Coordination in Noncooperative Multiplayer Matrix Games via Reduced Rank Correlated Equilibria

Jaehan Im, Yue Yu, David Fridovich-Keil, and Ufuk Topcu

Abstract— Coordination in multiplayer games enables players to avoid the lose-lose outcome that often arises at Nash equilibria. However, designing a coordination mechanism typically requires the consideration of the joint actions of all players, which becomes intractable in large-scale games. We develop a novel coordination mechanism, termed *reduced rank correlated equilibria*. The idea is to approximate the set of all joint actions with the actions used in a set of pre-computed Nash equilibria via a convex hull operation. In a game with n players and each player having m actions, the proposed mechanism reduces the number of joint actions considered from $\mathcal{O}(m^n)$ to $\mathcal{O}(mn)$ and thereby mitigates computational complexity. We demonstrate the application of the proposed mechanism to an air traffic queue management problem. Compared with the correlated equilibrium—a popular benchmark coordination mechanism—the proposed approach is capable of solving a problem involving four thousand times more joint actions while yielding similar or better performance in terms of a fairness indicator and showing a maximum optimality gap of 0.066% in terms of the average delay cost. In the meantime, it yields a solution that shows up to 99.5% improvement in a fairness indicator and up to 50.4% reduction in average delay cost compared to the Nash solution, which does not involve coordination.

I. INTRODUCTION

In a multiplayer game, each player minimizes its own cost by choosing its strategy—a probability distribution over its available actions—based on the other players' strategies. At a *Nash equilibrium*, no player can reduce its cost by unilaterally changing its current strategy. Multiplayer games enable the modeling of noncooperative interactions in various applications, including air traffic management systems.

Coordination mechanisms in games allow players to avoid *lose-lose outcomes*, which often occur in competitive interactions. The prisoner's dilemma and the chicken game [1] are typical examples of such games. Coordination mechanisms help players avoid lose-lose outcomes by recommending joint actions to each player through a coordinator.

A *correlated equilibrium* in a multiplayer game is a probability distribution over all joint actions of all players [2] which provides a coordination mechanism as follows. Suppose a coordinator samples a joint action according to a correlated equilibrium known to all players and recommends it to all players. Then, each player cannot reduce its own cost by unilaterally deviating from the recommendation. Several studies have shown that the correlated equilibrium is useful

J. Im, and D. Fridovich-Keil are with the Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, TX, 78712, USA (emails: jaehan.im@utexas.edu, dfk@utexas.edu). Y. Yu, and U. Topcu are with the Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, TX, 78712, USA (emails: yueyu@utexas.edu, utopcu@utexas.edu).
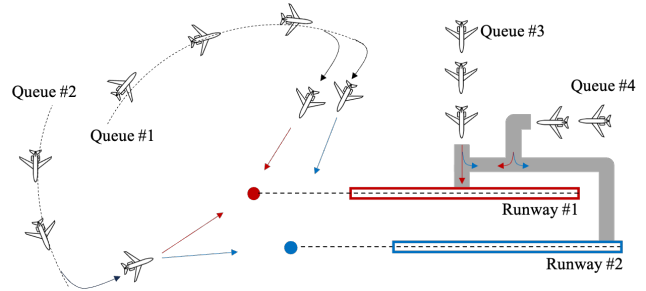
Fig. 1: Illustration of a multiple departure-arrival queue coordination scenario. Each player (queue) may occupy both runways, a single runway, or yield to others.

for coordinating chicken games [1], 2-by-2 games [3], Battle-of-Sexes games [4], and public goods game [5].

However, computing a correlated equilibrium becomes intractable as the number of players and actions increases, because it requires considering all joint actions. In a game with n players where each player has m actions, the number of all joint actions is m^n. This makes the computation of correlated equilibria intractable as n and m increase, which limits its usage in large-scale coordination problems [1].

We develop an algorithm that efficiently computes correlated equilibria based on a novel concept termed *reduced rank correlated equilibria (RRCE)*. The set of RRCE inhabits the convex hull of multiple Nash equilibria, and this set approximates the set of all possible correlated equilibria. The proposed algorithm first computes multiple Nash equilibria and then obtains a set of RRCE. Unlike the correlated equilibrium, the computation of the Nash equilibrium only requires the consideration of the actions of individual players in isolation rather than all joint actions. As a result, the proposed algorithm reduces the number of joint actions considered during the correlated equilibrium computation from $\mathcal{O}(m^n)$ to $\mathcal{O}(mn)$.

We evaluate the proposed algorithm by conducting a series of Monte Carlo simulations on an air traffic management problem, which typically requires coordination for efficient operation. The proposed algorithm solves a problem involving a four thousand times larger number of joint actions than a direct computation of the correlated equilibrium can solve. The proposed algorithm yields a solution that shows a 0.066% gap (at worst) to the correlated equilibrium regarding the average cost per player. Yet our approach demonstrates superior fairness and average cost per player compared to a Nash solution that does not involve coordination.

II. RELATED WORKS

The challenge of obtaining the correlated equilibrium in large-scale problems is widely acknowledged [1], [6], [7]. One research direction that addresses this issue is to utilize an evolutionary algorithm to obtain the correlated equilibrium [8], [9], [6]. These algorithms let players gradually converge to a correlated equilibrium through repeated games while players are set to follow a specific regret-minimization rule. However, this approach converges to a random correlated equilibrium, preventing the user from obtaining an equilibrium that fits their interests.

Learning dynamics is another approach to bypass the direct computation of the correlated equilibrium. This approach learns a strategy converging to a *coarse correlated equilibrium*, a relaxed notion of correlated equilibrium, that minimizes the sum of players' costs [10], [11], [12]. However, a coarse correlated equilibrium is an equilibrium that does not guarantee that the players comply with the recommended actions, making it inappropriate to be applied for coordination purposes.

Various multiplayer games have extensively implemented a Nash equilibrium as coordination [13], [14], [15] and outcome prediction [16] tool. This approach computes an existing Nash equilibrium and uses it as a control command relayed to the players. If multiple Nash equilibria exist, the coordinator chooses the one that best fits the coordinator's interests [13]. However, since the Nash equilibrium does not consider joint actions, it generally yields a less efficient solution than the correlated equilibrium [10], [12].

Approximation of the correlated equilibrium has been attempted in mean-field games [17] and graphical games [18] to improve solution quality and reduce computation burden. However, to our knowledge, no research has focused on both (i) addressing the computational intractability of the correlated equilibrium and (ii) yielding a solution that fits the coordinator's interests.

III. NASH AND CORRELATED EQUILIBRIA IN MATRIX GAMES

A. Matrix games

We consider a game with $n \in \mathbb{N}$ players in which each player has $m \in \mathbb{N}$ actions. In this paper, we denote a ranged set of natural numbers $\{1, 2, \ldots, n\}$ as $\mathbb{N}_{[1,n]}$. Let $-i$ denote the set excluding the i-th element itself, *i.e.*, $\{1, \ldots, n\} \setminus i$. Lastly, let $v \in \mathbb{R}^n$ be a vector and let $[v]_k$ indicate the k-th element of v for $k \in \mathbb{N}_{[1,n]}$.

1) Player's action: Let $a_i \in \mathbb{N}_{[1,m]}$ denote an action of player i and $\boldsymbol{a} := \{a_1, \ldots, a_n\}$ denote a joint action of all players. Let $\mathcal{A}$ be a set of all possible joint actions, $\boldsymbol{a}$. The cardinality of this set, $|\mathcal{A}|$, is m^n.

2) Player's strategy: Let $\Delta_m := \{v \in \mathbb{R}^m_{\geq 0} \mid v^\top \mathbf{1}_m = 1\}$ and $x_i \in \Delta_m$ denote player i's *strategy* as follows:

$$[x_i]_{a_i} := \mathbb{P}[\text{Player } i \text{ taking action } a_i]. \quad (1)$$

3) Joint action distribution: Let $\boldsymbol{z} \in \mathbb{R}^{m \times ... \times m}_{\geq 0}$ denote the *joint action probability distribution* and $\boldsymbol{z_a} \in [0,1]$ denote the probability of specific joint action profile $\boldsymbol{a}$ occuring, *i.e.*,

$$\boldsymbol{z_a} := \mathbb{P}[\text{Players take joint action } \boldsymbol{a}] = \mathbb{P}[\boldsymbol{a}], \quad (2)$$

where $\sum_{\boldsymbol{a}} \boldsymbol{z_a} = 1$.

The joint action probability distribution profile $\boldsymbol{z}$ can be expressed in terms of the players' strategy x_i, *i.e.* the conversion equation is shown below.

$$\boldsymbol{z_a} = \prod_i [x_i]_{[\boldsymbol{a}]_i}. \quad (3)$$

Note that this conversion is only valid from x to $\boldsymbol{z}$. The reverse (decomposition of $\boldsymbol{z}$ to x) may not exist or result in indefinite strategy profiles x.

4) Cost: The cost that player i incurs by taking action a_i when player j takes action a_j (for all $j \neq i$) is given by $\sum_{j \neq i} [C^{ij}]_{a_i a_j}$, where $[C^{ij}]_{a_i a_j}$ is the $a_i a_j$-th element (a_i-th row and a_j-th column) in the cost matrix $C^{ij} \in \mathbb{R}^{m \times m}$.

The cost that player i experiences, c_i, can be expressed as follows:

$$c_i(\boldsymbol{z}) := \sum_{\boldsymbol{a}} \boldsymbol{z_a} \sum_{j \in -i} [C^{ij}]_{[\boldsymbol{a}]_i [\boldsymbol{a}]_j}. \quad (4)$$

Note that one can compute c_i with individual player's strategy x_i by transforming x_i to $\boldsymbol{z}$ via (3).

B. Nash equilibrium (NE)

The Nash equilibrium is a set of individual strategies in which no player can reduce its cost by unilaterally changing its strategy. It exists regardless of the presence of coordination.

Definition 1 (Nash equilibrium). *A Nash equilibrium is a strategy $x^\star$ such that for all $i \in \mathbb{N}_{[1,n]}$,*

$$c_i(x_i^\star, x_{-i}^\star) \leq c_i(x_i, x_{-i}^\star), \forall x_i \in \Delta_m, x_i^\star \neq x_i. \quad (5)$$

Under an appropriate constraint qualification, the Karush-Kuhn-Tucker (KKT) conditions for all players must be satisfied at a Nash equilibrium [19, p.26]. The KKT conditions for player i are

$$\nabla_{x_i} \mathcal{L}_i = \sum_{j \neq -i} C^{ij} x_j - \lambda^i - \mu^i \mathbf{1}_m = \mathbf{0}_m,$$
$$\mathbf{1}_m^\top x_i = 1, \ \lambda^{i\top} x_i = 0, \ x_i \geq \mathbf{0}_m, \ \lambda^i \geq \mathbf{0}_m, \quad (6)$$

where $\lambda^i = [\lambda_1^i, \ldots, \lambda_m^i]^\top \in \mathbb{R}^m$ and $\mu^i \in \mathbb{R}$ are Lagrange multipliers, and $\mathcal{L}_i$ is player i's Lagrangian.

C. Correlated equilibrium (CE)

A correlated equilibrium is a joint action probability distribution, $\boldsymbol{z}$, such that after a particular joint action profile $\boldsymbol{a}^\star$ is drawn from $\boldsymbol{z}$ and each action $a_i^\star \in \boldsymbol{a}^\star$ is recommended to each player i, playing $a_i^\star$ is the optimal choice for each player i. Here, we assume that the players know the distribution $\boldsymbol{z}$.

Definition 2 (Correlated equilibrium). *A correlated equilibrium is a probability distribution $\boldsymbol{z}$ over the set of joint action profiles $\mathcal{A}$ such that*

$$\mathbb{E}[c_i(a_i^\star)|a_i^\star, \boldsymbol{z}] \leq \mathbb{E}[c_i(a_i)|a_i^\star, \boldsymbol{z}], a_i^\star = [\boldsymbol{a}^\star]_i, \quad (7)$$

where $a_i^\star \neq a_i$, for all $i \in \mathbb{N}_{[1,n]}$ and for all joint action profiles $\boldsymbol{a} \in \mathcal{A}$.

The expected cost that player i experiences when the player chooses action a_i with action suggestion $a_i^\star$ given $\boldsymbol{z}$, $\mathbb{E}[c_i(a_i^\star)|a_i^\star, \boldsymbol{z}]$, is denoted with $\Phi_{a_i^\star}^{\boldsymbol{z}}(a_i)$:

$$\Phi_{a_i^\star}^{\boldsymbol{z}}(a_i) = \sum_{\hat{\boldsymbol{a}}} \boldsymbol{z}_{\hat{\boldsymbol{a}}} \sum_{j \neq i} [C^{ij}]_{[\boldsymbol{a}]_i [\boldsymbol{a}]_j}, \qquad (8)$$

where $\hat{\boldsymbol{a}} \in \{\boldsymbol{a} \in \mathcal{A}|[\boldsymbol{a}]_i = a_i^\star\}$. Then, the correlated equilibrium is a joint action distribution $\boldsymbol{z}$ that satisfies the following conditions:

$$\begin{aligned} &\Phi_{a_i^\star}^{\boldsymbol{z}}(a_i^\star) \leq \Phi_{a_i^\star}^{\boldsymbol{z}}(a_i), \ \forall (a_i, a_i^\star) \in \mathbb{N}_{[1,m]}, \ \forall i \in \mathbb{N}_{[1,n]}, \\ &\textstyle\sum_{\boldsymbol{a} \in \mathcal{A}} \boldsymbol{z}_{\boldsymbol{a}} = 1, \ \ \boldsymbol{z}_{\boldsymbol{a}} \geq 0, \ \forall \boldsymbol{a} \in \mathcal{A}. \end{aligned} \qquad (9)$$

IV. REDUCED RANK CORRELATED EQUILIBRIA ALGORITHM

The correlated equilibrium computation becomes quickly intractable as the number of joint actions grows exponentially in the number of players (n). Recall that the number of joint actions considered to compute the correlated equilibrium is $\mathcal{O}(m^n)$ while for the Nash equilibrium, the figure is $\mathcal{O}(mn)$. This affects the computational complexity of solving for each equilibrium. Although computational complexity may vary according to which solver one uses, we provide an example when using the interior point method (IP).

Proposition 1 (Number of linear equations to find NE and CE with IP). *The computation of the Nash equilibrium and the correlated equilibrium with IP involves solving $n(m+1)$ and $2m^n + 3m^2 n + 1$ linear equations, respectively.*

Proof. The interior point (IP) method solves a system of linear equations at every iteration [20, p.393]. The Nash equilibrium is formulated as a linear complementarity problem with $n(m + 1)$ linear equations when solved with IP [19, p.1037]. The correlated equilibrium is formulated as a linear program with $2m^n + 3m^2 n + 1$ linear equations when solved with IP [20, p.394]. $\qquad\square$

To exploit this apparent difference in scaling, we propose an algorithm that approximates the set of correlated equilibria with multiple Nash equilibria.

A. Correlated equilibrium set approximation

Let $\mathcal{G} := \{\boldsymbol{z}|\boldsymbol{z} \text{ satisfies } (9)\}$ denote a set of correlated equilibria. Since (9) comprises only linear constraints, $\mathcal{G}$ is a convex polytope. In addition, the set of all Nash equilibria is a subset of $\mathcal{G}$ [21]. Therefore, the convex hull of Nash equilibria is a subset of the correlated equilibria.

Let a set with d distinct Nash equilibria denoted as $\mathcal{D}$, with the superscript k indicating the k-th Nash equilibrium

$$[\mathcal{D}]_k := \{x_1^k, x_2^k, \ldots, x_n^k\}, k \in \mathbb{N}_{[1,d]}. \qquad (10)$$

It is possible to convert $[\mathcal{D}]_k$ into a joint action probability distribution $\boldsymbol{z}$ using (11). Let $\boldsymbol{z}^k \in \mathbb{R}_{\geq 0}^{m \times .^n_{\cdot} \times m}$ denote the joint action probability distribution of the k-th equilibrium in $\mathcal{D}$ such that

$$\boldsymbol{z}_{\boldsymbol{a}}^k = \textstyle\prod_i [x_i^k]_{[\boldsymbol{a}]_i} \qquad (11)$$

for all $\boldsymbol{a} \in \mathcal{A}$. The convex hull of $\boldsymbol{z}^{k \in \mathbb{N}_{[1,d]}}$ is a subset of the correlated equilibrium set $\mathcal{G}$ [21], [22], *i.e.*

$$\mathcal{H} := \text{conv}(\boldsymbol{z}^k) \subseteq \mathcal{G}, \qquad (12)$$

where the convex hull of $\{\boldsymbol{z}^k\}_{k=1}^d$ is denoted as $\mathcal{H}$ and $\text{conv}(\boldsymbol{z}^k) = \{\sum_{i=1}^d [\boldsymbol{\gamma}]_i \boldsymbol{z}^i \mid \boldsymbol{\gamma} \in \mathbb{R}_{\geq 0}^d, \ \mathbf{1}_d^\top \boldsymbol{\gamma} = 1\}$.

Definition 3 (Rank of a tensor). *A simple tensor T^s is a tensor that can be expressed with the outer products of p vectors v where p is the dimension of a tensor, i.e. $[T^s]_{i, .^p_{\cdot}, k} = [v_1]_i \ldots [v_p]_k$. T is of rank t if and only if t is the smallest number of simple tensors required to express T [23, p.309], i.e. $T = T_1^s + \ldots + T_t^s$.*

The joint action probability distribution of the Nash equilibrium, $\boldsymbol{z}_{\boldsymbol{a}}^k$, has a rank of 1 as the Nash equilibrium is expressed with an outer product of the player's strategies x_i as shown in (11), and is thus a simple tensor. Since all Nash equilibria are elements of the set of correlated equilibria $\mathcal{G}$ [21], an element in $\mathcal{G}$ with equal or higher rank than any elements in $\mathcal{H}$ always exists. Based on this characteristic, elements within the set $\mathcal{H}$ are termed *reduced rank correlated equilibria (RRCE)*.

B. Multiplayer coordination and RRCE algorithm

We consider an optimization problem that seeks a correlated equilibrium $\boldsymbol{z}$, which minimizes a particular cost function J that considers the social context. By recommending actions sampled from the correlated equilibrium to each player, a coordinator can help the players reach a joint action that minimizes J. Such a socially optimal joint action is often unachievable without a coordination mechanism.

The cost function J that evaluates the quality of a correlated equilibrium—such as efficiency and fairness—is denoted as

$$J : \mathbb{R}_{\geq 0}^{m \times .^n_{\cdot} \times m} \to \mathbb{R}. \qquad (13)$$

In particular, we search for an *optimal* correlated equilibrium by solving the following optimization:

$$\begin{aligned} &\underset{\boldsymbol{z}}{\text{minimize}} && J(\boldsymbol{z}) \\ &\text{subject to} && \text{condition from (9)}. \end{aligned} \qquad (14)$$

We denote this method as the *Correlated Equilibrium (CE) algorithm*. As stated previously, solving this problem requires high computation costs.

In contrast, we propose an algorithm that computes RRCE, termed the *RRCE algorithm*. The RRCE algorithm consists of two phases. First, the algorithm searches for multiple Nash equilibria—each satisfies Definition 1—and computes the corresponding joint action distributions according to (11). The computed joint action distribution is denoted $\{\boldsymbol{z}^1, \ldots, \boldsymbol{z}^d\}$. We suggest two methods for searching Nash equilibria: (i) *random initialization* method, which randomly initializes the numerical solver [24] when solving (6), and (ii) *brute-force* method, which exhaustively enumerates all deterministic joint actions and identifies all pure Nash equilibria by brute force [25].

Second, the RRCE algorithm computes a reduced rank correlated equilibrium given by $\sum_{k=1}^{d}[\gamma^\star]_i z^k$, where $\gamma^\star \in \mathbb{R}_{\geq 0}^d$ is the optimal solution of the following optimization problem:

$$\begin{aligned}
\underset{\gamma}{\text{minimize}} \quad & J(\textstyle\sum_{i=1}^{d}[\gamma]_i z^i) \\
\text{subject to} \quad & \gamma \geq \mathbf{0}_d, \quad \mathbf{1}_d^\top \gamma = 1.
\end{aligned} \tag{15}$$

V. Coordination in Air Traffic Management

A noncooperative multiplayer coordination problem is frequently observed in air traffic management. Consider a situation where two aircraft are trying to land at an airport with a single runway. In this case, the players are aircraft, and the limited resource is runway occupancy. Each aircraft prefers to occupy the runway (*i.e.* land) immediately rather than wait until the other aircraft lands and vacates the runway. However, neither can occupy the runway simultaneously without causing a crash and incurring a heavy penalty.

This two-aircraft landing scenario can be expressed in a normal form game:

	Occupy	Yield
Occupy	δ,δ	$0,\rho$
Yield	$\rho,0$	ρ,ρ

where $\delta \in \mathbb{R}_+$ is a penalty that is arbitrarily larger than any other values in this matrix and $\rho \in \mathbb{R}_+$ is a small penalty that occurs when a player chooses the Yield action. The cost matrix for this game is as follows:

$$C^{ij} = C^{ji} = \begin{bmatrix} \delta & 0 \\ \rho & \rho \end{bmatrix} \tag{16}$$

The best outcome, 0, occurs when each player occupies while the opponent yields. The worst case occurs when both players Occupy, thus resulting in δ penalty for each player.

This game setting can be expanded to various air traffic management scenarios. Players could be airlines competing over airport slots, departure and arrival queues competing over runway usage, and airports competing over using a particular air route.

A. Airport departure/arrival queue game

We will focus on the scenario of the departure and arrival queues competing over the limited number of runways, as illustrated in Figure 1.

1) Players: Let *queues* indicate a queue of aircraft waiting for departure or arrival. A queue that holds aircraft waiting for departure is *departure queue*, and one with aircraft waiting for arrival is an *arrival queue*. An aircraft is added to the i-th queue according to a Poisson process at a specific rate v_i for each queue, $\text{Pois}(v_i)$. Each departure or arrival queue is a player in this game, and each wants to reduce their queue length by deploying the aircraft in their queue to occupy runways.

2) Actions: The game is played every 5 minutes. Every time the game is played, the players have two actions for each runway: Occupy the runway or yield its usage to the others. Each player can decide whether to occupy or yield each runway independently if multiple runways exist. Let r be the number of runways; then the number of actions, m, becomes

$m = 2^r$. For example, if there are two runways, then there are four possible joint actions: $\mathcal{A} = \{(\text{Occupy}, \text{Occupy}), (\text{Occupy}, \text{Yield}), (\text{Yield}, \text{Occupy}), (\text{Yield}, \text{Yield})\}$. In practice, the control tower (coordinator) will choose among these joint actions based on z and then radio a recommended action to each aircraft in the front of each queue.

3) Cost matrix: The cost for each player (queue) in this game represents the delay applied to all the aircraft within the queue. Since the game is played every 5 minutes, if a player chooses to Yield, it causes 5-minute delays to all the scheduled aircraft in its queue. We designate 5 as ρ, the Yield penalty. As the average number of added aircraft to each queue per unit time i is proportional to v_i, the cost for Yield becomes $5v_i$, or ρv_i minutes per runway. If a particular player chooses Occupy and all other players do Yield the runway, that player does not incur any costs. However, if the simultaneous occupation of a runway occurs, all the occupying players receive a large penalty $\delta \gg \rho$.

An example cost matrix for queue i when there are 2 runways, or $m = 4$ is,

$$C^{ij} = v_i \begin{bmatrix} 2\delta & \delta & \delta & 0 \\ \delta+\rho & \delta+\rho & \rho & \rho \\ \delta+\rho & \rho & \delta+\rho & \rho \\ 2\rho & 2\rho & 2\rho & 2\rho \end{bmatrix}, \forall j \in -i \tag{17}$$

where the order of actions is (Occupy, Occupy), (Occupy, Yield), (Yield, Occupy), (Yield, Yield).

VI. Numerical Experiments

We evaluate the effectiveness of the RRCE algorithm by comparing it with the algorithm based on the correlated equilibrium and Nash equilibrium, respectively [24]. The algorithm based on the correlated equilibrium, termed the *CE algorithm*, solves the optimization in (14). The algorithm based on the Nash algorithm termed the *Nash algorithm*, finds a solution that satisfies the conditions in (6). It converges to a different equilibrium under different initialization when multiple Nash equilibria exist.

The proposed RRCE algorithm has two variations depending on which strategy it uses to find multiple Nash equilibria; RRCE using the *random initialization* method is denoted as the *Random-RRCE*. It repeats *random initialization* method for a fixed number of times, and the newly discovered Nash equilibrium is returned. The RRCE using the *brute-force* method is denoted as the *Brute-RRCE*. It performs an exhaustive search for *pure* Nash equilibrium (with a deterministic strategy) by enumerating the best responses of each player for all joint actions.

In summary, four algorithms are used for the experiment (two RRCE algorithms, the CE algorithm, and the Nash algorithm). Note that other baseline algorithms are neglected since there have not been any algorithms that share common objectives with the proposed algorithm, making the direct comparison unnecessary, per the discussion in Section II.

To capture the variance in computation time and solution quality of each algorithm, we conduct a series of Monte Carlo experiments. Fifty trials with fixed arrival rates v_i are tested for each distinct number of players (n) and actions (m). A total of 18 settings for (n, m) are investigated, with

n varying from 2 to 7 and the number of runways r varying from 1 to 3, which is equivalent to the number of actions, $m \in \{2, 4, 8\}$ respectively. These test cases correspond to the number of joint actions ranging from 2^2 to 2^{21}. We implement algorithms in the Julia programming language, using the ParametricMCPs.jl package [26] and the PATH mixed complementarity program solver [24]. All experiments were performed on an AMD Ryzen 9 7950X processor.

A. Evaluation criteria

We use three metrics to measure the performance of each algorithm: the computation time, the Gini index, and the average delay cost per player.

1) Computation time (CT): The computation time measures the time required to compute the solution. We evaluate the computation time in two ways: (i) *Solver time*, which is the runtime for the solver to find the solution, and (ii) *Total computation time*, which is a combination of the solver time and the *pre-processing time* required to compile the problem before running the solver.

2) Average delay cost (AC): The average delay cost measures the average cost of the players as

$$\mathrm{AC}(\mathbf{c}) = \tfrac{1}{n} \sum_{i \in \mathbb{N}_{[1,n]}} c_i, \qquad (18)$$

where $\mathbf{c} = \{c_1, c_2, \ldots, c_n\}$.

3) Gini index (GI): The Gini index is an indicator measuring fairness among the players. The smaller the discrepancy in the cost between the players, the better the fairness. A small Gini index indicates the players achieve a more fair outcome. We compute the Gini index as

$$\mathrm{GI}(\mathbf{c}) = \tfrac{1}{2\mathrm{AC}(\mathbf{c})n^2} \sum_{i,j \in \mathbb{N}_{[1,n]}} |c_i - c_j|, \qquad (19)$$

where $\mathrm{AC}(\mathbf{c})$ is an average delay cost computed from [27].

B. Objective function

There are two objectives that the coordinator in this scenario has to satisfy: (i) Minimize the total cost and (ii) Minimize the cost differences between the players. Qualitatively speaking, the first objective promotes the overall system's efficiency, and the second objective promotes fairness among the players.

We adopted a cost function that balances both objectives: the *fairness-threshold-criteria with maximin fairness* [27]. It is formulated as,

$$J(\mathbf{z}) = -n\Delta + \sum_{i=1}^{n} \max(c_i + \Delta, \mathbf{c}_{\max}), \qquad (20)$$

where Δ is a pre-specified fairness-threshold value and $\mathbf{c}_{\max}$ is the maximum value of $\mathbf{c}$. Note that each c_i can be computed from $\mathbf{z}$ directly via (4). If the difference between player i's cost and $\mathbf{c}_{\max}$ is less than Δ, the cost function adds c_i into consideration. However, if the difference is larger than Δ, or c_i is less than $\mathbf{c}_{\max}$ by more than Δ, the cost function focuses on $\mathbf{c}_{\max}$ rather than c_i.

When the differences of costs between all the players are within Δ, the cost function becomes $J(\mathbf{c}) = \sum_{i=1}^{n} c_i$. If a single player incurs a higher cost than any other player by more than Δ, the overall cost depends only upon that player's cost, and $J(\mathbf{c}) = -n\Delta + n\mathbf{c}_{\max}$.
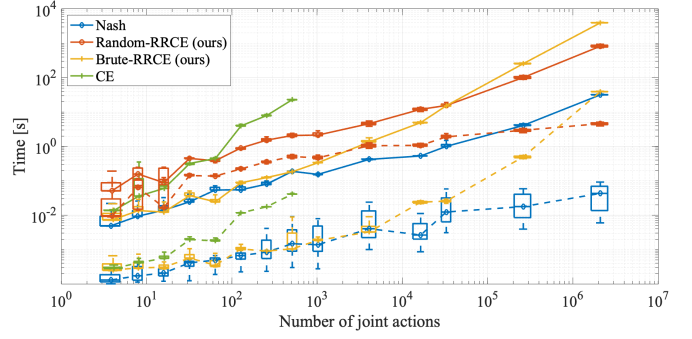

Fig. 2: Computation time plot in log-log scale. Solver runtime plot (dotted) and total computation time (solid) that includes both solver runtime and preprocessing time.
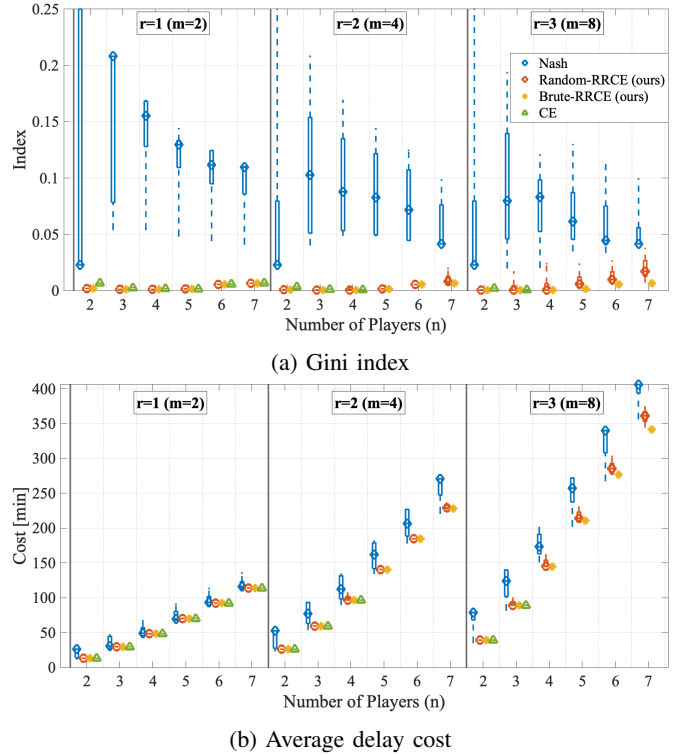

(a) Gini index

(b) Average delay cost

Fig. 3: Unfairness indicator (Gini index) plot (upper) and average delay cost plot (bottom). Results are shown per runway number r and the number of players n.

C. Result

The RRCE and Nash algorithm solves all the test cases involving up to 2^{21} joint actions. However, the CE algorithm fails to solve several test cases due to a lack of available memory, thus leaving missing data points in the figure. The maximum number of joint actions the CE algorithm can handle is 2^9.

1) Computation time: Figure 2 shows the computation time regarding the number of joint actions in each test case. Random-RRCE shows the largest value in terms of solver time for most of the test cases. However, it shows a polynomial increasing trend, which is also true for the Nash algorithm. The Brute-RRCE and CE algorithms show a much

greater rate of increase in solver time, and the CE's rate of increase is the greatest (*i.e.* worst) among the algorithms.

A similar trend exists in the total computation time. The Random-RRCE and Nash show polynomial increasing trends, while Brute-RRCE and CE show a greater rate. The total computation time for CE exceeds that of the Random-RRCE in the test case involving 64 joint actions, becoming the most time-consuming algorithm. The maximum total computation time reduction observed from Random-RRCE to CE was 91.0% in a test case involving 2^9 joint actions.

2) Gini index: Figure 3 shows the solution quality indicators (Gini index, average delay cost) with respect to the number of players (n) and actions (m). Regarding fairness, Figure 3a shows that the RRCE and CE algorithms kept the costs between the players similar. Among the algorithms apart from the Nash algorithm, the Random-RRCE shows the highest (worst) median Gini index value and higher variance, and the trend becomes notable in the cases involving larger n and m. This degradation is due to the reduced ratio of the number of sampled Nash equilibria to the number of all Nash equilibria in the game as the problem size increases.

3) Average delay cost: According to Figure 3b, the median of average delay costs for CE and RRCE algorithms are consistently lower than that of the Nash equilibrium. RRCE and CE yield a solution showing similar performance to the cases CE solved. Random-RRCE shows performance between these two extremes, with the average delay cost reduced by 1.8% up to 50.4% than that of the Nash approach, while showing a maximum optimality gap of 0.066% compared to the observable CE algorithm results.

VII. Conclusion and Future Works

We propose a highly scalable algorithm that computes the correlated equilibrium, a coordination mechanism that has been proven helpful in multiplayer noncooperative games, by approximating the set of correlated equilibria with the convex hull of multiple Nash equilibria. Numerical experiments show that the proposed algorithm demonstrates improved scalability compared to the standard correlated equilibrium computation and superior solution quality (fairness and average cost per player) to the Nash solution, which does not involve coordination.

Despite this promising result, there is room for further development. As this paper focuses on the correlated equilibrium set approximation method, we adopted a simple strategy to search for multiple Nash equilibria. However, as the problem scale increases, the Nash equilibria recovered represent a decreasing fraction of the total number of Nash points, which makes the convex hull approximation worse. Thus, exploring a method that can search for Nash equilibrium to maximize the volume of the convex hull effectively will be an exciting topic for future study.

References

[1] C. Papadimitrioiu, "Computing correlated equilibria in multi-player games," *Proc. of the Annu. ACM Symp. on Theory of Comput.*, pp. 49–56, 01 2005.

[2] R. J. Aumann, "Subjectivity and correlation in randomized strategies," *J. of Math. Econ.*, vol. 1, no. 1, pp. 67–96, 1974.

[3] J. Bone, M. Droiuvelis, and I. Ray, "Coordination in 2 x 2 games by following recommendations from correlated equilibria," 2013.

[4] J. Duffy, E. K. Lai, and W. Lim, "Coordination via correlation: an experimental study," *Econ. Theory*, vol. 64, pp. 265–304, Aug 2017.

[5] A. Cavaliere, "Coordination and the provision of discrete public goods by correlated equilibria," *J. of Public Economic Theory*, vol. 3, no. 3, pp. 235–255, 2001.

[6] D. Wu, Y. Cai, L. Zhou, Z. Zheng, and B. Zheng, "Cooperative strategies for energy-aware ad hoc networks: A correlated-equilibrium game-theoretical approach," *IEEE Trans. on Veh. Technol.*, vol. 62, no. 5, pp. 2303–2314, 2013.

[7] Y. Babichenko, S. Barman, and R. Peretz, "Simple approximate equilibria in large games," in *Proc. of the Fifteenth ACM Conf. on Econ. and Comput.*, ser. EC '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 753–770.

[8] S. Hart and A. Mas-Colell, "A simple adaptive procedure leading to correlated equilibrium," *Econometrica*, vol. 68, no. 5, pp. 1127–1150, 2024/02/19/ 2000, full publication date: Sep., 2000.

[9] J. Lenzo and T. Sarver, "Correlated equilibrium in evolutionary models with subpopulations," *Games and Econ. Behav.*, vol. 56, no. 2, pp. 271–284, 2006.

[10] J. Arifovic, J. F. Boitnott, and J. Duffy, "Learning correlated equilibria: An evolutionary approach," *Journal of Econ. Behav. & Org.*, vol. 157, pp. 171–190, 2019.

[11] J. R. Marden, "Selecting efficient correlated equilibria through distributed learning," *Games and Econ. Behav.*, vol. 106, pp. 114–133, 2017.

[12] H. P. Borowski, J. R. Marden, and J. S. Shamma, "Learning efficient correlated equilibria," in *53rd IEEE Conf. on Decis. and Control*, 2014, pp. 6836–6841.

[13] S. G. Park and P. K. Menon, "Game-theoretic trajectory-negotiation mechanism for merging air traffic management," *J. of Guid., Control, and Dyn.*, vol. 40, no. 12, pp. 3061–3074, 2017.

[14] D. Pisarski and C. Canudas-de Wit, "Nash game-based distributed control design for balancing traffic density over freeway networks," *IEEE Trans. on Control of Netw. Syst.*, vol. 3, no. 2, pp. 149–161, 2016.

[15] L. Yu, E. Masabo, and C. Mutimukwe, "Nash equilibrium: Better strategy for agents coordination," in *2008 IEEE Asia-Pacific Services Comput. Conf.*, 2008, pp. 795–800.

[16] D. A. Braun, P. A. Ortega, and D. M. Wolpert, "Nash equilibria in multi-agent motor interactions," *PLoS Comput. biol.*, vol. 5, no. 8, p. e1000468, 2009.

[17] S. Sanjari, N. Saldi, and S. Yüksel, "Nash equilibria for exchangeable team against team games and their mean field limit," in *2023 Amer. Cont. Conf. (ACC)*. IEEE, 2023, pp. 1104–1109.

[18] H. Kamisetty, E. P. Xing, and C. J. Langmead, "Approximating correlated equilibria using relaxations on the marginal polytope." in *ICML*, 2011, pp. 1153–1160.

[19] P. Harker and J.-S. Pang, "Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications." *Math. Program.*, pp. 18–25, 03 1990.

[20] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[21] R. Nau, S. G. Canovas, and P. Hansen, "On the geometry of nash equilibria and correlated equilibria," *Int. J. of Game Theory*, vol. 32, no. 4, pp. 443–453, Aug 2004.

[22] S. R. Phade and V. Anantharam, "On the geometry of nash and correlated equilibria with cumulative prospect theoretic preferences," *Decision Analysis*, vol. 16, no. 2, pp. 142–156, 2019.

[23] N. Bourbaki, *Algebra: Chapter 2*. Springer Nature, 2023.

[24] S. P. Dirkse and M. C. Ferris, "The path solver: a nommonotone stabilization scheme for mixed complementarity problems," *Optim. Methods and Softw.*, vol. 5, no. 2, pp. 123–156, 1995.

[25] H. Li, W. Huang, Z. Duan, D. H. Mguni, K. Shao, J. Wang, and X. Deng, "A survey on algorithms for nash equilibria in finite normal-form games," *Comput. Sci. Rev.*, vol. 51, p. 100613, 2024.

[26] L. Peters, "ParametricMCPs.jl," 2022. [Online]. Available: https://github.com/lassepe/ParametricMCPs.jl

[27] V. Xinying Chen and J. N. Hooker, "A guide to formulating fairness in an optimization model," *Ann. of Oper Res.*, vol. 326, no. 1, pp. 581–619, Jul 2023.