This article has been accepted for publication in IEEE Control Systems Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2026.3704525

Learning generalized Nash equilibria from pairwise preferences

Pablo Krupa, Alberto Bemporad

.

***Abstract*— Generalized Nash Equilibrium Problems (GNEPs) arise in many applications, including non-cooperative multi-agent control problems. Although many methods exist for finding generalized Nash equilibria, most of them rely on assuming knowledge of the objective functions or being able to query the best responses of the agents. We present a method for learning solutions of GNEPs only based on querying agents for their preference between two alternative decisions. We use the collected preference data to learn a GNEP whose equilibrium approximates a GNE of the underlying (unknown) problem. Preference queries are selected using an active-learning strategy that balances exploration of the decision space and exploitation of the learned GNEP. We present numerical results on game-theoretic linear quadratic regulation problems, as well as on other literature GNEP examples, showing the effectiveness of the proposed method.**

***Index Terms*— Game theory, Generalized Nash equilibrium, Preference-based learning, Active learning.**

I. Introduction

GENERALIZED Nash Equilibrium Problems (GNEPs) arise in settings where multiple agents have conflicting, and possibly coupled, interests and constraints [1]. Some examples include economics [2], energy management [3], or control of multi-agent systems [4]. A Generalized Nash Equilibrium (GNE) is a solution of a GNEP in which no individual agent has an incentive to change its decision, given the decisions from other agents [1]. Finding a GNE is a challenging problem that has received a lot of attention from the research community, leading to the development of many methods with different convergence properties under assumptions of the objective functions and constraints of the GNEP. When the objective functions of the agents are known, a GNE can be found using optimization-based methods such as operator splitting methods [5] or interior-point methods [6]. Distributed methods have also been proposed [5], [7], [8], where each agent knows its own objective function, but full centralized knowledge is not required.

An alternative approach when full knowledge of the objective functions is not available is to learn a GNE using data obtained by querying either the objective functions or best responses of the agents. Although literature on learning GNE from data is less extensive, several articles consider this setting. In [9], the authors present a simulated annealing routine to learn an approximate Nash equilibrium using best response evaluations. In [10] and [11] the authors propose Active Learning (AL) [12] strategies for learning GNEs by querying best responses. In [13], equilibrium points of GNEPs with discrete search space are learned using a Bayesian optimization [14] method that queries the objective functions using two proposed exploration-exploitation approaches. An alternative Bayesian optimization approach to learn Nash equilibria, also based on querying objective function values, is presented in [15]. In [16], the authors propose an iterative data-driven approach to learn a GNE of the discrete-time game-theoretic Linear Quadratic Regulator (LQR) problem using random exploration of control actions.

A requirement of these data-based approaches is direct access to evaluations of the objective functions or best response solutions. In contrast, preference-based learning [17] is a machine learning strategy that leverages information in the form of preferences, providing an alternative approach to learn a GNE when direct evaluations of the objective functions or best responses are unavailable. Although preference-based learning has been mostly applied to train large language models [18], [19], it has also been applied to control-related problems, such as tuning proportional-integral controllers [20] or model predictive controllers [21]–[23].

In this paper, we propose a preference-based learning approach to find a GNE that only relies on iteratively querying each agent for their preference between two different decisions, given the other agents' decisions. We use the collected preference data to learn the objective functions of a GNEP whose equilibrium approximates a GNE of the underlying (unknown) GNEP. To achieve this, we select preference queries using an AL strategy that balances exploration of the decision space and exploitation of the learned GNEP problem. At each iteration of the proposed AL method, we query each agent for their preference, and then update the objective functions of the learned GNEP. In contrast with previous works, the proposed approach does not require knowledge of the agents' objective functions, nor queries of the objective functions or best response of the agents. Additionally, the method does not attempt to learn surrogate functions of the objective functions, which can be useful in settings where privacy is desired. We present numerical results highlighting the effectiveness of the proposed method, both on GNEP problems taken from the literature, as well as on game-theoretic

This work was funded by the European Union (ERC Advanced Research Grant COMPACT, No. 101141351). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Corresponding author: Pablo Krupa. The authors are with the IMT School for Advanced Studies, Lucca, Italy. Email: {pablo.krupa, alberto.bemporad}@imtlucca.it

LQR problems. An open source Python implementation of the proposed method is available at https://github.com/pablokrupa/prefGNEP.

The remainder of this paper is organized as follows. Section II introduces the problem setting and shows how we apply preference-based learning to GNEPs. The proposed AL preference-based method for finding a GNE is presented in Section III. We show numerical results in Section IV and conclude the paper in Section V.

Notation: Given $x, y \in \mathbb{R}^n$, $x \leq (\geq)\ y$ denotes componentwise inequalities. The exponential function is denoted by $\exp(\cdot)$. The set of natural numbers (including 0) is denoted by $\mathbb{N}$. For $i, j \in \mathbb{N}$ with $i \leq j$, $\mathbb{N}_i^j \doteq \{i, i+1, \dots, j\}$.

II. Learning a GNE from preferences

Consider a GNEP with n decision variables in which N agents take decisions individually, each one responding with a decision vector

$$x_i^*(x_{-i}) \in \arg\min_{x_i \in \mathcal{X}_i} J_i(x_i, x_{-i}) \tag{1a}$$
$$\text{s.t. } g(x) \leq 0, \tag{1b}$$
$$h(x) = 0, \tag{1c}$$

where $x_i \in \mathbb{R}^{n_i}$ is the decision vector of agent i, $\sum_{i=1}^N n_i = n$, $x_{-i} \in \mathbb{R}^{n-n_i}$ is the vector collecting all other agents' decisions x_j, $j \in \mathbb{N}_1^N$, $j \neq i$, $x = (x_1, \dots, x_N) \in \mathbb{R}^n$ is the stacked vector of all agents' decision variables[1], functions $J_i \colon \mathbb{R}^{n_i} \times \mathbb{R}^{n-n_i} \to \mathbb{R}$ are the objective functions of each agent, $g \colon \mathbb{R}^n \to \mathbb{R}^{n_g}$, $h \colon \mathbb{R}^n \to \mathbb{R}^{n_h}$ define global constraints, and the compact sets $\mathcal{X}_i \subset \mathbb{R}^{n_i}$ are local constraints.

A point $x^* = (x_1^*, x_2^*, \dots x_N^*)$ is a GNE of (1) if, for each $i \in \mathbb{N}_1^N$, $J_i(x_i^*, x_{-i}^*) \leq J_i(x_i, x_{-i}^*)$, $\forall x_i \in \mathcal{F}_i(x_{-i}^*)$, where

$$\mathcal{F}_i(x_{-i}) \doteq \{x_i \in \mathcal{X}_i \colon g((x_i, x_{-i})) \leq 0,\ h((x_i, x_{-i})) = 0\}.$$

We assume that the objective functions $J_i(x_i, x_{-i})$ are unknown, and the constraint sets $\mathcal{X}_i$ and functions g and h are known. We take as a standing assumption that problem (1) has at least one GNE. Additionally, we consider that the best responses $x_i^*(x_{-i})$ cannot be measured directly. However, we have access to measurements of preferences of the form

$$\pi_i(x_i^1, x_i^2; x_{-i}) = \begin{cases} 1 & \text{if } J_i(x_i^1, x_{-i}) \leq J_i(x_i^2, x_{-i}) \\ 0 & \text{otherwise,} \end{cases}$$

which indicates the preference between two decision vectors x_i^1 and x_i^2 given the other agents' decisions x_{-i}.

Next, consider functions $\hat{J}_i \colon \mathbb{R}^{n_i} \times \mathbb{R}^{n-n_i} \to \mathbb{R}$ parameterized by $\theta_i \in \mathbb{R}^{n_{\theta_i}}$, and the resulting GNEP problem

$$\hat{x}_i^*(\hat{x}_{-i}) \in \arg\min_{x_i \in \mathcal{X}_i} \hat{J}_i(x_i, \hat{x}_{-i}; \theta_i) \tag{2a}$$
$$\text{s.t. } g(x) \leq 0, \tag{2b}$$
$$h(x) = 0. \tag{2c}$$

Our objective is to learn parameters θ_i of functions $\hat{J}_i$ so that the GNE $\hat{x}^*$ of (2) approximates a GNE of (1). Note that this objective is achieved if all $\hat{J}_i = J_i$, as (1) and (2) would be the same. That is, if we considered functions $\hat{J}_i$ as *surrogate functions* that we train to match J_i using data. This approximation can be achieved using global optimization techniques, such as Bayesian optimization [14], if evaluations of the objective functions are available, see [13], [15]. In this paper, however, we take an alternative approach that does not require direct queries of the objective functions nor best responses of the agents. Instead, our approach is based on training functions $\hat{J}_i$ to obtain classifiers of the preferences between pairwise strategies as follows.[2]

Consider datasets $\mathcal{D}_i = \{x_i^{j,1}, x_i^{j,2}, x_{-i}^j, \pi_i^j\}_{j=0}^M$ for each agent $i \in \mathbb{N}_1^N$, where M is the size of the dataset, $x_i^{j,1}$ and $x_i^{j,2}$ are the two options given to agent i for the context x_{-i}^j, and $\pi_i^j \doteq \pi_i(x_i^{j,1}, x_i^{j,2}; x_{-i}^j)$ is the corresponding preference query. We train θ_i so as to maximize the satisfaction of

$$\pi_i(x_i^{j,1}, x_i^{j,2}; x_{-i}^j) = 1 \iff \hat{J}_i^1 \leq \hat{J}_i^2,\ \forall j \in \mathbb{N}_1^M, \tag{3}$$

where $\hat{J}_i^1 \doteq \hat{J}_i(x_i^{j,1}, x_{-i}^j; \theta_i)$ and $\hat{J}_i^2 \doteq \hat{J}_i(x_i^{j,2}, x_{-i}^j; \theta_i)$. We do this by posing a logistic regression classification problem. Consider the sigmoid function

$$P_i(x_i^1, x_i^2, x_{-i}) = \frac{1}{1 + \exp\left(\dfrac{\hat{J}_i^1 - \hat{J}_i^2}{d_i(x_i^1, x_i^2)}\right)}, \tag{4}$$

where $d_i \colon \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{R}$ is a dissimilarity function that we add to improve classification accuracy when $x_i^1 \simeq x_i^2$ (see Remark II.2). We take

$$d_i(x_i^1, x_i^2) = \log(\|x_i^1 - x_i^2\|_\infty + 1 + \epsilon_d), \tag{5}$$

where $\epsilon_d > 0$ is some small number that is added to avoid division by 0 in (4). Using (4), we pose the learning problem

$$\min_{\theta_i \in \Theta_i} r_i(\theta_i) + \frac{1}{M} \sum_{j=1}^M \mathcal{L}(\pi_i^j, P_i(x_i^{j,1}, x_i^{j,2}, x_{-i}^j)), \tag{6}$$

where $r_i \colon \mathbb{R}^{n_{\theta_i}} \to \mathbb{R}$ is a regularization term for θ_i (typically an ℓ_2 or ℓ_1 penalization, e.g., $r_i(\theta_i) = \rho_i \|\theta_i\|_2^2$ for $\rho_i > 0$), $\mathcal{L} \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the cross-entropy loss

$$\mathcal{L}(p, \hat{p}) = -p \log(\hat{p}) - (1-p) \log(1-\hat{p})$$

measuring the likelihood of the prediction $\hat{p} \in \mathbb{R}$ matching the target $p \in \{0, 1\}$, and $\Theta_i \doteq \{\theta_i \colon \underline{\theta_i} \leq \theta_i \leq \overline{\theta_i}\}$ are non-empty (and possibly non-compact) box constraints on the parameters θ_i. Problem (6) can be solved using standard machine learning tools and solvers, e.g. L-BFGS-B [24], Adam [25], or a combination of both [26].

We found that we can learn a GNE of (1) by making $\hat{J}_i$ be a good local preference classifier around a GNE of (1). Indeed, note that the learning problem (6) is a logistic classification problem in which $\hat{J}_i$ are trained to classify the preferences between the pairwise strategies in the datasets $\mathcal{D}_i$. If datasets $\mathcal{D}_i$ contain pairs that are close to a GNE of (1), then we can expect the $\hat{J}_i$ obtained from (6) to be good local classifiers of the GNE. The following section presents an AL loop whose objective is first to make $\hat{J}_i$ be a general approximation of J_i by exploring the whole decision space, so that solutions

[1] We may also write $x = (x_i, x_{-i})$ to highlight dependence on x_i and x_{-i}.

[2] In the sequel we call functions $\hat{J}_i$ *surrogate functions* for convenience, even though, again, the objective is not to train them to match functions J_i.

Algorithm 1: AL for GNE preference-based learning

Hyper-params: $\delta > 0$, $\sigma, \underline{\delta}, \underline{\sigma} \geq 0$, $k_{\max}, p_\delta, p_\sigma \geq 1$.
Input: Initial datasets $\mathcal{D}_i^0$, functions $\hat{J}_i$, d_i, z_i and r_i.
Learn initial θ_i^0 by solving GNEP (6) using $\mathcal{D}_i^0$.
for $k = 1, 2, \ldots, k_{\max}$ **do**
  Select δ^k and σ^k using (10), (13) and Remark III.2.
  Obtain $(x_i^{k,1}, x_{-i}^k)$, by solving the GNEP (9).
  Obtain $\hat{x}_i^{k,2}$ by solving problems (11).
  Obtain $x_i^{k,2}$ using (12).
  Query preferences $\pi_i^k = \pi_i(x_i^{k,1}, x_i^{k,2}; x_{-i}^k)$.
  $\mathcal{D}_i^k \leftarrow \mathcal{D}_i^{k-1} \cup \{(x_i^{k,1}, x_i^{k,2}, x_{-i}^k, \pi_i^k)\}$.
  Update θ_i^k by solving (6) using $\mathcal{D}_i^k$.

Approximate GNE of (1)**:** GNE of (2) using $\theta_i^{k_{\max}}$.

of (2) resemble solutions of (1), and then to make $\hat{J}_i$ be a good local classifier of a GNE of (1), while simultaneously making solutions of (2) converge towards solutions of (1).

Remark II.1 (On the existence of a GNE for problem (2))**.** *A requirement of the proposed method is for problem (2) to possess a GNE solution. We note that g, h and $\mathcal{X}_i$ are assumed to be known. Therefore, under certain constraint qualification conditions, functions $\hat{J}_i$ and the constraint sets Θ_i of parameters θ_i may be selected to ensure this. We refer the reader to [1] for conditions on the existence of a GNE.*

Remark II.2 (On the inclusion of the dissimilarity functions)**.** *One of the contributions of this paper is the inclusion of the dissimilarity functions d_i in (4). The reason for their inclusion is to improve the classification accuracy of the learned functions $\hat{J}_i$ between points (x_i^1, x_{-i}) and (x_i^2, x_{-i}) in which $\|x_i^1 - x_i^2\|$ is small, without affecting its accuracy when $\|x_i^1 - x_i^2\|$ is larger. Note that, if $\|x_i^1 - x_i^2\| = 0$, then $P_i(x_i^1, x_i^2, x_{-i}) = 0.5$ for any function $\hat{J}_i$, as you always have $\hat{J}_i^1 = \hat{J}_i^2$. Similarly, if $\|x_i^1 - x_i^2\|$ is very small, the minimization of the cross-entropy loss leads to a function $\hat{J}_i$ with a very large Lipschitz constant around x_i^1, so that small differences in the function argument push P_i towards either 1 or 0 (note that π_i is either 1 or 0). We include the dissimilarity functions d_i to alleviate this issue. When $\|x_i^1 - x_i^2\|$ is small, we have that $d_i(x_i^1, x_i^2)$ is small, and thus we allow small differences in $\hat{J}_i$ to indicate strong preference between x_i^1 or x_i^2, according to the preference model (4). On the other hand, as $\|x_i^1 - x_i^2\|$ increases, we require a larger difference between the values of $\hat{J}_i$. Finally, we note that in this paper we take d_i as (5), as we find that it provides the best results, although other choices of the dissimilarity function are possible, such as $d_i(x_i^1, x_i^2) = \|x_i^1 - x_i^2\|_2 + \epsilon_d$, or $d_i(x_i^1, x_i^2) = \sqrt{\|x_i^1 - x_i^2\|_2} + \epsilon_d$.*

III. Active learning method for finding a GNE

Let $\mathcal{D}_i^0 = \{x_i^{j,1}, x_i^{j,2}, x_{-i}^j, \pi_i^j\}_{j=0}^{M_0}$ be the initial datasets, with $x_i^{j,1}, x_i^{j,2} \in \mathcal{X}_i$, and where $x^{j,\ell} = (x_i^{j,\ell}, x_{-i}^j)$, $\ell \in \mathbb{N}_1^2$, satisfy global constraints, i.e., $g(x^{j,\ell}) \leq 0$ and $h(x^{j,\ell}) = 0$. These initial datasets can be obtained by randomly sampling points satisfying the local and global constraints.

The proposed AL method is an iterative loop, where at each iteration k we add a new sample to each dataset using the exploration-exploitation approach presented in the sequel. This leads to datasets $\mathcal{D}_i^k = \{x_i^{j,1}, x_i^{j,2}, x_{-i}^j, \pi_i^j\}_{j=0}^{M_k}$, where $M_k = M_{k-1} + 1$. At each iteration k, we solve the learning problem (6) using datasets $\mathcal{D}_i^k$. The AL loop presented in this section is summarized in Algorithm 1, where $k_{\max} \geq 1$ is the user-selected number of AL iterations. The output of Algorithm 1 is the proposed approximate GNE of (1). We note that this learned approximate GNE always satisfies the local and global constraints, as it is a solution of (2).

We consider pure exploration functions $z_i^k : \mathbb{R}^{n_i} \to \mathbb{R}$ for each agent i that we want to maximize for promoting exploration of the decision space. Examples of exploration functions are the IDW function [27] based on the already collected data $\{x_i^{j,1}, x_i^{j,2}\}_{j=1}^{M_k}$, or the space-filling function

$$z_i^k(x_i) = \min_{\ell \in \{1,2\}, j \in \mathbb{N}_1^{M_k}} \|x_i - x_i^{j,\ell}\|. \tag{7}$$

A simpler alternative is to consider the concave quadratic function

$$z_i^k(x_i) = -\frac{1}{2}\|x_i - \bar{x}_i^k\|_2^2 \tag{8}$$

where points $\bar{x}_i^k$ are either randomly sampled from $\mathcal{X}_i$ or $\bar{x}_i^k = \arg\max_{x_i \in \mathcal{X}_i} \bar{z}_i^k(x_i)$ and $\bar{z}_i^k$ is any of the aforementioned exploration functions. At each iteration k, we generate a new query point $x^k = (x_1^k, \ldots, x_N^k)$ by solving the GNEP

$$x_i^k(x_{-i}^k) \in \arg\min_{x_i \in \mathcal{X}_i} \hat{J}_i(x_i, x_{-i}^k; \theta_i^{k-1}) - \delta^k z_i^k(x_i) \tag{9a}$$
$$\text{s.t. } g(x) \leq 0, \tag{9b}$$
$$h(x) = 0, \tag{9c}$$

where $\delta^k > 0$ is a weighting factor trading off between exploration and exploitation. A typical approach in AL methods is to start with a large exploration term that is reduced as k increases, see, e.g., [21]. The idea is to promote exploration at the beginning of the AL loop, when the surrogate functions $\hat{J}_i$ are (presumably) poor candidates, and then rely more on exploitation of functions $\hat{J}_i$ as the iteration counter k increases. In [21], the authors propose a linear decay function for δ^k. However, we find that better results are typically obtained in our problem setting if we use an exponential decay function of the form

$$\delta^k = \delta\left(1 - \frac{k}{k_{\max}}\right)^{p_\delta}, \tag{10}$$

where $\delta > 0$ and $p_\delta \geq 1$ are user-selected hyperparameters. If $p_\delta = 1$, we recover the linear decay rate used in [21]. The datasets $\mathcal{D}_i^k$ are obtained by augmenting $\mathcal{D}_i^{k-1}$ with the new samples $(x_i^{k,1}, x_i^{k,2}, x_{-i}^k, \pi_i^k)$, where $x_i^{k,1} = x_i^k$ are the solutions obtained from solving (9), $\pi_i^k = \pi_i(x_i^{k,1}, x_i^{k,2}; x_{-i}^k)$, and we take each $x_i^{k,2}$ by first computing the best response for the x_{-i}^k obtained from solving (9), according to the current surrogate function $\hat{J}_i$, i.e., by solving problems

$$\hat{x}_i^{k,2} \in \arg\min_{x_i \in \mathcal{X}_i} \hat{J}_i(x_i, x_{-i}^k; \theta_i^{k-1}) \tag{11a}$$
$$\text{s.t. } g((x_i, x_{-i}^k)) \leq 0, \tag{11b}$$
$$h((x_i, x_{-i}^k)) = 0, \tag{11c}$$

and then taking

$$x_i^{k,2} = \hat{x}_i^{k,2} + \sigma^k w^k \|\hat{x}_i^{k,2}\|_\infty, \tag{12}$$

where each element of $w^k \in \mathbb{R}^{n_i}$ is sampled from a uniform distribution in the interval $[-0.5, 0.5]$, and

$$\sigma^k = \sigma \left(1 - \frac{k}{k_{\max}}\right)^{p_\sigma} \tag{13}$$

for some user-selected hyperparameters $\sigma \geq 0$ and $p_\sigma \geq 1$. The proposed AL approach compares $x_i^{k,1}$, which is the solution of a GNEP (9) that combines the current surrogate $\hat{J}_i$ with an exploration term z_i^k, with $x_i^{k,2}$, which is a noise-altered best response (without exploration term) for the solution obtained from (9). If $\delta = \sigma = 0$, then $x_i^{k,1} = x_i^{k,2}$.

Remark III.1 (On adding noise to $x_i^{k,2}$)**.** *We note that a more straightforward approach would be to take $x_i^{k,2} = \hat{x}_i^{k,2}$, i.e., as the best response for the solution x_i^k of the GNEP (9). Instead, we add some random noise to $x_i^{k,2}$ in (12), normalized by the magnitude of $x_i^{k,2}$. The reason is that by adding this noise we obtain a vector $(x_i^{k,2}, x_{-i}^k)$ that might violate constraints. Although this might seem counter-intuitive, note that we approach the problem of learning a GNE of (1) as a classification problem. That is, we seek to learn functions $\hat{J}_i$ that locally classify preferences on J_i around solutions of the GNEP (2). We therefore want functions $\hat{J}_i$ to classify correctly at the boundary of the constraints when (1) has a GNE with active constraints. Adding a (relatively small) random noise to $x_i^{k,2}$ allows us to learn a GNE with active constraints, as it provides information about J_i on points that are unfeasible but close to the boundary of the constraints. The noise term σ can be set to 0 in problems where querying agents for non-admissible decisions is unreasonable.*

Remark III.2 (Minimum values of δ^k and σ^k)**.** *As discussed in Remark II.2, we include a dissimilarity term d_i in (4) to improve classification performance when $|\hat{J}_i^1 - \hat{J}_i^2|$ is small. However, we can still run into numerical issues when $\hat{J}_i^1$ and $\hat{J}_i^2$ are too similar. Note that, since $\delta^k \to 0$ and $\sigma^k \to 0$ as $k \to \infty$, we can expect $\|x_i^{k,1} - x_i^{k,2}\| \to 0$, and therefore that $|\hat{J}_i^1 - \hat{J}_i^2| \to 0$. To avoid this, we set a user-defined minimum value $\underline{\delta} \geq 0$ for δ^k, i.e., $\delta^k \leftarrow \max(\delta^k, \underline{\delta})$. We do the same for σ^k with a user-defined $\underline{\sigma} \geq 0$, taking $\sigma^k \leftarrow \max(\sigma^k, \underline{\sigma})$.*

IV. Numerical results

We present numerical results using the implementation of Algorithm 1 publicly available at `https://github.com/pablokrupa/prefGNEP`, which uses the `NashOpt` Python package [28] (version `1.1.0`) to solve the GNEP and best response problems in Steps 4 and 5 of Algorithm 1. The numerical results in this section can be reproduced by running the example scripts in the repository. For the learning problem (6), we take the regulation terms as $r(\theta_i) = 0.001\|\theta_i\|_2^2$, and use 500 iterations of Adam (with learning rate of 0.001 and decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$) followed by a maximum of 1000 iterations of the L-BFGS-B solver from the `jaxopt` package (using a history size of 10), initialized from the solution returned by Adam. We use the dissimilarity functions (5) and take the exploration functions z_i^k as (8), with $\bar{x}_i^k$ randomly sampled from the local constraints $\mathcal{X}_i$. We take the surrogate functions as

$$\hat{J}_i(x_i, x_{-i}; \theta_i) = \frac{1}{2} x_i^\top P_i x_i + q_i^\top x_i + x_{-i}^\top A_i x_i, \tag{14}$$

where θ_i contains $q_i \in \mathbb{R}^{n_i}$, $A_i \in \mathbb{R}^{(n-n_i)\times n_i}$ and the non-zero elements of the Cholesky decomposition of the positive definite matrix $P_i \in \mathbb{R}^{n_i \times n_i}$. Lower bounds $\underline{\theta}_i$ can be imposed on θ_i in (6) to ensure that P_i is positive definite, see, e.g., [23]. All the numerical results use $\sigma = 0.3$, $\underline{\delta} = 10^{-4}$, $\underline{\sigma} = 10^{-3}$, $p_\delta = 5$ and $p_\sigma = 3$ as hyperparameters of Algorithm 1, and use initial datasets of size $M_0 = 50$.

A. Game-theoretic linear quadratic regulator

Consider a discrete linear time-invariant system

$$\xi(t+1) = A\xi(t) + Bu(t), \tag{15}$$

where $\xi(t) \in \mathbb{R}^{n_\xi}$ and $u(t) \in \mathbb{R}^m$ are the state and control input at sample time t, $A \in \mathbb{R}^{n_\xi \times n_\xi}$ and $B \in \mathbb{R}^{n_\xi \times m}$. Additionally, we have N agents, each one controlling a portion of the control inputs, such that $u(t) = (u_1(t), u_2(t), \dots u_N(t))$, with $u_i(t) \in \mathbb{R}^{m_i}$ being the control action of agent $i \in \mathbb{N}_1^N$, and $\sum_{i=1}^N m_i = m$. The objective of each agent is to control (15) so as to minimize the Linear Quadratic Regulator (LQR) cost for matrices $Q_i \in \mathbb{R}^{n_\xi \times n_\xi}$ and $R_i \in \mathbb{R}^{m_i \times m_i}$, with $Q_i \succeq 0$ and $R_i \succ 0$, see, e.g. [16, §II] or [28, §5.1] for a more in-depth explanation of this problem setting.

It is well known that this game-theoretic LQR problem admits a solution of the form $u_i(t) = -K_i\xi(t)$, where $K_i \in \mathbb{R}^{n_\xi \times m_i}$ is the state-feedback gain for agent i, see, e.g., [16]. We compute approximate gains K_i by solving the GNEP [28]

$$\begin{aligned} K_i^*(K_{-i}) \in \arg\min_{K_i} \; & \sum_{j=0}^{T} \xi_j^\top Q_i \xi_j + u_{i,j}^\top R_i u_{i,j} \\ \text{s.t. } & \xi_{j+1} = (A - BK_{-i})\xi_j + B_i u_{i,j} \\ & u_{i,j} = -K_i \xi_j, \end{aligned} \tag{16}$$

where a more accurate solution of the infinite horizon LQR game is obtained as $T > 0$ is increased. Problem (16) can be posed as (1) by taking J_i as the squared Frobenius norm of the deviation between K_i and the best response $K_i^*(K_{-i})$ for agent i, i.e., as the best response deviation

$$J_i(K_i, K_{-i}) = \|K_i^*(K_{-i}) - K_i\|_F^2. \tag{17}$$

We consider three examples, each with the number of agents N and system dimensions $n_\xi = m$ shown in each row of Table I. We take $m_i = m/N$, $R_i = I_{m_i}$ and Q_i as a matrix of zeros with ones in the diagonal elements corresponding to states $(i-1)m_i + 1$ to im_i. Matrices A and B of (15) are randomly generated so that A is an unstable matrix with spectral radius equal to 1.1. Taking $T = 50$, we obtain a solution of problem (16) using the `NashLQR` solver from the `NashOpt` Python package [28], which considers (17).

For each example, we learn a GNE of the game-theoretic LQR problem using Algorithm 1 with $\delta = 5$, where x_i and

Parameters	$k_{\max} = 100$ RMSE	$k_{\max} = 100$ $\max_i J_i$	$k_{\max} = 200$ RMSE	$k_{\max} = 200$ $\max_i J_i$
$n_\xi = m = 6,\ N = 3$	0.00343	0.0970	0.00109	0.0202
$n_\xi = m = 8,\ N = 3$	0.00675	0.0596	0.00288	0.0144
$n_\xi = m = 12,\ N = 4$	0.01109	0.3009	0.01229	0.2152

TABLE I: Results for game-theoretic LQRs

x_{-i} are the vectorized form of K_i and K_{-i}, We denote by $\hat{K}_i^k$ the gain matrices learned by Algorithm 1 at iteration k.

We note that we consider the objective functions (17) to provide a numerical example in which the preference queries π_i use the same objective function that is used in the `NashOpt` package, which we use as our back-end GNEP solver. In a practical setting, the preference queries π_i would be taken by asking each agent for their preference between two alternative controllers, e.g., by performing a closed-loop experiment with each controller. Algorithm 1 only requires the preference information, i.e., the values of the underlying functions J_i driving the preference decision are not required.

Table I shows the results obtained for the three systems when taking $k_{\max} = 100$ and $k_{\max} = 200$. To compare a learned $\hat{K}_i^{k_{\max}}$ with the Nash solution K_i^*, we take 100 random initial states. For each initial state, we perform a closed-loop simulation of length T using $\hat{K}_i^{k_{\max}}$ and K_i^*, and store the costs obtained with each, as measured using the LQR cost in (16). We can then compare $\hat{K}_i^{k_{\max}}$ and K_i^* by computing the normalized Root Mean Square Error (RMSE) between the costs obtained in the 100 tests, where the RMSE is normalized by dividing by the difference between the maximum and minimum costs obtained with K_i^* (so that RMSE values are comparable between the three different system). Table I also shows the maximum value of costs $J_i(\hat{K}_i^{k_{\max}}, \hat{K}_{-i}^{k_{\max}})$ (17) between the N agents, which measures the deviation of the learned $\hat{K}_i^{k_{\max}}$ to a GNE.

Fig. 1 shows the results obtained with the gain matrices $\hat{K}_i^k$ from each iteration k of Algorithm 1, for the case $k_{\max} = 200$ and the system with $n_\xi = 12$, $m = 12$, $N = 4$ (see Table I). Fig. 1a shows the best response deviations J_i of each agent, Fig. 1b the normalized RMSE (using the same 100 initial states used for the results in Table I), and Fig. 1c shows a closed-loop simulation starting from a random initial state using the GNE gains K_i^* obtained from `NashOpt` and the learned gains $\hat{K}_i^{k_{\max}}$. We plot the output $y(t)$ of the system, which we take as the summation of the states. The results shown in Figs. 1a and 1b show that the iterates of Algorithm 1 converge non-monotonically towards a GNE of the underlying GNEP. We note that the results obtained for the other examples listed in Table I are comparable to the ones shown in Fig. 1.

B. Solving GNEPs taken from the literature

We present numerical results using Algorithm 1 to solve two GNEPs taken from [29, Expl. A.3] and [5, Expl. 1]. For the GNEP from [5, Expl. 1], we take the configuration presented in [10, §VI.B], which considers $N = 10$ agents.

Figs. 2a and 2c show the evolution of the solutions x^k of problem (2) for the θ_i^k obtained at each iteration of Algorithm 1, taking $\delta = 0.3$. The results highlight how the algorithm transitions from an initial phase where the exploration term δ^k is dominant, towards one where exploitation and local exploration drive x^k towards a GNE of the underlying GNEP (1).

To evaluate the sensitivity of Algorithm 1 to the choice of δ, we test $\delta \in \{0.1, 0.2, 0.3, 0.4\}$, performing 10 random tests for each value of δ, i.e., with different realization of the initial datasets $\mathcal{D}_i^0$, exploration points $\bar{x}_i^k$ in (8), and noise terms w_k in (12). As seen in Figs. 2a and 2c, the final iterates of Algorithm 1 are influenced by the noise induced by the random variables $\bar{x}_i^k$ and w^k. Therefore, to reduce the effect of the noise, we take the approximate GNE of (1) provided by Algorithm 1 as the average of the solutions of (2) for the last 5 iterates θ_i^k of the algorithm. Let $\hat{x}^*$ denote the output of Algorithm 1 obtained this way and $\tilde{x}_i^*$ its corresponding best response for agent i for the underlying GNEP (1). For each test, we evaluate the maximum normalized best-response deviation among the N agents:

$$\phi(\hat{x}^*) \doteq \max_{i \in \mathbb{N}_1^N} \frac{\|(\tilde{x}_i^*, \tilde{x}_{-i}^*) - (\hat{x}_i^*, \hat{x}_{-i}^*)\|_2}{\|(\tilde{x}_i^*, \tilde{x}_{-i}^*)\|_2}. \tag{18}$$

A value of $\phi(\hat{x}^*) = 0$ indicates that $\hat{x}^*$ is a GNE of (1), with small values of $\phi(\hat{x}^*)$ indicating that $\hat{x}^*$ is a good approximation of a GNE of (1). Figs. 2b and 2d present the results of the sensitivity analysis of δ, showing that the proposed method reliably finds good GNE approximations for different values of δ.

V. Conclusions

We presented an AL method to learn a GNE from preference data only, i.e., with no access to the objective function values or the best responses of the agents. Preferences are used to train the objective functions of a *surrogate* GNEP, whose equilibrium approximates a GNE of the underlying GNEP as the exploitation term of the AL method becomes dominant. By selecting query points that are close to the current GNE estimate, we train the surrogate functions to form a better local preference classifier. Numerical results show that, even though the proposed approach is not guaranteed to converge towards a GNE of (1), it is effective at finding good GNE approximations in practice. An open source implementation of the proposed method can be found at `https://github.com/pablokrupa/prefGNEP`.

References

[1] F. Facchinei and C. Kanzow, "Generalized Nash equilibrium problems," *Annals of Operations Research*, vol. 175, no. 1, pp. 177–211, 2010.

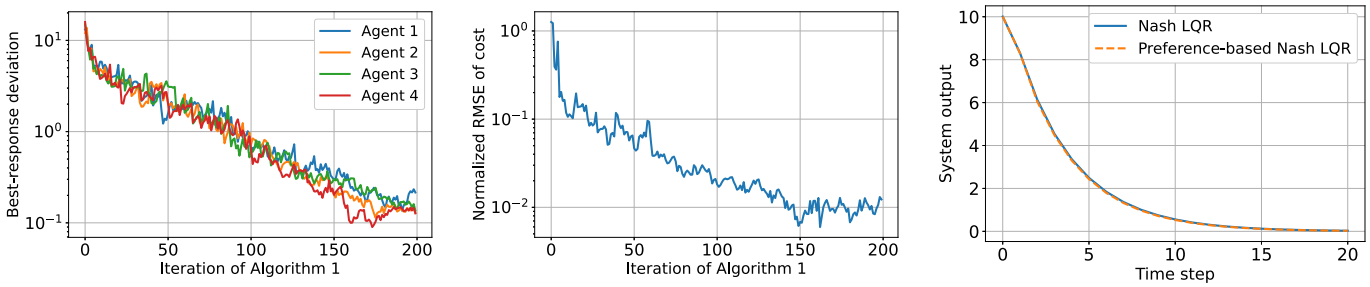
[2] E. Dockner, S. Jørgensen, N. Van Long, and G. Sorger, *Differential games in economics and management science*. Cambridge, U.K.: Cambridge University Press, 2000.

[3] S. Hall, G. Belgioioso, D. Liao-McPherson, and F. Dorfler, "Receding horizon games with coupling constraints for demand-side management," in *IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 3795–3800.

[4] D. Cappello and T. Mylvaganam, "Distributed differential games for control of multi-agent systems," *IEEE Transactions on Control of Network Systems*, vol. 9, no. 2, pp. 635–646, 2022.

[5] F. Salehisadaghiani, W. Shi, and L. Pavel, "An ADMM approach to the problem of distributed nash equilibrium seeking," *CoRR*, 2017.

This article has been accepted for publication in IEEE Control Systems Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2026.3704525

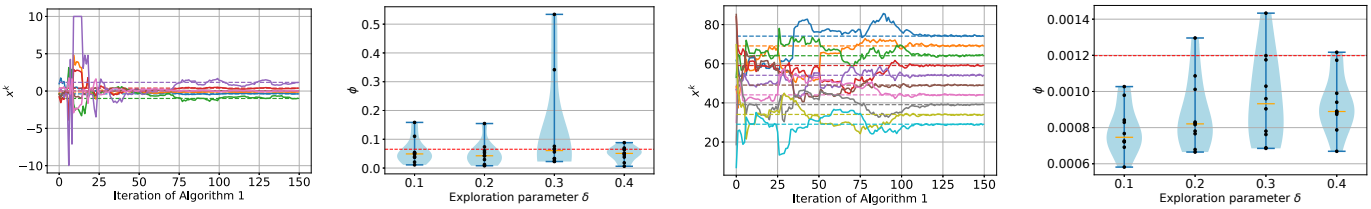


(a) Best response deviation.

(b) Normalized RMSE of closed-loop cost.

(c) Closed-loop trajectories.

Fig. 1: Results of Algorithm 1 for game-theoretic LQR problem with $n_\xi = m = 12$ and $N = 4$.



(a) GNEP from [29, Expl. A.3].

(b) ϕ for [29, Expl. A.3].

(c) GNEP from [5, Expl. 1] for $N = 10$.

(d) ϕ for [5, Expl. 1] for $N = 10$.

Fig. 2: Figures (a) and (c) show the values of x^k obtained from solving (2) for the θ_i^k learned at each iteration k of Algorithm 1 (solid lines) converging towards a GNE of the underlying GNEP (dashed lines). Figures (b) and (d) show the maximum normalized best-response deviations ϕ (18) of the tests for different values of δ. Orange lines indicate the median values. For reference, the dashed red lines indicate the value of ϕ obtained in Figures (a) and (c).

[6] A. Dreves, F. Facchinei, C. Kanzow, and S. Sagratella, "On the solution of the KKT conditions of generalized Nash equilibrium problems," *SIAM Journal on Optimization*, vol. 21, no. 3, pp. 1082–1108, 2011.

[7] T. Tatarenko and M. Kamgarpour, "Learning generalized Nash equilibria in a class of convex games," *IEEE Transactions on Automatic Control*, vol. 64, no. 4, pp. 1426–1439, 2019.

[8] G. Carnevale, F. Fabiani, F. Fele, K. Margellos, and G. Notarstefano, "Distributed equilibrium seeking in aggregative games: linear convergence under singular perturbations lens," in *IEEE 63rd Conference on Decision and Control (CDC)*, 2024, pp. 3918–3923.

[9] Y. Vorobeychik and M. P. Wellman, "Stochastic search methods for nash equilibrium approximation in simulation-based games," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, vol. 2, 2008, pp. 1037–1044.

[10] F. Fabiani and A. Bemporad, "An active learning method for solving competitive multiagent decision-making and control problems," *IEEE Transactions on Automatic Control*, vol. 70, no. 4, pp. 2374–2389, 2025.

[11] B. Franci, F. Fabiani, and A. Bemporad, "Actively learning equilibria in Nash games with misleading information," *IEEE Control Systems Letters*, vol. 9, pp. 312–317, 2025.

[12] M. Prince, "Does active learning work? A review of the research," *Journal of Engineering Education*, vol. 93, no. 3, pp. 223–231, 2004.

[13] V. Picheny, M. Binois, and A. Habbal, "A Bayesian optimization approach to find Nash equilibria," *Journal of Global Optimization*, vol. 73, no. 1, pp. 171–192, 2019.

[14] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv:1012.2599*, 2010.

[15] A. Al-Dujaili, E. Hemberg, and U.-M. O'Reilly, "Approximating nash equilibria for black-box games: A bayesian optimization approach," *arXiv preprint arXiv:1804.10586*, 2018.

[16] B. Nortmann, A. Monti, M. Sassano, and T. Mylvaganam, "Nash equilibria for linear quadratic discrete-time dynamic games via iterative and data-driven algorithms," *IEEE Transactions on Automatic Control*, vol. 69, no. 10, pp. 6561–6575, 2024.

[17] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz, "A survey of preference-based reinforcement learning methods," *Journal of Machine Learning Research*, vol. 18, no. 136, pp. 1–46, 2017.

[18] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving, "Fine-tuning language models from human preferences," *arXiv:1909.08593*, 2019.

[19] N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. Christiano, "Learning to summarize with human feedback," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 3008–3021.

[20] J. P. Coutinho, I. Castillo, and M. S. Reis, "Human-in-the-loop controller tuning using preferential Bayesian optimization," in *IFAC-PapersOnLine*, vol. 58, no. 14, 2024, pp. 13–18.

[21] M. Zhu, D. Piga, and A. Bemporad, "C-GLISp: Preference-based global optimization under unknown constraints with applications to controller calibration," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 5, pp. 2176–2187, 2022.

[22] K. Shao, A. Chakrabarty, A. Mesbah, and D. Romeres, "Coactive preference-guided multi-objective Bayesian optimization: An application to policy learning in personalized plasma medicine," *IEEE Control Systems Letters*, vol. 8, pp. 3081–3086, 2024.

[23] P. Krupa, H. El Hasnaouy, M. Zanon, and A. Bemporad, "Learning the MPC objective function from human preferences," in *IEEE 64th Conference on Decision and Control (CDC)*, 2025, pp. 6474–6479.

[24] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on scientific computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] A. Bemporad, "An L-BFGS-B approach for linear and nonlinear system identification under ℓ_1 and group-lasso regularization," *IEEE Transactions on Automatic Control*, vol. 70, no. 7, pp. 4857–4864, 2025.

[27] A. Bemporad and D. Piga, "Active preference learning based on radial basis functions," *Machine Learning*, vol. 110, no. 2, pp. 417–448, 2021.

[28] A. Bemporad, "Nashopt - A Python library for computing generalized Nash equilibria," *arXiv preprint arXiv:2512.23636*, 2025.

[29] F. Facchinei and C. Kanzow, "Penalty methods for the solution of generalized Nash equilibrium problems (with complete test problems)," *Sapienza University of Rome*, 2009.