarchy is defined as

$$\rho^X(f^A) \triangleq \max_{x^*} \frac{\sum_i f_i^A(x^*)}{\min_x \sum_i f_i^A(x)} \geq 1.$$  (2)

Note that computing the price of anarchy precisely requires solving for both the optimal welfare and the worst case Nash equilibrium. We explain how we circumvent this issue in §2.4.

2.3 Compromise as an Optimization Problem

Given a game, we want to minimize the price of anarchy by perturbing the original agent losses:

$$\min_{f' \in \mathcal{V}_A(f)} \rho^X(f') + v D(f, f')$$  (3)

$$\forall f' \in \mathcal{X}' \Rightarrow f \neq f'$$

The strict definition of price of anarchy assumes positive losses. This is relaxed in §2.5 to allow for losses in $\mathbb{R}$.
where \( f \) and \( f' = \psi(f) \) denote the vectors of original and perturbed losses respectively, \( \psi : \mathbb{R}^n \to \mathbb{R}^n \) is parameterized by weights \( A, \nu \) is a regularization hyperparameter, and \( D \) penalizes deviation of the perturbed losses from the originals or represents constraints through an indicator function. To ensure minimizing the price of anarchy of the perturbed game improves on the original, we incorporate the constraint that the sum of perturbed losses equals the sum of original losses, \( \mathbf{1}^T f' = \mathbf{1}^T f \). We refer to this approach as \( \rho \)-minimization.

Our agents reconstruct their losses using the losses of all other agents as a basis. For simplicity, we consider linear transformations of their loss functions, although the theoretical bounds hereafter are independent of this simplification. We also restrict ourselves to convex combinations so that agents do not learn incentives that are directly adverse to other agents. The problem can now be reformulated. Let \( \tilde{\psi}_A(f) = A^T f \) and \( D(f, f') = \sum_i D_{KL}(e_i || A_i) \) where \( A \in \mathbb{R}^{n \times n} \) is a right stochastic matrix (rows are non-negative and sum to 1), \( e_i \in \mathbb{R}^n \) is a unit vector with a 1 at index \( i \), and \( D_{KL} \) denotes the Kullback-Leibler divergence.

2.4 A Local Price of Anarchy

The price of anarchy, \( \rho \geq 1 \), is defined over the joint strategy space of all players. Computing it is intractable for general games. However, many agents learn via gradient-based training, and so only observe the portion of the strategy space explored through their learning trajectory. Hence, we imbue our agents with the ability to locally estimate the price of anarchy along this trajectory.

**Definition 1 (Local Price of Anarchy).** Define

\[
\rho_x(f^A, \Delta t) = \frac{\max_{x'} \sum_{i} f^A_i(x')}{\min_{x} \sum_{i} f^A_i(x - \tau F(x))} \geq 1
\]

(4)

where \( F(x) = [\nabla x_1 f^A_1(x); \ldots; \nabla x_n f^A_n(x)] \), \( \Delta t \) is a small step size, \( f^A_i \) is assumed positive \( \forall i \), and \( X' \) denotes the set of equilibria of the game when constrained to the line.

![](image)

**Figure 1:** Agents estimate the price of anarchy assuming the joint strategy space, \( X' \), of the game is restricted to a local linear region, \( X_T \), extending from the currently learned joint strategy, \( x_t \), to the next, \( x_{t+1} \). \( \rho_x \) and \( \rho_X \) denote the global and local price of anarchy.

To obtain bounds, we leverage theoretical results on smooth games, summarized as a class of games where “the externality imposed on any one player by the others is bounded” [36]. We assume a Lipschitz property on all \( f^A_i(x) \) (details in Theorem 1), which allows us to appeal to this class of games. The bound in equation 7 is tight for some games. Proofs can be found in Appx. D.

For convenience, we repeat the core definition and lemma put forth by Roughgarden [36] here.

**Definition 2 (Smooth Game).** A game is \((\lambda, \mu)\)-smooth if:

\[
\sum_{i=1}^{n} f^A_i(x_i, x'_i) \leq \lambda \sum_{i=1}^{n} f^A_i(x_i, x_{-i}) + \mu \sum_{i=1}^{n} f^A_i(x'_i, x'_{-i})
\]

(5)

for all \( x, x' \in X \) where \( \lambda > 0, \mu < 1 \). \( x_{-i} \) denotes all player \( j \neq i \) strategies and \( \sum_i f^A_i(x) \) is assumed to be non-negative for any \( x \in X \).

The last condition is needed for the price of anarchy, a ratio of welfare, to be meaningful as a positive measure of inefficiency.

**Lemma 1 (Smooth Games Imply a Bound on Price of Anarchy).** The price of anarchy is bounded above by a ratio of the coefficients that satisfy the smooth game definition [36]:

\[
1 \leq \rho_x(f^A) \leq \inf_{\lambda > 0, \mu < 1} \left[ \frac{\lambda}{1 - \mu} \right].
\]

(6)

**Theorem 1 (Local Utilitarian Price of Anarchy).** Assuming each agent’s loss is positive and its loss gradient is Lipschitz, there exists a learning rate \( \Delta t > 0 \) sufficiently small such that, to \( O(\Delta t^2) \), the local utilitarian price of anarchy of the game, \( \rho_x(f^A, \Delta t) \), is upper bounded by

\[
\max_i \left[ 1 + \Delta t \right. \log \left( \frac{\mu \sum_{i} f^A_i(x)}{\bar{\mu}} \right) + \left. \frac{||\nabla x_i f^A_i(x)||^2}{\bar{\mu} f^A_i(x)} \right] \]

(7)

where \( i \) indexes each agent, \( \bar{\mu} \in \mathbb{R}_{\geq 0} \) is a user-defined upper bound on the true \( \mu, \log U(z) \equiv \max(0, z) \), and Lipschitz implies there exists a \( \beta_i \) such that \( ||\nabla x_i f^A_i(x) - \nabla y_i f^A_i(y)|| \leq \beta_i ||x - y|| \forall x, y \). A.

**Proof Sketch:** For a small enough region (grayed in Figure 1), we can approximate each agent’s loss function with its Taylor series expansion. By rewriting all losses in the smoothness constraint (equation 5) in terms of expansions about \( x \) or \( x' \), i.e., quantities we can measure before and after a joint gradient step, we can proceed to define the smoothness constraint with \( \mu \) and \( \lambda \) in terms of measurable quantities. The smoothness constraint is formulated as a sum over the \( n \) agents, but we can decompose this constraint into \( n \) individual constraints with their own \( \mu_i \)’s and \( \lambda_i \)’s. If each agent can ensure local individual smoothness, which is possible for a small enough region, we show this is sufficient to satisfy the original local smoothness condition with \( \mu = \max_i \mu_i \) and \( \lambda = \max_i \lambda_i \). Each agent can further estimate their own individual price of anarchy, \( \rho_i \), via equation 6 which reduces to a tractable two dimensional constrained optimization problem with a closed form solution. We further show that we can upper bound the local price of anarchy for the group (equation 4) with the max of these individual estimates. Finally, using another expansion along with the log-trick famous from the policy gradient theorem, we recover the final result presented in Theorem 1 below. The Lipschitz assumption exists simply to ensure the series approximations are sufficiently accurate for a small enough region. The full proof is in Appx. D.

Recall that this work focuses on price of anarchy defined using total loss as the value of the game. This is a utilitarian objective. We also derive an upper bound on the local egalitarian price of anarchy where value is defined as the max loss over all agents (replace \( \sum_i \) with max; see equation 4; see Appx. D.2), possibly of independent interest.

**Theorem 2.** Given \( n \) positive losses, \( f^A_i(x), i \in \{1, \ldots, n\} \), with \( \beta_i \)-Lipschitz gradients there exists a \( \Delta t > 0 \) sufficiently small such
that, to $O(\Delta^2)$, the local egalitarian price of anarchy of the game is upper bounded by

$$\rho_e \leq 1 + \Delta t \text{ReLU} \left( \frac{d}{dt} \log(\max_{i} [f_i^A(x)]) + \sum_{i=1}^n \left| \frac{\nabla x_i f_i^A(x)}{\mu \max_i [f_i^A(x)]} \right|^2 \right).$$  \hspace{1cm} (8)

### 2.5 Decentralized Learning of the Loss Mixture Matrix $A$

Minimizing equation 3 w.r.t. $A$ can become intractable if $n$ is large. Moreover, if solving for $A$ at each step is the responsibility of a central authority, the system is vulnerable to this authority failing. A distributed solution is therefore appealing, and the local price of anarchy bound admits a natural relaxation that decomposes over agents ($\max_i z_i \leq \sum_i z_i$ for $z_i \geq 0$). Equation 3 then factorizes as

$$\min_{A_i \in \mathcal{M}_i^{-1}} \rho_i + \nu \mathcal{D}_{KL}(\mathcal{E} || A_i)$$

where $\rho_i = 1 + \Delta t \text{ReLU} \left( \frac{d}{dt} \log(\max_i [f_i^A(x)]) + \sum_{i=1}^n \left| \frac{\nabla x_i f_i^A(x)}{\mu \max_i [f_i^A(x)]} \right|^2 \right)$. Local price of anarchy is subdifferentiable w.r.t. each $A_i$ with gradient

$$\nabla A_i \rho_i \approx \nabla A_i \text{ReLU} \left( \frac{d}{dt} \log(f_i^A(x)) + \sum_{i=1}^n \left| \frac{\nabla x_i f_i^A(x)}{f_i^A(x) \mu} \right|^2 \right).$$  \hspace{1cm} (10)

The log appears due to price of anarchy being defined as the worst case Nash total loss divided by the minimal total loss. We propose the following modified learning rule for a hypothetical price of anarchy which is defined as a *difference* and accepts negative loss: $A_i \leftarrow A_i - \eta A \nabla A_i (\rho_i + \nu \mathcal{D}_{KL})$ where $\eta A$ is a learning rate and

$$\nabla A_i \rho_i \leftarrow \nabla A_i \text{ReLU} \left( \frac{d}{dt} \log(f_i^A(x)) + \sum_{i=1}^n \left| \frac{\nabla x_i f_i^A(x)}{f_i^A(x) \mu} \right|^2 \right) + \epsilon.$$

The update direction in (11) is proportional to $\nabla A_i \rho_i$ asymptotically for large $f_i^A$; see Appx. D.1.1 for further discussion. Each agent $i$ updates $x_i$ and $A_i$ simultaneously using $\nabla x_i f_i^A(x)$ and $\nabla A_i (\rho_i + \nu \mathcal{D}_{KL})$.

### Improve-Stay, Suffer-Shift—Win-Stay, Lose-Shift (WSLS)

This is formally accomplished with approximate one-shot gradient estimates [37] or *evolutionary strategies* [34]. A one-shot gradient of $\rho_i(A_i)$ is estimated by first perturbing $A_i$ with entropic mirror ascent [4] as $\tilde{A}_i = \log \max \left( \frac{\Delta i}{\Delta n} \right)$ where $\Delta > 0$ and $\Delta_i \sim \mathcal{U}_{\rho_i}(n)$ is drawn uniformly from the unit sphere in $\mathbb{R}^n$. The perturbed weights are then evaluated $\tilde{\rho_i} = \rho_i(\tilde{A}_i)$. Finally, an unbiased gradient is given by $\frac{\tilde{\rho}_i}{\tilde{\Delta}} \tilde{a}_i$. In practice, we cannot evaluate in one shot the $\frac{d}{dt} f_i^A(x)$ term that appears in the definition of $\rho_i$. Instead, Algorithm 1 uses finite differences and we assume the evaluation remains accurate enough across training steps.

#### Algorithm 1 D3C Update for RL Agent $i$

```
Input: $\eta A, \delta, \nu, \tau_{min}, \tau_{max}, A_i^0, \epsilon, I, \mathbb{L}$, iterations $T$
$A_i \leftarrow A_i^0$ \quad \{Initialize Mixing Weights\}
$G = 0$ \quad \{Initialize Mean Return of Trial\}
\{Draw Initial Random Mixing Trial\}
$A_i, \tilde{a}_i, \tau, t_b, G_b \leftarrow trial1(\delta, \tau_{min}, \tau_{max}, A_i, 0, G)$
for $t = 1 : T$ do
    $g = L_i(\tilde{A}_i \forall j)$ \quad \{Update Policy With Mixed Rewards\}
    $\Delta t_b = t - t_b$ \quad \{Elapsed Trial Steps\}
    $G = (G/(\Delta t_b + 1)) / \Delta t_b$ \quad \{Update Mean Return\}
    if $\Delta t_b = \tau$ \{Trial Complete\} then
        $\hat{\rho}_i = \text{ReLU} \left( \frac{G_b + \epsilon}{\tau} \right)$ \quad \{Approximate $\rho$\}
        $\nabla A_i = \hat{\rho}_i \tilde{a}_i - \nu \mathcal{E} \oplus A_i$ \quad \{Estimate Gradient \rightarrow (11)\}
        $A_i = \text{softmax} \left( L_i(A_i) - \eta A \nabla A_i \right)$ \quad \{Update\}
        $G_b \leftarrow \text{trial1}(\delta, \tau_{min}, \tau_{max}, A_i, t, G)$
    end if
end for
```

Algorithm 1 requires several arguments: $\eta A$ is a global learning rate for each $A_i$, $\delta$ is a perturbation scalar for the one-shot gradient estimate, $\tau_{min}$ and $\tau_{max}$ specify the lower and upper bounds for the duration of the mixing trial for estimating a finite difference of $\frac{d}{dt} f_i^A(x)$ \rightarrow $(G - G_b) / \tau$, $I$ and $\mathbb{L}$ specify lower and upper bounds for clipping $A$ in logit space $(\mathbb{L}, \mathbb{K})$, and $\mathcal{L}_i$ (Algorithm 3) represents any generic reinforcement learning algorithm augmented to take $A$ as input (in order to mix rewards) and outputs discounted return. $\oplus$ indicates elementwise division.
Algorithm 2 trial—helper function

Input: $\delta$, $\tau_{\text{min}}$, $\tau_{\text{max}}$, $A_i$, $t$, $G$

1. $\tilde{a}_i \sim U_p(n)$ [Sample Perturbation Direction]
2. $A_1 = \text{softmax}(\log(A_1) + \delta \tilde{a}_i)$ [Perturb Mixture]
3. $\tau \sim \text{Uniform}([\tau_{\text{min}}, \tau_{\text{max}}])$ [Draw Random Trial Length]

Output: $A_i$, $\tilde{a}_i$, $\tau$, $t$, $G$

Algorithm 3 $L_\pi$—example learner

Input: $\tilde{A} = [\tilde{A}_1; \ldots; \tilde{A}_n]$

while episode not terminal do
1. draw action from agent policy
2. play action and observe reward $r_i$
3. broadcast $r_i$ to all agents
4. update policy with $\tilde{r}_i = \sum_j \tilde{A}_{ji} r_j$
end while

Output: return over episode $g$

2.7 Assessment

We assess Algorithm 1 with respect to our original design criteria. As described, agents perform gradient descent on a decentralized and local upper bound on the price of anarchy. Recall that a minimal global price of anarchy ($\rho = 1$) implies that even the worst case Nash equilibrium of the game is socially optimal; similarly, Algorithm 1 searches for a locally socially optimal equilibrium. By design, $A_i \in \Delta^{n-1}$ ensures the approach is budget-balancing. We justify the agents learning weight vectors $A_i$ by initializing them to attend primarily to their own losses as in the original game. If they can minimize their original loss, then they never shift attention according to equation 11 because $\frac{df_i}{df} \leq 0$ for all $t$. They only shift $A_i$ if their loss increases. We also include a KL term to encourage the weights to return to their initial values. In addition, in our experiments with symmetric games, learning $A$ helps the agents’ outcomes in the long run. We also consider experiments in Appx. E.2.1 where only a subset of agents opt into the mechanism. If each agent's original loss is convex with diagonally dominant Hessian and the strategy space is unconstrained, the unique, globally stable fixed point of the game defined with mixed losses is a Nash (see Appx. H.4). Exact gradients $\nabla_{A_i, \rho_i}$ require each agent differentiates through all other agents’ losses precluding a fully decentralized and scalable algorithm. We circumvent this issue with noisy oneshot gradients. All that is needed in terms of centralization is to share the mixed scalar rewards; this is cheap compared to sharing $\chi_j \in \mathbb{R}^{d \times n}$. As mentioned in the introduction, the cost of communicating rewards can be mitigated by learning $A_i$ via sparse optimization or sampling but is outside the scope of this paper.

2.8 Related Work

Collective Intelligence or COIN, surveyed in [46], examines the problem of how to design reward functions for individual agents such that a decentralized multiagent system maximizes a global world utility function. Wolpert and Tumer [46] describe several approaches taken by an array of diverse fields and motivate the creation of a collective intelligence as an important challenge. Follow-up works focus on aiding researchers in deriving static agent reward functions that are consistent with optimizing the desired world utility via, for instance, useful visualizations [1, 2]. Unlike conventional COIN approaches, D3C learns agent reward functions dynamically through online interaction with the environment. On the other hand, like D3C, studies in COIN find that agents optimizing modified versions of their original reward functions not only achieve high global utility, but also perform better individually [41].

In recent MARL work, Lupu and Precup [25] augment the agents’ action space with a “gifting” action where agents can send a $+1$ reward to another agent. They evaluate this approach on a variant of Harvest we explore in Appx. F.4. They look at three different reward budget settings; ours is most similar to their zero-sum setting in which gifts are budget-balanced by matching $-1$ penalties. In contrast to [25], we consider a continuum of “gifting” amounts automatically grounded in the scale of the original rewards via mixing on the simplex.

Similarly, Hostallero et al. [16] introduce PED-DQN where agents gift their peers by a reciprocal amount proportional to the positive externality they perceive (as measured by their td-error) receiving from the group. Although they make no direct reference to price of anarchy, the stated goal is to shift the system’s equilibrium towards an outcome that maximizes social welfare. In contrast to [16], D3C agents learn to share varying rewards with individual agents rather than sharing an average gift with everyone in their predefined peer group. This is important as the latter prevents the possible discovery of teams as demonstrated by D3C in Appx. F.5.

Yang et al. [47] propose an algorithm LIO (Learning to Incentivize Others) that equips agents with “gifting” policies represented as neural networks. At each time step, each agent observes the environment and actions of all other agents to determine how much reward to gift to the other agents. The parameters of these networks are adjusted to maximize the original environment reward (without gifts) minus some penalty regularizer for gifting meant to approximately maintain budget-balance. In order to perform this maximization, each agent requires access to every other agent’s action-policy, gifting-policy, and return making this approach difficult to scale and decentralize. Yang et al. [47] demonstrate LIO’s ability to maximize welfare and achieve division of labor on a very restricted version of the Cleanup game we evaluate in §3.5. We also evaluate D3C on this restricted variant in the Appx. F.3.

Inspired by social psychology, McKee et al. [26] explored imbuing agents with a predisposed social value orientation that modifies their rewards. Populations with heterogeneous populations achieved higher fitness scores than homogeneous ones in an evolutionary training approach (i.e., learning occurs outside the agent’s lifetime).

One key innovation of D3C beyond the above works is its budget-balance guarantee. In [16, 26, 47], agents manifest extra reward to gift to peers, but no explanation is given for where this extra reward might come from. Also, none of these works tie their proposed approaches to the fundamental game theoretic concept price of anarchy. The derivation of D3C from first principles provides an explicit link, showing an agent-centric learning rule can be approximately consistent with the global objective of maximal social welfare.

Like D3C, OpenAI Five [31] also linearly mixed agents rewards which each other, but where the single “team spirit” mixture parameter ($\tau$) is manually annealed throughout training from 0.3 to 1.0 (i.e., $A_{ij} = 1 - 0.8\tau$, $A_{ij} = 0.2\tau$, $j \neq i$).
Finally, we point out that loss transformation is consistent with human behavior. Within social psychology, *interdependence theory* [19] holds that humans make decisions based on self interest and social preferences, allowing them to avoid poor Nash equilibria.

## 3 EXPERIMENTS

Here, we show that agents minimizing local estimates of price of anarchy achieve lower loss on average than selfish, rational agents in five domains. In the first two domains, a traffic network (4 players) and a generalized prisoner’s dilemma (10 players), players optimize using exact gradients (see equation 11). Then in three RL domains—Trust-Your-Brother, Coins and Cleanup—players optimize with approximate gradients as handled by Algorithm 1. Agents train with deep networks and A2C [11]. We refer to both algorithms as D3C (decentralized, differentiable, dynamic compromise).

For D3C, we initialize $A_{ij} = 0.99$ and $A_{ij} = \frac{0.01}{n-1}$, $j \neq i$. We initialize away from a onehot because we use entropic mirror descent [4] to update $A_{ij}$, and this method requires iterates to be initialized to the interior of the simplex. In the RL domains, updates to $A_{ij}$ are clipped in logit-space to be within $-5$ and $5$ (see Algorithm 1). We set the $D_{KL}$ coefficient to 0 except for in Coins, where $v = 10^{-5}$. Additional hyperparameters are specified in Appx. G. In experiments where we cannot compute price of anarchy (equation 2) exactly, we either report the total loss of the learning algorithm (e.g., D3C) or the ratio of these losses referred to as “ratio to optimal”.

### 3.1 Traffic Networks and Braess’s Paradox

In 2009, New York city’s mayor closed Broadway near Times Square to alleviate traffic congestion [28]. This counter-intuitive phenomenon, where restricting commuter choices improves outcomes, is called Braess’s paradox [5, 6, 43], and has been observed in real traffic networks [40, 48]. Braess’s paradox is also found in physics [48], decentralized energy grids [45], and can cause extinction cascades in ecosystems [37]. Knowing when a network may exhibit this paradox is difficult, which means knowing when network dynamics may result in poor outcomes is difficult.

Figure 2a presents a theoretical traffic network. Without edge AB, drivers commute according to the Nash equilibrium, either learned by gradient descent or D3C. Figure 3a shows the price of anarchy approaching 1 for both algorithms. If edge AB is added, the network now exhibits Braess’ paradox. Figure 3b shows that while gradient descent converges to Nash ($\rho = \frac{80}{99}$), D3C achieves an average “ratio to optimal” near 1. Figure 2b shows that when faced with a randomly drawn network, D3C agents achieve shorter commutes on average than agents without the ability to compromise.

### 3.2 Prisoner’s Dilemma

In an $n$-player prisoner’s dilemma, each player must decide to defect or cooperate with each of the other players creating a combinatorial action space of size $2^n - 1$. This requires a payoff tensor with $2^{n(n-1)}$ entries. Instead of generalizing prisoner’s dilemma [33] to $n$ players using nth order tensors, we translate it to a game with convex loss functions. Figure 4a shows how we can accomplish this. Generalizing this to $n$ players, we say that for all $i, j, k, \text{distinct}$, 1) player $i$ wants to defect against player $j$, 2) player $i$ wants player $j$ to defect against player $k$, and 3) player $i$ wants player $j$ to cooperate with itself. In other words, each player desires a free-for-all with the exception that no one attacks it. See Appx. E.2 for more details.

For three players, we can define the vector of loss functions with

$$f(x) = \sum_{\text{columns}} \left( \left\| \begin{bmatrix} x_i^T \\ x_j^T \\ x_k^T \end{bmatrix} - C \right\|^2 \right)$$ (12)

where $x = \{x_{ij}\}$ is a column vector ($i \in [1, n], j \in [1, n-1]$) containing the players’ (randomly initialized) strategies and $C$ is an $n \times n(n-1)$ matrix with entries that either equal 0 or $c \in \mathbb{R}^+$. The price of anarchy grows unbounded as $c \to 0^+$. We set $n = 10$ and $c = 1$ ($\rho = \frac{10}{99}$) in this experiment with additional settings explored in Appx. F.1.

Figure 6 highlights a single training run. Both agents are initialized to minimize their original loss, but then learn over training to minimize the mean of the two player losses.

### 3.3 Trust-Your-Brother

In this game, a predator chases two prey around a table. The predator uses a hard-coded policy to move towards the nearest prey unless it is already adjacent to a prey, in which case it stays put. If
Figure 3: Traffic Network—(a) Without edge AB, agents are initialized with random strategies and train with either gradient descent (left) or D3C (right)—similar performance is expected. Statistics of 1000 runs are plotted over training. Median $\rho_{\text{max}}$ tracks the median over trials of the longest-commute among the four drivers. The shaded region captures $\pm$ 1 stddev around the mean. (b) After edge AB is added, agents are initialized with random strategies and trained with either gradient descent (left) or D3C (right).

Figure 4: (a) A reformulation of the prisoner’s dilemma using convex loss functions instead of a normal form payoff table. (b) A bot chases two agents around a table. The predator’s prey can only escape if the other prey simultaneously moves out of the way. Selfish (top), cooperative (bottom).

3.4 Coin Dilemma

In the Coins game [8, 22], two agents move on a fully-observed $5 \times 5$ gridworld, on which coins of two types corresponding to each agent randomly spawn at each time step with probability 0.005. When an agent moves into a square with a coin of either type, they get a...
We provide additional results on Cleanup, a five-player gridworld game [17]. Agents are rewarded for eating apples, but must keep a river clean to ensure apples receive sufficient nutrients. The option to freeload and only eat apples presents a social dilemma. D3C increases both welfare and individual reward over A2C (no loss mixing). We also observe that direct welfare maximization (Cooperation always results in three agents collecting rewards from apples while two agents sacrifice themselves and clean the river. In contrast, D3C avoids this stark division of labor. Agents take turns on each task and all achieve some positive cumulative return.

## 4 CONCLUSION

We formulate learning incentives as a price of anarchy minimization problem and propose a decentralized, gradient-based approach (D3C) that incrementally adapts agent incentives to the environment at hand. We demonstrate its effectiveness on achieving near-optimal agent outcomes in socially adversarial environments.

It is conceptually possible to scale our approach to very large populations through randomly sharing incentives according to the learned mixture weights or sparse optimization over the simplex [20, 24, 32], but we leave this challenge to future work.

### ACKNOWLEDGMENTS

We are grateful to Jan Balaguer for fruitful discussions and advice on revising parts of the manuscript.

### REFERENCES


