# On the Optimal Mixing Problem of Approximate Nash Equilibria in Bimatrix Games

Xiaotie Deng[a], Dongchen Li[b], Hanyu Li[a]

[a] *CFCS, School of Computer Science, Peking University, No.5 Yiheyuan Road, Haidian District, 100871, Beijing, China*
[b] *School of Computer Science, The University of Hong Kong, Pokfulam Road, Central and Western District, 999077, Hong Kong, China*

**Abstract**

This paper introduces the optimal mixing problem, a natural extension of the computation of approximate Nash Equilibria (NE) in bimatrix games. The problem focuses on determining the optimal convex combination of given strategies that minimizes the approximation (i.e., regret) in NE computation. We develop algorithms for the exact and approximate optimal mixing problems and present new complexity results that bridge both practical and theoretical aspects of NE computation. Practically, our algorithms can be used to enhance and integrate arbitrary existing constant-approximate NE algorithms, offering a powerful tool for the design of approximate NE algorithms. Theoretically, these algorithms allow us to explore the implications of support restrictions on approximate NE and derive the upper-bound separations between approximate NE and exact NE. Consequently, this work contributes to theoretical understandings of the computational complexity of approximate NE under various constraints and practical improvements

*Email addresses:* xiaotie@pku.edu.cn (Xiaotie Deng), dongchen.li@connect.hku.hk (Dongchen Li), lhydave@pku.edu.cn (Hanyu Li)

in multi-agent reinforcement learning (MARL) and other fields where NE computation is involved.

*Keywords:* Approximate Nash equilibria, Bimatrix games, Optimal mixing, Approximation algorithms, Computational complexity

---

## 1. Introduction

The problem of approximate *Nash equilibrium* (NE) computation is interesting and fundamental from both theoretical and pragmatic perspectives. Theoretically, approximate NE builds bridges between several important complexity classes related to TFNP [1], especially PPAD [2, 3]. Practically, an approximate NE solver is a core component in multi-agent reinforcement learning (MARL) [4, 5, 6, 7], which has been successfully applied to train machine agents that can defeat the top human players in electronic games [8, 9].

To define the approximate NE problem, consider a two-player bimatrix game with payoff matrices $R \in [0,1]^{m\times n}$ and $C \in [0,1]^{m\times n}$. For any strategies $x, y$ of both players, respectively, we define

$$f(x,y) = \max\{f_R(x,y), f_C(x,y)\} \geq 0$$

with $f_R(x,y) = \max\{Ry\} - x^{\mathsf{T}}Ry$ and $f_C(x,y) = \max\{C^Tx\} - x^{\mathrm{T}}Cy$. Intuitively, in the game given by payoff matrices $R, C$ where the two players select $x, y$ respectively, $f(x,y)$ is a measure of their willingness to unilaterally deviate from the current strategy. Following [10, 11], the goal of NE computation in bimatrix games can be written as:

$$\operatorname*{argmin}_{x,y} f(x,y) \quad \text{s.t.} \quad x \in \Delta_m, y \in \Delta_n. \tag{1}$$

In the literature [12], $f(x, y)$ is called the *approximation* of $(x, y)$. A strategy profile $(x, y)$ is an NE if $f(x, y) = 0$ and an $\epsilon$-NE if $f(x, y) \leq \epsilon$. Since $f \geq 0$ and NE always exists [13], the solution of (1) must be an NE.

### *1.1. The Optimal Mixing Problem*

We propose the *optimal mixing problem* of approximate NE, which is a natural extension of the approximate NE computation. Given $s, t$ strategies $x_1, \ldots, x_s$ and $y_1, \ldots, y_t$ of the two players, the $(s, t)$-optimal mixing problem is:

$$\operatorname*{argmin}_{\alpha,\beta} f(\alpha_1 x_1 + \cdots + \alpha_s x_s, \beta_1 y_1 + \cdots + \beta_t y_t) \quad \text{s.t.} \quad \alpha \in \Delta_s, \beta \in \Delta_t. \quad (2)$$

Intuitively, the problem seeks the convex combination of $x_1, \ldots, x_s$ and $y_1, \ldots, y_t$ that has the minimum approximation. We also define the *$\epsilon$-optimal $(s, t)$-mixing problem* by allowing an additive tolerance of $\epsilon$ in the objective, i.e., the output is required to have approximation no more than $f^* + \epsilon$, where $f^*$ is the optimal value of (2).

The approximate and exact optimal mixing problem is a nature extension of the NE computation: when the input of the approximate and exact optimal mixing problem is $e_1, \ldots, e_m$ and $e_1, \ldots, e_n$, the standard basis of $\mathbb{R}^m$ and $\mathbb{R}^n$, it is exactly the NE computation problem (1).

The motivation of the optimal mixing problem is twofold.

- **Algorithm design and analysis.**

  To guarantee a certain approximation bound, current polynomial-time algorithms for approximate NE all follow a *search-and-mix* method [12]. It can be divided into two polynomial-time phases. In the search phase,

an algorithm computes a fixed number of strategies of each player. In the mixing phase, the algorithm then make *specific* convex combinations of the selected strategies and outputs the one with the minimum $f$ value. However, such a design paradigm has several limitations:

  - **Different approaches seldom integrate.** The search phases of these algorithms follow very different and even incomparable approaches, e.g., gradient descent [10, 11], linear programming [14], and zero-sum game [15, 16]. However, it is worth considering whether an algorithm with better approximation bounds can be designed by combining these different approaches. The optimal mixing problem, which allows to mix any strategies, offers a unified framework for such a combination.
  - **Overemphasis on worst cases.** To guarantee an approximation bound, all mixing phase design in the literature only focuses on the worst-case instances [12], which are rare compared to other instances [17, 18, 12, 19]. However, such focuses may hinder the practical usefulness of these algorithms. It is natural to directly find the *optimal* convex combination for every instance, which is essentially an optimal mixing problem.

- **Computational complexity of approximate NE under support restrictions.** It has been shown in [20, 21] that adding certain natural requirements increases the complexity of computing a Nash Equilibrium (NE) from PPAD-complete [2] to NP-complete. However, this conclusion does not necessarily hold when considering approximate

Nash Equilibria, which is a more computationally appropriate notion that accounts for bounded rationality.[1] For example, deciding the existence of an exact NE with a specific support is NP-complete [20]. In contrast, by slightly adjusting the probabilities in the strategy profile, we can demonstrate that for any support, there always exists an $\epsilon$-NE on it.

From above observations, we know that the complexity of approximate NE computation could be very different from that of exact NE computation if we put certain restrictions on the solution. The optimal mixing problem provides a unified framework to reexamine the effect of such restriction over approximate NE, especially the support restriction.

### *1.2. Our contributions*

Bearing these motivations, we develop algorithms for the approximate and exact optimal mixing problems. Informally, we have that:

**Theorem 1.1** (Main results)**.** *When $s \leq 2, t \leq 3$ or $s = 1, t = \text{poly}(n)$, there exists an algorithm solving any optimal mixing problem in* $\text{poly}(m, n)$ *time. Moreover, there exists an algorithm solving any $\epsilon$-optimal $(s, t)$-mixing problem in time* $\text{poly}(m, n, s, t) \left(e + \frac{e}{\sqrt{\epsilon/2}}\right)^{s+t}$ *and space* $\text{poly}(m, n, s, t)$.

This theorem provides various implications, as is described below.

- **Algorithm design and analysis**:

[1] According to [22], approximate NE is a more realistic solution concept as it involves bounded rationality.

- **Integration of different approaches**: Our algorithms can be used to combine *arbitrary* strategies computed by different approaches. Thus, the combined strategy fuses advantages from these different approaches.

- **Instance-optimal mixing phases**: With these algorithms, we propose general approximate and exact polynomial-time algorithms for instance-optimal mixing phases. There is no need to design the mixing phase ad hoc in the future. Interestingly, [23] shows that there is an automatic way to derive the approximation bound for any search-and-mix algorithm, even when using our algorithms as the mixing phase. Thus, together with our algorithms, the mixing phase design is fully automated.[2]

- **Computational complexity of approximate NE under support restrictions**:

  - **Restrictions enlarge the complexity**: As is shown in Table 1, our algorithms establish upper bounds for finding the best approximate NE over certain support. Table 1 shows the upper bound complexity of approximate NE with support restrictions (the left column, where the support of their approximate NE must be over certain pure strategies) is significantly larger than that of approximate NE without support restrictions (the right column). This

[2] A perhaps surprising result given by [23] is that for each algorithm in the literature, the approximation bound with our algorithms as the mixing phase is the same as original ad hoc ones!

is consistent with the cases in exact NE computation [20, 21].

– **Approximate NE with restrictions could be more difficult than exact NE**: The upper bound results reveal a counter-intuitive separation between the complexities of approximate and exact Nash Equilibria (NE). Specifically, the complexity of deciding the existence of a $1/n^{O(1)}$-approximate NE under support restrictions is $2^{O(n \log n)}$. Surprisingly, this is even higher than the complexity for finding an exact NE under the same support restrictions, which is $2^{O(n)}$ by support enumeration [24]. While it is commonly believed that approximate NE are easier to compute than exact NE, this intuition does not always hold when there is no guarantee of existence due to support restrictions. In such cases, demonstrating the non-existence of a $1/n^{O(1)}$-approximate NE can be more challenging than showing the non-existence of an exact NE.

**Remark 1.1.** *One may note that in Theorem 1.1, we stop at* $(2, 3)$ *for exact optimal mixing problem. We discuss the reason in Section 6.*

### *1.3. Related Work*

**Complexity and Approximation of NE.**

The computational complexity of approximate NE has been extensively studied. Initially, Papadimitriou [27] introduces a general complexity class PPAD and shows that computing $1/2^n$-NE lies in PPAD. Later, computing $1/\operatorname{poly}(n)$-NE is shown to be PPAD-complete for $k$-player games with any

| $\epsilon$ \ $s,t$ | $\Theta(n) \leq n$ | $n$ |
|---|---|---|
| $> 1/3$ | ?; $\mathbf{2^{O(n)}}$ | poly$(n)$ [10] |
| $\in [\epsilon^*, 1/3]$ | ?; $\mathbf{2^{O(n)}}$ | ?; $n^{O(\log n/\epsilon^2)}$ [22] |
| $< \epsilon^*$, const | quasi-poly$(n)^2$ [25]; $\mathbf{2^{O(n)}}$ | quasi-poly$(n)^2$ [25]; $n^{O(\log n/\epsilon^2)}$ [22] |
| $= 1/n^{O(1)}$ | PPAD-hard [2]; $\mathbf{2^{O(n \log n)}}$ | PPAD-complete [2]; $2^{O(n)}$ [26] |
| $= 1/2^{O(n)}$ | PPAD-hard [2]; $\mathbf{2^{O(n^2)}}$ | PPAD-complete [2]; $2^{O(n)}$ [26] |
| $= 0$ (exact) | FNP-hard [20, 21]; ? | PPAD-complete [2]; $2^{O(n)}$ [26] |

Table 1: This table shows the results of $\epsilon$-optimal $(s,t)$-mixing problem with distinct pure-strategy input. Without loss of generality, we assume $m = n$ and $s \leq t$. The results are presented in "lower bound; upper bound" pairs if the two bounds are not matched. Our new results are in boldface.

fixed $k \geq 2$ [2, 3]. Moreover, computing NE in two-player games is hard even in the smoothed meaning [2], or restricting the rank of game to constant [28]. These results establish the hardness of approximate NE computing with polynomial-small approximation. It is well-believed that computing NE could require exponential time (ETH for PPAD). See, e.g. [29, 25].

For constant approximation, it seems to be easier than polynomial-small approximation. For any given $\epsilon > 0$, there is an algorithm finding an $\epsilon$-NE [22] in $n^{O(\log n/\epsilon^2)}$ time (QPTAS). Assuming ETH for PPAD, Rubinstein [25] shows that there exists a constant $\epsilon^* > 0$ such that computing an $\epsilon^*$-NE in a two-player $n \times n$ game requires $n^{\log^{1-o(1)} n}$ time. This matches the QPTAS result [22] up to $o(1)$ term.

The lower bound results on constant approximation lead to the study of the upper bound, i.e., seeking the minimum $\epsilon$ such that there exists a

polynomial-time algorithm computing an $\epsilon$-NE. Most literature focuses on two-player (bimatrix) games in the literature. A series of polynomial-time algorithm
[15, 2, 14, 30, 10, 31, 11] have been proposed, with the approximation from the beginning of 3/4 [31] to the state-of-the-art $1/3 + \delta$ [10]. For a more thorough introduction, see [12].

Notably, all the results above, including algorithms and hardness results, heavily rely on the existence of NE. In fact, if we want to find NE with certain natural requirements, such as strong NE, NE over a certain support, or with certain social welfare, such NE may not exist. Moreover, by using SAT to make reductions, [20, 21] show that deciding the existence of such NE is NP-compete. The FNP-hardness result in Table 1 is a direct corollary of this reduction. However, to the best of our knowledge, there is no literature at all for similar discussions over approximate NE.

**NE Computation in Practical Applications.**

Emerging from game theory, NE computation has been widely applied in many fields, including Internet economics, computer science, and machine learning. Most prominently, NE computing is a core component in many multi-agent reinforcement learning (MARL) algorithms, including PSRO [7], Nash-Q [4], Nash-VI [6], and Nash-V learning [5]. MARL has been successfully applied to train machine agents that can defeat the top human players in electronic games, including AlphaStar [9] in StarCraft II and OpenAI Five [8] in Dota 2. With such fruitful applications, it is demanding to design efficient algorithms for NE.

*1.4. Paper Organization*

This paper is organized as follows. In Section 2, we introduce the basic concepts and notations. In Section 3, we present the polynomial-time algorithms for the optimal mixing problem. In Section 4, we present an algorithm for the approximate optimal mixing problem. In Section 5, we show how to apply the optimal mixing problem to make an instance-optimal enhancement to the search-and-mix method in the literature. In Section 6, we conclude the paper, discuss the limitations of our work, and propose future directions.

## 2. Preliminaries

**Asymptotic Notations.**

We use the standard asymptotic notations $O(\cdot)$ and $\Theta(\cdot)$ to describe the asymptotic behavior of functions. For two positive functions $f$ and $g$, $f = O(g)$ means that there exists a constant $c > 0$ such that $f(n) \leq c \cdot g(n)$ for all sufficiently large $n$. $f = \Theta(g)$ means that $f = O(g)$ and $g = O(f)$.

**Vectors and Matrices.**

Denote the $n$-dimensional Euclidean space by $\mathbb{R}^n$. The standard orthonormal basis of $\mathbb{R}^n$ is $e_1, \ldots, e_n$. Notation $[n] := \{1, \ldots, n\}$ represents an index set. For vector $v \in \mathbb{R}^n$, denote its $i$th item by $v_i$. For vector $u \in \mathbb{R}^n$, define the following operators: $\max\{u\} := \max\{u_1, \ldots, u_n\}$, $\min\{u\} := \min\{u_1, \ldots, u_n\}$. For two vectors $v, w \in \mathbb{R}^n$, notation $v \geq w$ represents that $v_i \geq w_i$ holds for every $i \in [n]$.

For an $m \times n$ matrix $A$, denote its $i$th row by $A_i$, its $j$th column by $A^j$, and its item at $i$th row $j$th column by $A_{ij}$. Its transpose is denoted by $A^\mathsf{T}$.

**Simplex and Convex Combinations.**

A standard $(n-1)$-*simplex* is the set $\Delta_n := \{\alpha \in \mathbb{R}^n : \alpha \geq 0 \text{ and } \sum_{i=1}^n \alpha_i = 1\}$. A simplex can be viewed as a probability space and its elements are probability vectors. For given elements $z_1, \ldots, z_w$ from $\mathbb{R}^n$, the set of their *convex combinations* is defined to be $\{\alpha_1 z_1 + \cdots + \alpha_w z_w : \alpha \in \Delta_w\}$, where any vector $\alpha \in \Delta_w$ determines a convex combination $\alpha_1 z_1 + \cdots + \alpha_w z_w$.

**Games, Mixed Strategies, and Best Responses.**

We only focus on *bimatrix games*, in which there are two players. We refer to them as the row player and the column player. A game can be defined by a pair of payoff matrices $R$ and $C$ in $[0,1]^{m \times n}$. When the row player chooses the $i$th row and the column player chooses the $j$th column, their payoffs are denoted by $R_{ij}$ and $C_{ij}$, respectively.

For each player, a *(mixed) strategy* of the row (column) player is a vector $x \in \Delta_m$ ($y \in \Delta_n$). In particular, pure strategies are a specific pure strategy is chosen with a probability of 1. A *strategy profile* $(x, y)$ refers to a pair of mixed strategies $x$ and $y$ from the row and column players, respectively. Given the strategy profile, the payoffs of the row player and the column player are $x^\mathsf{T} R y$ and $x^\mathsf{T} C y$, respectively. A *best response* against a strategy $x$ ($y$) from the row (column) player is a mixed strategy of the column (row) player that maximizes the expected payoff against $x$ ($y$).

**Approximate Nash Equilibria from the Optimization Viewpoint.**

We follow [11] to define $\epsilon$-NE. First, define the regret of the row player and the column player as follows:

$$f_R(x, y) := \max\{Ry\} - x^\mathsf{T} R y \text{ and } f_C(x, y) := \max\{C^\mathsf{T} x\} - x^\mathsf{T} C y.$$

Define $f(x,y) := \max\{f_R(x,y), f_C(x,y)\}$. Then a strategy profile $(x,y)$ is an $\epsilon$-NE if and only if $f(x,y) \leq \epsilon$. $f(x,y)$ is called the *approximation* of $(x,y)$. Particularly, a strategy profile is an NE if it is a 0-NE. The minimum of $f$ over $\Delta_m \times \Delta_n$ is always 0 by the existence of NE [13].

## 3. Polynomial-Time Algorithms for Optimal Mixing Problems

In this section, we propose polynomial-time algorithms for the optimal mixing problem. In Section 3.1, we summarize the results. In Section 3.2, we sketch the main ideas of the algorithms. The detailed algorithms and proofs are given in Section 3.3.

Recall that the optimal mixing problem is defined as follows.

**Definition 3.1** (Optimal $(s,t)$-mixing problems)**.** *An* optimal $(s,t)$-mixing problem *has the following input and output.*

- Input*: Bimatrix game $(R,C)$, mixed strategies $x_1,\ldots,x_s$ of the row player and $y_1,\ldots,y_t$ of the column player.*
- Output*: Coefficients $\alpha^* \in \Delta_s$, $\beta^* \in \Delta_t$ that minimize*

$$f(\alpha_1 x_1 + \cdots + \alpha_s x_s, \beta_1 y_1 + \cdots + \beta_t y_t).$$

For convenience, we name an algorithm as an *optimal $(s,t)$-mixing algorithm* if it solves any optimal $(s,t)$-mixing problem.

### *3.1. Summary of Results*

A summary of the results in this section is presented in Theorem 3.1.

**Theorem 3.1.** *The following statements hold.*

1. *For any $t$, there exists an optimal $(1,t)$-mixing algorithm in $O(mnt + L(t,m))$ time, where $L(t,m)$ is the time complexity of solving a linear program with $t$ variables and $m$ constraints.*

2. *There exists an optimal $(2,2)$-mixing algorithm in $O(mn)$ time.*

3. *There exists an optimal $(2,3)$-mixing algorithm in $O(m^2(n+\log m) + n\log n)$ time.*

*3.2. Sketch of the Ideas.*

We now sketch the main ideas of the algorithms. We begin by scrutinizing the form of the problem. The objective function in Definition 3.1 can be expanded as follows:

$$\begin{aligned}\max\{&\max\{R(\beta_1 y_1 + \cdots + \beta_t y_t)\} - (\alpha_1 x_1 + \cdots + \alpha_s x_s)^{\mathsf{T}} R(\beta_1 y_1 + \cdots + \beta_t y_t),\\ &\max\{C^{\mathsf{T}}(\alpha_1 x_1 + \cdots + \alpha_s x_s)\} - (\alpha_1 x_1 + \cdots + \alpha_s x_s)^{\mathsf{T}} C(\beta_1 y_1 + \cdots + \beta_t y_t)\}.\end{aligned} \tag{3}$$

A direct observation is that we can suppose without loss of generality that $s \le t$. Otherwise, we simply exchange the positions of the players.

We first consider the simplest situation where $s = 1$. In this case, $\Delta_s$ is degenerated to a single point. (3) is degenerated to the following form:

$$\begin{aligned}\max\{&\max\{R(\beta_1 y_1 + \cdots + \beta_t y_t)\} - x_1^{\mathsf{T}} R(\beta_1 y_1 + \cdots + \beta_t y_t),\\ &\max\{C^{\mathsf{T}} x_1\} - x_1^{\mathsf{T}} C(\beta_1 y_1 + \cdots + \beta_t y_t)\}.\end{aligned} \tag{4}$$

This can further be expanded to:

$$\begin{aligned}\max\{&\beta_1 (Ry_1)_1 + \cdots + \beta_t (Ry_t)_1 - \beta_1 \left(x_1^{\mathsf{T}} Ry_1\right) - \cdots - \beta_t \left(x_1^{\mathsf{T}} Ry_t\right), \ldots,\\ &\beta_1 (Ry_1)_m + \cdots + \beta_t (Ry_t)_m - \beta_1 \left(x_1^{\mathsf{T}} Ry_1\right) - \cdots - \beta_t \left(x_1^{\mathsf{T}} Ry_t\right),\\ &\left(C^{\mathsf{T}} x_1\right) - \beta_1 \left(x_1^{\mathsf{T}} Cy_1\right) - \cdots - \beta_t \left(x_1^{\mathsf{T}} Cy_t\right)\}.\end{aligned} \tag{5}$$

Now, the objective function becomes the maximum of $m+1$ functions being all linear in $\beta$, respectively. In addition, the constraint $\beta \in \Delta_t$ is also linear in $\beta$. Using a standard transformation, we can transform the problem into a linear program with $t+1$ variables and $m+t+2$ constraints. Since the linear program can be solved in polynomial time [32], we obtain a polynomial-time optimal $(1,t)$-mixing algorithm.

Then, we consider the general optimal $(s,t)$-mixing problem. In this case, all terms in (3) are non-degenerated. There are three major components in (3): inner maximum terms $\max\{R(\beta_1 y_1 + \cdots + \beta_t y_t)\}$ and $\max\{C^{\mathsf{T}}(\alpha_1 x_1 + \cdots + \alpha_s x_s)\}$, bilinear terms $(\alpha_1 x_1 + \cdots + \alpha_s x_s)^{\mathsf{T}} R(\beta_1 y_1 + \cdots + \beta_t y_t)$ and $(\alpha_1 x_1 + \cdots + \alpha_s x_s)^{\mathsf{T}} C(\beta_1 y_1 + \cdots + \beta_t y_t)$, and the outermost maximum operator (i.e., $\max\{f_R, f_C\}$). Different terms present different difficulties:

1. Inner maximum terms are piecewise-linear in $\beta$ and $\alpha$, respectively, thus convex but non-differentiable.

2. Bilinear terms are bilinear in $\beta$ and $\alpha$, thus differentiable but nonconvex.

3. The outermost maximum operator is non-differentiable.

Our solution is sketched below:

1. Since the inner maximum terms have a piecewise-linear structure, we can divide the problem into subproblems on each linear piece.

    - To determine the linear pieces, we resort to the famous *half-plane intersection problem* in computational geometry.

- This overcomes the first difficulty.

2. For each subproblem, we derive necessary conditions for the global optima. Thus, by scanning all points satisfying the conditions, we can find a global optimum.

   - To give the necessary conditions, we combine discrete geometry (linear-algebraic characterizations of polytopes) and optimization (KKT conditions).
   - To scan these points, we rewrite the conditions into several optimization problems (e.g., univariate quadratic programs and linear programs) and solve them.
   - This overcomes the second and the third difficulty.

3. Finally, we show that there are polynomial number of linear pieces and in each linear piece a constant number of points to check. Thus, we can find the global optimum in polynomial time.

To demonstrate the main idea of the algorithms, below we sketch the implementation of the optimal $(2,2)$-mixing algorithm over arbitrary strategies $x_1, x_2$ and $y_1, y_2$.

Denote the set of all possible convex combinations as $\mathcal{A} := \{(\alpha_1 x_1 + \alpha_2 x_2, \beta_1 y_1 + \beta_2 y_2) : \alpha, \beta \in \Delta_2\}$. Observe that the form of the function $f_R$ over the mixing region $\mathcal{A}$ is:

$$\begin{aligned} & f_R(\alpha_1 x_1 + \alpha_2 x_2, \beta_1 y_1 + \beta_2 y_2) \\ =& \max\{R(\beta_1 y_1 + \beta_2 y_2)\} - (\alpha_1 x_1 + \alpha_2 x_2)^\mathsf{T} R(\beta_1 y_1 + \beta_2 y_2). \end{aligned}$$

Note that $\beta_2 = 1-\beta_1$, thus the max term can be written as $\max\{\beta_1 R(y_1 - y_2) + Ry_2\}$. It has the form of the maximum of $m$ linear functions about $\beta_1$, which is piecewise linear in $\beta_1$.

Now, we want to compute the exact form of the piecewise linear function, given by a sequence of breakpoints $0 = b_1 \leq \cdots \leq b_t = 1$ $(t \leq m+1)$ so that $f$ is linear in $\beta$ on each $[b_i, b_{i+1}]$. We need to compute the exact form of this problem, that is to compute the value of $R(y_1 - y_2)$ and $Ry_2$ with time $O(mn)$. Then, this becomes a famous problem in computational geometry called the *envelope problem*, which can be solved in time $O(m \log m)$.

Similarly, we can compute the linear pieces given by breakpoints $0 = a_1 \leq \cdots \leq a_s = 1$ $(s \leq n+1)$ in time $O(n \log n)$. Therefore, on each grid $[a_i, a_{i+1}] \times [b_j, b_{j+1}], i \in [s], j \in [t]$, both $f_R$ and $f_C$ are linear in $\alpha$ and $\beta$, respectively.

Then, we minimize the objective function over each grid and compare the results to take the one with minimal $f$ value. By doing so, we obtain the global minimum of $f$ on region $\mathcal{A}$.

On each grid, the objective function is in the form of the maximum of two bilinear functions $g_1$ and $g_2$. However, it is still non-differentiable. We apply the KKT condition from continuous optimization to obtain the necessary optimal conditions for this problem. We can show that the minimum must be attained at the following three kinds of points:

1. Points where the partial derivative of $g_1$ or $g_2$ with respect to $\alpha$ or $\beta$ is zero.

2. The four vertices of the grid.

3. Points where $g_1 = g_2$.

Now we show that the number of points to be checked of each kind is bounded by a constant. For the first kind, since the partial derivatives of bilinear functions $g_1$ and $g_2$ are linear, the problem finally reduces to a univariate linear program, which can be solved in constant time. For the second kind, there are only four points. Finally, for the third kind, we can solve the relation between $\alpha$ and $\beta$ from $g_1 = g_2$ and substitute it into the objective function. Then, we obtain a univariate quadratic program, which can be easily minimized by checking at most six points.

In words, on each grid, we only need constant time to compute the minimum $f$. Thus, by scanning over all grids in $O(mn)$ time, we can compute the global minimum of $f$ on $\mathcal{A}$. The total complexity is given by $O(mn + m \log m + n \log n) = O(mn)$.

### *3.3. Optimal Mixing Algorithms*

In this section, we present the detailed algorithms for the optimal mixing problems. First, we present several auxiliary results to address the two basic difficulties mentioned in Section 3.2. Then, we present the optimal mixing algorithms for the optimal $(1, t)$-mixing problem, the optimal $(2, 2)$-mixing problem, and the optimal $(2, 3)$-mixing problem.

#### *3.3.1. Linear Piece Partitioning*

In this part, we provide the solution for the first difficulty concerning the non-differentiability of the inner maximum terms by linear piece partitioning. The idea is to partition the domain into regions where both $f_R$ and $f_C$ are linear in $\alpha$ and $\beta$, respectively. To make a concise description, for a function

$F : X \to \mathbb{R}$, we say a *linear piece* of $F$ is the maximal region $\Omega \subseteq X$ such that $F$ is linear in each variable on $\Omega$.

Consider function $\max\{R(\beta_1 y_1 + \cdots + \beta_t y_t)\}$. We have:

$$\begin{aligned} &\max\{R(\beta_1 y_1 + \cdots + \beta_t y_t)\} \\ =&\max\{\beta_1 R y_1 + \cdots + \beta_t R y_t\} \\ =&\max_{1\le i\le m}\{\beta_1 (Ry_1)_i + \cdots + \beta_t (Ry_t)_i\}. \end{aligned} \tag{6}$$

It is the maximum of $m$ linear functions in $\beta$. Therefore, we can partition the domain into several linear pieces, as illustrated in Figure 1.

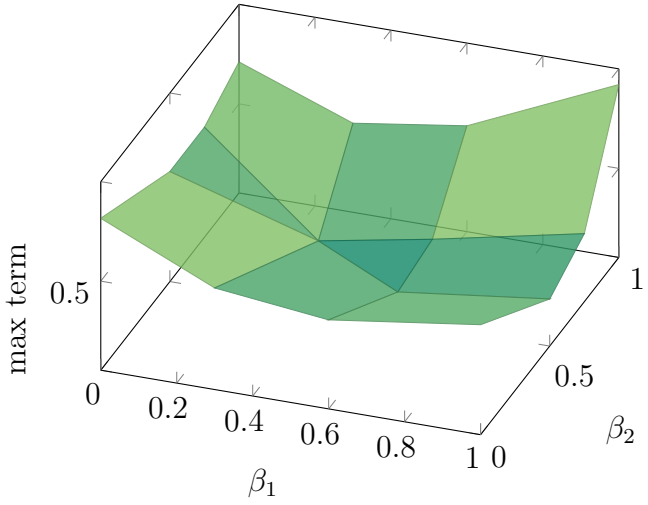


Figure 1: Illustration of linear pieces when $t = 2$. The function figure of $\max\{R(\beta_1 y_1 + \beta_2 y_2)\}$ is in green color. Each different green color represents a different linear piece.

An important observation is that the linear pieces can be expressed by linear inequalities. When the $i$th linear function attains the maximum, we have

$$\forall j \in [m], j \neq i, \beta_1 (Ry_1)_i + \cdots + \beta_t (Ry_t)_i \ge \beta_1 (Ry_1)_j + \cdots + \beta_t (Ry_t)_j.$$

Namely,

$$\forall j \in [m], j \neq i, \beta_1[(Ry_1)_i - (Ry_1)_j] + \cdots + \beta_t[(Ry_t)_i - (Ry_t)_j] \geq 0.$$

Thus, each linear piece of the $\max\{Ry\}$ term can be determined by $m-1$ inequalities, and forming a (possibly empty) polytope (see Appendix C for the formal definition). There are $m$ such polytopes, denoted by $P_1^C, \ldots, P_m^C$, where notation $C$ means that they are collections of the column player's strategies.

Similarly, the linear pieces of $\max\{C^\mathsf{T} x\}$ term can be determined by $n-1$ inequalities. We denote the polytopes by $P_1^R, \ldots, P_n^R$.

Our solution is to divide the problem into each polytope $P_i^R \times P_j^C$ $(i, j \in [n] \times [m])$, which we call the *separated polytope*. In this way, we can eliminate the inner max term of (3) and obtain the following problem:

$$\begin{aligned} \min_{\alpha \in (\Delta_s \cap P_i^R), \beta \in (\Delta_t \cap P_j^C)} \max\{ & \beta_1 (Ry_1)_i + \cdots + \beta_t (Ry_t)_i \\ & -(\alpha_1 x_1 + \cdots + \alpha_s x_s)^\mathsf{T} R(\beta_1 y_1 + \cdots + \beta_t y_t), \\ & \alpha_1 (C^\mathsf{T} x_1)_j + \cdots + \alpha_s (C^\mathsf{T} x_s)_j \\ & -(\alpha_1 x_1 + \cdots + \alpha_s x_s)^\mathsf{T} C(\beta_1 y_1 + \cdots + \beta_t y_t)\}. \end{aligned} \tag{7}$$

To meet further needs, we also care about solving and expressing the separated polytopes efficiently. Formally, we want to solve the following problem:

**Definition 3.2** ($(t, m)$-separation algorithm)**.**

- Input*: dimension $m$, $t$ vectors $x_1, \ldots, x_t$ in $\mathbb{R}^m$.*

- Output*: a clockwise enumeration of vertices of* $P_1, \ldots, P_m$ *such that* $P_i$ *is the polytope* $\{\beta \in \Delta_t : \beta_1[(x_1)_i - (x_1)_j] + \cdots + \beta_t[(x_t)_i - (x_t)_j] \geq 0, \forall j \in [m]\}$.

When $t \leq 3$, we represent the polytopes with a clockwise enumeration of its vertices. In this case, the separation problem can be restated as a famous problem in computational geometry called the *half-plane intersection problem* [33]. Benefiting from geometric intuitions, we obtain polynomial-time algorithms for $t \leq 3$, as stated below.

**Theorem 3.2.** *There exists a* $(2, m)$*-separation algorithm in time* $O(m \log m)$.

**Theorem 3.3.** *There exists a* $(3, m)$*-separation algorithm in time* $O(m^2 \log m)$.

While these algorithms are standard in computational geometry, for completeness, we still provide the detailed algorithms and complexity analysis in Appendix A.4 and Appendix A.5.

We also note that if we only require the "appropriate" expression in Definition 3.2 to be a vertex enumeration of the polytope, then in general cases, it can be stated in the famous *vertex enumeration problem*.

Suppose we are given a polytope in $\mathbb{R}^t$ determined by $m$ inequalities, then McMullen's upper bound theorem [34] gives a close upper bound $\mathcal{A}(m^{t/2})$ on the number of its vertices $|V|$.

Several algorithms are proposed for the vertex enumeration problem. Using the pivoting method, Dyer [35] proposed an $O(mt^2|V|)$-time algorithm. Then, Avis and Fukuda [36] proposed an $O(mt|V|)$-time algorithm, which has remained state-of-the-art since then. For a brief summary of this subject, see [37] as a reference.

We note that our algorithms (Algorithm 2 and Algorithm 3) are faster than these algorithms in the corresponding cases. Indeed, when $t = 2$, the time complexity of vertex enumeration is $O(m^3)$. For $t = 3$, the time complexity is $O(m^{3.5})$.

It is also worth mentioning the complexity results regarding this problem. For the unbounded case (polyhedra), vertex enumeration has been proven to be NP-hard [38]. However, for the bounded case (the case of our problem, which is bounded in $\Delta_t$), it is still an open problem. There is a strong indication of NP-hardness, though, as [39] proved that uniformly sampling the vertices is NP-hard.

*3.3.2. Optimization over Polytopes*

Using linear piece separation, we have transferred the form of the problem into solving the subproblem (7). To solve this subproblem, we first derive optimal conditions for a slightly generalized problem. We present the results used for algorithms in this part. Since the proof of these results is either standard or technical, we defer them to the appendix.

For differentiable functions $g_1, g_2$ and polytope $S$, consider the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \max\{g_1(x), g_2(x)\} \\ \text{s.t.} \quad & x \in S. \end{aligned} \tag{8}$$

We begin our preparation by a direct application of the KKT condition (see theorem 12.1 in [40] for details).

**Lemma 3.1.** *Consider any $U \in \mathbb{R}^{k\times n}, V \in \mathbb{R}^k, R \in \mathbb{R}^{j\times n}, T \in \mathbb{R}^j$ such that every row of $U, R$ is not zero. Define a convex polytope $S = \{x \in \mathbb{R}^n : Ux \leq$*

$V, Rx = T\}$ . *Suppose* $g_1$ *and* $g_2$ *are two real-valued differentiable functions defined on* $S$*. Set* $g = \max\{g_1, g_2\}$*. If the ranges of* $g_1$,$g_2$ *on* $S$ *are* $[m_1, M_1]$ *and* $[m_2, M_2]$*, respectively, then we have:*

*1. If* $m_1 \geq M_2$, $\min_S g(x) = m_1$*. The minimum is attained precisely on set* $g_1^{-1}(m_1)$*.*

*2. If* $m_2 \geq M_1$, $\min_S g(x) = m_2$*. The minimum is attained precisely on set* $g_2^{-1}(m_2)$*.*

*3. Otherwise,* $\min_S g(x) = \min_{S^*} g(x)$ *, where* $S^*$ *is the union of following sets:*

$$\{x \in S : g_1(x) = g_2(x)\},$$

$$\left\{ \begin{array}{l} x \in S : g_1(x) > g_2(x), \\ \\ \exists \lambda \geq 0 \ \textit{such that} \ \nabla g_1(x) + \lambda^{\mathsf{T}} \begin{pmatrix} U \\ R \end{pmatrix} = 0, \ \forall i \in [k], \ \lambda_i (U_i x - V_i) = 0 \end{array} \right\},$$

$$\left\{ \begin{array}{l} x \in S : g_2(x) > g_1(x), \\ \\ \exists \lambda \geq 0 \ \textit{such that} \ \nabla g_2(x) + \lambda^{\mathsf{T}} \begin{pmatrix} U \\ R \end{pmatrix} = 0, \forall i \in [k], \lambda_i (U_i x - V_i) = 0 \end{array} \right\}.$$

*And the minimum must be attained on* $S^*$*.*

The proof is presented in Appendix A.1.

Now we turn to polytopes. For concepts in polytopes, see Appendix C and textbook [41]. The following proposition captures the relationship of geometric properties and constraint expressions, which helps in the further analysis of the minimization problem on a certain polytope.

**Proposition 3.1.** *Consider the polytope* $S = \left\{x \in \mathbb{R}^n : a_i^\mathsf{T} x \le b_i, \forall i \in [k]\right\}$, *where* $a_i \in \mathbb{R}^n \setminus \{0\}$, $b_i \in \mathbb{R}$. *Suppose the dimension of* $S$, *denoted by* $\dim(S)$, *is* $m \le n$. *Then we have*

1. *There exists vectors* $u_1, \ldots, u_{n-m} \in \mathbb{R}^n$ *and real numbers* $v_1, \ldots, v_{n-m} \in \mathbb{R}$ *such that the affine hull* $\mathrm{aff}(S)$ *of* $S$ *can be written in the form* $\left\{x \in \mathbb{R}^n : u_i^\mathsf{T} x = v_i, \forall i \in [n-m]\right\}$.

2. *Vector* $d$ *is parallel to* $S$ *(denoted by* $d \parallel S$*) if and only if for every* $i \in [n-m]$, $u_i^\mathsf{T} d = 0$.

*The representations of geometric concepts about* $S$ *can be presented in the following order.*

3. *(Representation of* $S$*) There exists a set* $W \subseteq [k]$ *of indices such that:*

$$S = \left\{x \in \mathbb{R}^n : u_i^\mathsf{T} x = v_i, \forall i \in [n-m]\right\} \cap \left\{x \in \mathbb{R}^n : a_i^\mathsf{T} x \le b_i, \forall i \in W\right\}.$$

4. *(Representation of boundary* $\partial S$ *and interior* $S^\circ$*) Moreover:*

$$\partial S = \{x \in S : \exists i \in W, a_i^\mathsf{T} x = b_i\}, S^\circ = \{x \in S : \forall i \in W, a_i^\mathsf{T} x < b_i\}.$$

5. *(Representation of facets of* $S$*) For every* $j \in W$, $S'_j := \left\{x \in S : a_j^\mathsf{T} x = b_j\right\}$ *is a distinct facet of* $S$, *and every facet of* $S$ *coincides with exactly one* $S'_j$.

6. *(Representation of faces of* $S$*) For any face* $T$ *of* $S$, $\dim(T) \le m-1$, *and* $T$ *can be expressed as the intersection of facets of* $S$.

The proof is presented in Appendix A.2.

Now we combine discrete geometry and optimization. We derive three corollaries from Lemma 3.1 to deal with simpler cases.

**Corollary 3.1.** *For any convex polytope $S \in \mathbb{R}^n$ such that $\dim(S) = n$, suppose without loss of generality that it has a form that $S = \{x \in \mathbb{R}^n : Ux \leq V\}$, where $U \in \mathbb{R}^{m\times n}$, $V \in \mathbb{R}^m$ and no rows of $U$ are zero. We have the following statements.*

*1. The minimum of $g$ on $S$ must be obtained on*

$$\begin{aligned} S^+ =& \partial S \cup \{x \in S : \nabla g_1(x) = 0\} \cup \{x \in S : \nabla g_2(x) = 0\} \\ & \cup \{x \in S : g_1(x) = g_2(x)\}. \end{aligned} \tag{9}$$

*2. Let $e_1, \ldots, e_n$ be the standard orthonormal basis. For any $e_i$, we can divide the facets of $S$ into two collections: $P_i$ and $N_i$ according to whether they are parallel to $e_i$. Define $\partial S_P = \bigcup_{T\in P_i} T$ and $\partial S_N = \bigcup_{T\in N_i} T$. $\partial S_P \cup \partial S_N = \partial S$. For any index $i$, statement 1 still holds if we substitute $\partial S$ with*

$$\left(\partial S_P \bigcap \bigcup_{k=1,2} \left\{x \in S : \frac{\partial g_k}{\partial x_i}(x) = 0\right\}\right) \bigcup \partial S_N.$$

*3. If the polytope $S$ has the form $[m_1, M_1] \times [m_2, M_2] \times \cdots \times [m_n, M_n]$ with $m_i < M_i$, then the minimum must be obtained on*

$$\begin{aligned} S^+ =& \{x \in \mathbb{R}^n : \forall i, x_i \in \{m_i, M_i\}\} \bigcup \\ & \bigcup_{i\in[n],k\in\{1,2\}} \left(\left\{x \in S : \frac{\partial g_k}{\partial x_i}(x) = 0\right\}\right) \bigcup \\ & \{x \in S : g_1(x) = g_2(x)\}. \end{aligned} \tag{10}$$

The proof is presented in Appendix A.3.

Statement 1 can be used to compute the minimum of $g$ on any polytope $S$ with recursion. Since all components of $S^+$ have at most $(n-1)$ dimensions ($\partial S$ can be split into many facets), we can compress certain dimensions and

recursively compute the $(n-1)$-dimensional case. Statement 3 is a special case of statement 2, where the polytope is a hyperrectangle. Although we only present algorithms to solve cases where $t \leq 3$, we present statements 2 and 3 in a very general form. They are useful for further investigation of cases with $t > 3$.

### *3.4. Detailed Algorithms*

With all above preparations, we are now able to derive our algorithms for the optimal mixing problem.

We first consider the optimal $(1, t)$-mixing problem. Note that this problem can be directly transformed into a linear program given by Algorithm 1. We denote the complexity of solving a standard-form linear program with $t$ variables and $m$ inequalities by $L(t, m)$, which is polynomial in $t$ and $m$. See, e.g. [42]. Then, the complexity of our optimal $(1, t)$-mixing algorithm is $O(mnt + L(t, m))$.

**Algorithm 1** Optimal $(1, t)$-mixing algorithm

**Input:** An $m \times n$ bimatrix game $(R, C)$, mixed strategies $x_1$ for the row player and $y_1, y_2, \ldots, y_t$ for the column player.

**Output:** $\beta \in \Delta_t$ that minimizes $f(x_1, \beta_1 y_1 + \cdots + \beta_t y_t)$.

1: Calculate and store the $m$-dimensional vectors $Ry_1, \ldots, Ry_t$ and the values $x_1^{\mathsf{T}} Ry_1 \ldots x_t^{\mathsf{T}} Ry_t$. `// This can be done by matrix multiplication within` $O(mnt + mt)$ `time.`

2: Solve the optimal $\alpha$ of the following linear program and output it.

$$\begin{aligned}
\min_{\alpha} \quad & h \\
\text{s.t.} \quad & h \geq \max(C^{\mathsf{T}} x_1) - \alpha_1(x_1^{\mathsf{T}} C y_1) - \cdots - \alpha_t(x_t^{\mathsf{T}} C y_t), \\
& \text{for every } i \in [m], \quad t \geq \alpha_1 (R y_1)_i + \cdots + \alpha_t (R y_t)_i - \\
& \qquad \alpha_1(x_1^{\mathsf{T}} R y_1) - \cdots - \alpha_t(x_t^{\mathsf{T}} R y_t), \\
& \text{for every } j \in [t], \quad \alpha_j \geq 0, \\
& \alpha_1 + \cdots + \alpha_t = 1.
\end{aligned}$$

`// The problem is to solving a non-negative linear programming problem with` $m+1$ `constraints and` $t+1$ `variables.`

To avoid overwhelming the paper with detailed case-by-case discussions, we only give sketches of the optimal $(2,2)$ and $(2,3)$-mixing algorithms here in Algorithm 2 and Algorithm 3. The full process, correctness, and time-complexity analysis are presented in Appendix A.6 and Appendix A.7.

**Algorithm 2** Optimal $(2,2)$-mixing algorithm

**Input:** A size $m \times n$ bimatrix game $(R, C)$, mixed strategies $x_1, x_2$ for the row player and $y_1, y_2$ for the column player.

**Output:** $\alpha, \beta \in \Delta_2$ that minimizes $f(\alpha_1 x_1 + \alpha_2 x_2, \beta_1 y_1 + \beta_2 y_2)$.

1: Apply the $(2,n)$-separation algorithm (see Appendix A.4) for $\alpha$ that outputs separated polytopes $P_i^R$, where $i \in [n]$ (actually intervals of $\alpha_1$). `//` `Time complexity` $O(n \log n)$

2: Apply the $(2,m)$-separation algorithm (see Appendix A.4) for $\beta$ that outputs separated polytopes $P_j^C$, where $j \in [m]$ (actually intervals of $\beta_1$). `// Time complexity` $O(m \log m)$

3: Compute the exact form of $F_i(\alpha,\beta) = f_i(\alpha x_1+(1-\alpha)x_2, \beta y_1+(1-\beta)y_2)$, where $i \in \{R, C\}$. `// Time complexity` $O(mn)$

4: **for** $i = 1:n$, $j = 1:m$ **do**

5: Minimize $f$ in each grid $P_i^R \times P_j^C$. Apply statement 3 in Corollary 3.1. It suffices to scan the following regions:

1. Points with $\partial F_k(\alpha,\beta)/\partial\alpha = 0$ or $\partial F_k(\alpha,\beta)/\partial\beta = 0$, where $k = R, C$.
2. The four vertices of its domain.
3. Points with $F_R(\alpha,\beta) = F_C(\alpha,\beta)$.

`// For details, see Appendix A.6.`

6: **end for**

`// We can show that each case can be done in constant time over` $m, n$. `Thus, the time complexity is` $O(mn)$.

7: Finally, compare the $f$-values of the minimum on the $mn$ grids and obtain the global minimum of $f$ on $\Delta_2 \times \Delta_2$. `// Time complexity` $O(mn)$

---

**Algorithm 3** Optimal $(2,3)$-mixing algorithm

---

**Input:** A size $m \times n$ bimatrix game $(R, C)$, mixed strategies $x_1, x_2$ for the row player and $y_1, y_2, y_3$ for the column player.

**Output:** $\alpha \in \Delta_2$, $\beta \in \Delta_3$ that minimizes $f(\alpha_1 x_1 + \alpha_2 x_2, \beta_1 y_1 + \beta_2 y_2 + \beta_3 y_3)$.

1: Apply the $(2,n)$-separation algorithm (Appendix A.4) for $\alpha$ that outputs separated polytopes $P_i^R, i \in [n]$ (Actually are intervals for $\alpha_1$). `// Time complexity` $O(n \log n)$

2: Apply the $(3,m)$-separation algorithm (Appendix A.5) for $\beta$ that outputs separated polytopes $P_j^C, j \in [m]$. `// Time complexity` $O(m^2 \log m)$

3: Compute the exact form of $F_i(\alpha, \beta, \gamma) = f_i(\alpha x_1 + (1-\alpha)x_2, \beta y_1 + \gamma y_2 + (1-\beta-\gamma)y_3), i \in R, C$. `// Time complexity` $O(mn)$

4: **for** $i = 1 : n$, $j = 1 : m$ **do**

5: Minimize $f$ in each grid $P_i^R \times P_j^C$. Apply statement 2 in Corollary 3.1, it suffices to scan the following regions:

1. $(\alpha, \beta)$ belongs to side surfaces of $S$ and
   (a) either there exists $k \in R, C$ such that $\partial F_k/\partial\gamma = 0$, or
   (b) $(\alpha, \beta)$ is in the intersection of side surfaces and top/bottom surfaces.
2. $(\alpha, \beta)$ belongs to top/bottom surfaces of $S$ and
   (a) there exists $k \in R, C$ such that either $\partial F_k/\partial\alpha = 0$ or $\partial F_k/\partial\beta = 0$, or
   (b) $(\alpha, \beta)$ is in the intersection of side surfaces and top/bottom surfaces.
3. $F_R(\alpha, \beta) = F_C(\alpha, \beta)$.
4. $\nabla F_R(\alpha, \beta) = 0$ or $\nabla F_C(\alpha, \beta) = 0$.

`// For details, see Section Appendix A.7`

6: **end for**

`// We can show that each case can be done in` $O(m)$ `time. Thus, the time complexity is` $O(m^2 n)$.

7: Finally, compare the $f$-values of the minimum on the $mn$ grids, and obtain the global minimum of $f$ on $\Delta_2 \times \Delta_3$. `// Time complexity` $O(mn)$

## 4. An Algorithm for Approximate Optimal Mixing Problems

In this section, we present an algorithm solving any $\epsilon$-optimal $(s,t)$-mixing problem. Recall that the $\epsilon$-optimal $(s,t)$-mixing problem is defined as follows.

**Definition 4.1** ($\epsilon$-Optimal $(s,t)$-mixing problem)**.** *Given $\epsilon > 0$, the $\epsilon$-optimal $(s,t)$-mixing problem has the following input and output:*

- Input*: Bimatrix game $(R,C)$, mixed strategies $x_1,\ldots,x_s$ of the row player and $y_1,\ldots,y_t$ of the column player, and an $\epsilon > 0$.*
- Output*: Coefficients $\tilde{\alpha} \in \Delta_s$, $\tilde{\beta} \in \Delta_t$ such that for any $\alpha \in \Delta_s, \beta \in \Delta_t$, the following inequality holds:*

  $$f(\tilde{\alpha}_1 x_1 + \cdots + \tilde{\alpha}_s x_s, \tilde{\beta}_1 y_1 + \cdots + \tilde{\beta}_t y_t) \leq f(\alpha_1 x_1 + \cdots + \alpha_s x_s, \beta_1 y_1 + \cdots + \beta_t y_t) + \epsilon.$$

The main result in this section is the following theorem.

**Theorem 4.1.** *There exists an algorithm solving any $\epsilon$-optimal $(s,t)$-mixing problem in time* $\mathrm{poly}(m,n,s,t)\left(e + \frac{e}{\sqrt{\epsilon/2}}\right)^{s+t}$ *and space* $\mathrm{poly}(m,n,s,t)$.

**Corollary 4.1.** *For $\epsilon$-optimal $(s,t)$-mixing problem, when $s,t$ are constant, there exists an FPTAS; when $s,t = O(\log n)$, there exists a PTAS.*

We note that term $\left(e + \frac{e}{\sqrt{\epsilon/2}}\right)^{s+t}$ is unlikely to be improved to $\mathrm{poly}(m,n,s,t,1/\epsilon)$. Otherwise, by taking all pure strategies as input, we can obtain an FPTAS for $\epsilon$-NE, which is proved by Theorem 1.3 in [2] to be impossible unless PPAD $\subseteq$ P.

As is observed in Section 3, the objective function $f$ contains the maximum terms and bilinear terms. The bilinear terms causes non-convexity, which makes it very hard to find the global minimum.

Based on such observations, our method adopts the following idea. First, we use small grids to cover the whole domain. Next, on each grid, we use linear functions to approximate the bilinear terms. After the linear approximation, the objective function becomes piecewise linear, which can be solved by linear programming. By a delicate selection of grid, we can ensure that the optimal solution of the linear approximation is close to the optimal solution of the original problem, thus giving a close enough approximation. Finally, we output the best solution among all grids.

The algorithm is described in Algorithm 4.

**Algorithm 4** Algorithm for $\epsilon$-Optimal $(s,t)$-Mixing Problem

**Input:** An $m \times n$ bimatrix game $(R, C)$, mixed strategies $x_1, \ldots, x_s$ of the row player and $y_1, \ldots, y_t$ of the column player, and a precision parameter $\epsilon > 0$.

**Output:** $\tilde{\alpha} \in \Delta_s$, $\tilde{\beta} \in \Delta_t$ that solves the $\epsilon$-optimal $(s,t)$-mixing problem.

1: Let $X := (x_1, \ldots, x_s)$ and $Y := (y_1, \ldots, y_t)$.

2:
$$g_R(\alpha, \beta) := f_R(X\alpha, Y\beta) = \max\{RY\beta\} - \alpha^\mathsf{T} X^\mathsf{T} RY\beta,$$
$$g_C(\alpha, \beta) := f_C(X\alpha, Y\beta) = \max\{C^\mathsf{T} X\alpha\} - \alpha^\mathsf{T} X^\mathsf{T} CY\beta,$$
$$g(\alpha, \beta) := \max\{g_R(\alpha, \beta), g_C(\alpha, \beta)\},$$
$$L_R(\alpha, \beta; \alpha_o, \beta_o) := \max\{RY\beta\} - \alpha_o^\mathsf{T} X^\mathsf{T} RY\beta - \alpha^\mathsf{T} X^\mathsf{T} RY\beta_o + \alpha_o^\mathsf{T} X^\mathsf{T} RY\beta_o,$$
$$L_C(\alpha, \beta; \alpha_o, \beta_o) := \max\{C^\mathsf{T} X\alpha\} - \alpha_o^\mathsf{T} X^\mathsf{T} CY\beta - \alpha^\mathsf{T} X^\mathsf{T} CY\beta_o + \alpha_o^\mathsf{T} X^\mathsf{T} CY\beta_o,$$
$$L(\alpha, \beta; \alpha_o, \beta_o) := \max\{L_R(\alpha, \beta; \alpha_o, \beta_o), L_C(\alpha, \beta; \alpha_o, \beta_o)\}.$$

3: Let $p := \left\lceil s/\sqrt{\epsilon/2} \right\rceil$ and $q := \left\lceil t/\sqrt{\epsilon/2} \right\rceil$.

4: Form the following points:

$$\left(\alpha_{(i_1,\ldots,i_s)}, \beta_{(j_1,\ldots,j_t)}\right) := \left(i_1/p, \ldots, i_s/p, j_1/q, \ldots, j_t/q\right),$$

where $i_1, \ldots, i_s \in [p]$, $\sum_{k=1}^{s} i_k = p$, $j_1, \ldots, j_t \in [q]$, and $\sum_{k=1}^{t} j_k = q$.

`// There are` $\binom{p+s-1}{s-1}\binom{q+t-1}{t-1}$ `points in total.`

5: **for** $i_1, \ldots, i_s \in [p]$, $\sum_{k=1}^{s} i_k = p$, $j_1, \ldots, j_t \in [q]$, $\sum_{k=1}^{t} j_k = q$ **do**

6: Define the region

$$\Gamma(i_1, \ldots, i_s, j_1, \ldots, j_t) := \{(\alpha, \beta) : \alpha \in \Delta_s, \beta \in \Delta_t,$$
$$\text{for every } k \in [s], \alpha_k \in [\tfrac{i_k - 1}{p}, \tfrac{i_k + 1}{p}], \text{ for every } l \in [t], \beta_l \in [\tfrac{j_l - 1}{q}, \tfrac{j_l + 1}{q}].\}$$

7: Solve the optimal $\alpha^*_{(i_1,\ldots,i_s)}, \beta^*_{(j_1,\ldots,j_t)}$ of the following optimization problem.

$$\begin{aligned} \min_{\alpha,\beta} \quad & L\left(\alpha, \beta; \alpha_{(i_1,\ldots,i_s)}, \beta_{(j_1,\ldots,j_t)}\right), \\ \text{s.t.} \quad & (\alpha, \beta) \in \Gamma(i_1, \ldots, i_s, j_1, \ldots, j_t). \end{aligned} \tag{11}$$

`// Note that this can be formulated in a linear program with` $m+n+3s+3t+2$ `constraints and` $s+t+1$ `variables. The problem is reduced to solving a non-negative linear programming.`

8: **end for**

9: Output the $\alpha^*_{(i_1,\ldots,i_s)}, \beta^*_{(j_1,\ldots,j_t)}$ with the smallest $g\left(\alpha^*_{(i_1,\ldots,i_s)}, \beta^*_{(j_1,\ldots,j_t)}\right)$ among all $i_1, \ldots, i_s, j_1, \ldots, j_t$.

---

Now we verify the correctness of Algorithm 4 and analyze its time complexity. We first show that the linear approximation is close to the original function.

We need the following lemma.

**Lemma 4.1.** *Consider two functions $f_1, f_2$ defined on the same set $\mathcal{X}$, if $|f_1(x) - f_2(x)| \le \epsilon$ for all $x \in \mathcal{X}$, then $|\min_x f_1(x) - \min_x f_2(x)| \le \epsilon$ and $|\max_x f_1(x) - \max_x f_2(x)| \le \epsilon$.*

*Proof.* Let $x_1$ be a global minimum point of $f_1$ and $x_2$ be a global minimum point of $f_2$. Then we have

$$\begin{aligned} f_2(x_2) &\geq f_1(x_2) - \epsilon \quad \text{(by assumption)} \\ &\geq f_1(x_1) - \epsilon. \quad \text{(since } x_1 \text{ is the minimizer of } f_1\text{)} \end{aligned}$$

Symmetrically, we have $f_1(x_1) \geq f_2(x_2) - \epsilon$. Thus we have

$$|f_1(x_1) - f_2(x_2)| \leq \epsilon,$$

as desired.

The proof of the maximum is similar. □

Then we have the following lemma, which shows that the linear approximation is good enough for the original problem on each grid.

**Lemma 4.2.** *On each* $\Gamma(i_1, \ldots, i_s, j_1, \ldots, j_t)$*, the following inequalities hold:*

- $|g(\alpha, \beta) - L(\alpha, \beta; \alpha_{(i_1,\ldots,i_s)}, \beta_{(j_1,\ldots,j_t)})| \leq \epsilon/2$,
- $|\min_{(\alpha,\beta)} g(\alpha, \beta) - \min_{(\alpha,\beta)} L(\alpha, \beta; \alpha_{(i_1,\ldots,i_s)}, \beta_{(j_1,\ldots,j_t)})| \leq \epsilon/2$.

*Proof.* To simplify the notation, we denote $\alpha_{(i_1,\ldots,i_s)}$ by $\alpha_o$ and $\beta_{(j_1,\ldots,j_t)}$ by $\beta_o$. We further denote $\Gamma(i_1, \ldots, i_s, j_1, \ldots, j_t)$ by $\Gamma_o$.

We first show that $|L_R - g_R| \leq \epsilon/2$ on $\Gamma_o$. For any $(\alpha, \beta) \in \Gamma(\alpha_o, \beta_o)$, by

definition, we have:

$$
\begin{aligned}
&|L_R(\alpha,\beta;\alpha_o,\beta_o) - g_R(\alpha,\beta)| \\
=&|(\alpha-\alpha_o)^\mathsf{T} X^\mathsf{T} RY(\beta-\beta_o)| \\
\le&\sum_{i=1}^{s}\sum_{j=1}^{t} |\alpha_i-\alpha_{o,i}|\cdot|\beta_j-\beta_{o,j}|\cdot|(X^\mathsf{T}RY)_{ij}| \\
\le&\sum_{i=1}^{s}\sum_{j=1}^{t} \frac{1}{pq}\underbrace{|(X^\mathsf{T}RY)_{ij}|}_{\le 1} \le \frac{st}{pq} \le \frac{st}{s/\sqrt{\epsilon/2}\cdot t/\sqrt{\epsilon/2}} = \epsilon/2.
\end{aligned}
$$

Similarly, we have $|L_C - g_C| \le \epsilon/2$ on $\Gamma_o$. Since $L = \max\{L_R, L_C\}$ and $g = \max\{g_R, g_C\}$, taking $\mathcal{X} = \{R, C\}$, by Lemma 4.1, we have $|L - g| = |\max_{x\in\mathcal{X}} L_x - \max_{x\in\mathcal{X}} g_x| \le \epsilon/2$ on $\Gamma_o$, which is the first part of the lemma.

Then, taking $\mathcal{X} = \Gamma_o$, by Lemma 4.1, we have

$$
\left|\min_{(\alpha,\beta)\in\mathcal{X}} g(\alpha,\beta) - \min_{(\alpha,\beta)\in\mathcal{X}} L(\alpha,\beta;\alpha_o,\beta_o)\right| \le \epsilon/2,
$$

which is the second part of the lemma. □

Next we show that all grids cover the whole domain $\Delta_s \times \Delta_t$.

**Lemma 4.3.** *For any $(\alpha,\beta) \in \Delta_s \times \Delta_t$, there exists $i_1,\dots,i_s,j_1,\dots,j_t$ such that $(\alpha,\beta) \in \Gamma(i_1,\dots,i_s,j_1,\dots,j_t)$.*

*Proof.* For any $(\alpha,\beta) \in \Delta_s \times \Delta_t$, take $(i_1,\dots,i_s,j_1,\dots,j_t)$ by the following procedure:

**Algorithm 5** Selection of $i_1,\dots,i_s$.

**Input:** $\alpha \in \Delta_s$, $\beta \in \Delta_t$.

**Output:** $i_1,\dots,i_s,j_1,\dots,j_t$.

1: **for** $k$ from 1 to $s$ **do**

2: **if** $\sum_{l=1}^{k-1}(\alpha_l - \frac{i_l}{p}) \geq 0$ **then**

3: Set $i_k = \lceil p\alpha_k \rceil$

4: **else**

5: Set $i_k = \lfloor p\alpha_k \rfloor$

6: **end if**

7: **end for**

---

The selection of $j_1, \ldots, j_t$ is similar. Now we show that the procedure is correct.

First, we show $i_1 + \cdots + i_s = p, i_1, \ldots, i_s \in [p]$. Since $\alpha \in \Delta_s$, it is clear that $i_1, \ldots, i_s \in [p]$. Then we show that the sum is equal to $p$.

For this purpose, we show $\left|\sum_{l=1}^{k}\left(\alpha_l - \frac{i_l}{p}\right)\right| < \frac{1}{p}$ for any $k \in [s]$ by induction over $k$.

- For $k = 1$, we have $\left|\alpha_1 - \frac{i_1}{p}\right| < \frac{1}{p}$, since $\alpha_1 \in [0, 1]$.
- For $k > 1$, suppose we have $\left|\sum_{l=1}^{k-1}\left(\alpha_l - \frac{i_l}{p}\right)\right| < \frac{1}{p}$ by induction hypothesis. If $\sum_{l=1}^{k-1}\left(\alpha_l - \frac{i_l}{p}\right) \geq 0$, then by $i_k = \lceil p\alpha_k \rceil$, we have $-\frac{1}{p} < \alpha_k - \frac{i_k}{p} \leq 0$, thus

$$-\frac{1}{p} < \sum_{l=1}^{k}\left(\alpha_l - \frac{i_l}{p}\right) = \sum_{l=1}^{k-1}\left(\alpha_l - \frac{i_l}{p}\right) + \left(\alpha_k - \frac{i_k}{p}\right) < \frac{1}{p} - 0 = \frac{1}{p}$$

  as desired. Another case is similar.

Thus, we have $\left|\sum_{l=1}^{s}\left(\alpha_l - \frac{i_l}{p}\right)\right| < \frac{1}{p}$.

Note that $\sum_{l=1}^{s}\alpha_l = 1$, we have $\left|1 - \sum_{l=1}^{s}\frac{i_l}{p}\right| < \frac{1}{p}$. Thus, $|p - \sum_{l=1}^{s} i_l| < 1$. Since $i_1, \ldots, i_s \in [p]$, we have $\sum_{l=1}^{s} i_l \in \mathbb{Z}$. Thus, we must have $i_1 + \cdots + i_s = p$.

Finally, we show that $(\alpha, \beta) \in \Gamma(i_1, \ldots, i_s, j_1, \ldots, j_t)$. It simply follows by the construction of $i_k$'s and $j_k$'s and the definition of $\Gamma$. $\square$

We summarize the above two lemmas in the following proposition.

**Proposition 4.1** (Correctness)**.** *Algorithm 4 is an $\epsilon$-optimal $(s,t)$-mixing algorithm.*

*Proof.* Suppose the exact solution of the optimal $(s,t)$-mixing problem is $(\alpha^*, \beta^*)$. By Lemma 4.3, there exists $i_1, \ldots, i_s, j_1, \ldots, j_t$ such that $(\alpha^*, \beta^*) \in \Gamma(i_1, \ldots, i_s, j_1, \ldots, j_t)$. For simplicity, we denote $L(\alpha, \beta; \alpha_{(i_1,\ldots,i_s)}, \beta_{(j_1,\ldots,j_t)})$ by $L(\alpha, \beta)$.

Consider the solution $(\alpha', \beta') = (\alpha^*_{(i_1,\ldots,i_s)}, \beta^*_{(j_1,\ldots,j_t)})$ in (11). It suffices to prove that

$$|g(\alpha^*, \beta^*) - g(\alpha', \beta')| \leq \epsilon.$$

By Lemma 4.2 part 1, we have $|g(\alpha^*, \beta^*) - L(\alpha', \beta')| = |\min_{\alpha,\beta} g(\alpha, \beta) - \min_{\alpha,\beta} L(\alpha, \beta)| \leq \epsilon/2$. By Lemma 4.2 part 2, we have $|g(\alpha', \beta') - L(\alpha', \beta')| \leq \epsilon/2$. Thus, by triangular inequality, we have $|g(\alpha^*, \beta^*) - g(\alpha', \beta')| \leq \epsilon$, as desired. $\square$

Finally, we analyze the time complexity of Algorithm 4, which completes the proof of Theorem 4.1.

*Proof of Theorem 4.1.* First, we need to compute $RY, X^\mathsf{T}RY, C^\mathsf{T}X, X^\mathsf{T}CY$ for further use. This can be done in $O(\text{poly}(m, n, s, t))$ time.

Then, the algorithm enters a loop that repeats $\binom{p+s-1}{s-1}\binom{q+t-1}{t-1}$ times. By

Stirling's approximation, we have

$$\begin{aligned}\binom{p+s-1}{s-1} &\le \frac{(p+s-1)^{s-1}}{(s-1)!} = \frac{(p+s-1)^{s-1}\cdot s}{s!} \\ &= O\left(\frac{(p+s-1)^{s-1}\sqrt{s}}{(\frac{s}{e})^s}\right) \\ &\le O\left(\frac{\left((1/\sqrt{\epsilon/2}+1)s\right)^s\sqrt{s}}{(\frac{s}{e})^s}\right) \\ &= O\left(\sqrt{s}\left(\left(\frac{1}{\sqrt{\epsilon/2}}+1\right)e\right)^s\right).\end{aligned}$$

Similarly, we have $\binom{q+t-1}{t-1} \le O\left(\sqrt{t}\left(1+\frac{e}{\sqrt{\epsilon/2}}\right)^t\right)$. Thus, the total number of iterations is $O\left(\sqrt{st}\left(e+\frac{e}{\sqrt{\epsilon/2}}\right)^{s+t}\right)$.

In each iteration, we need to solve (11). By a standard transformation, it is equivalent to the following linear program:

$$\begin{aligned}\min_{\alpha,\beta,t}\quad & t, \\ \text{s.t.}\quad & t \ge (RY)_i\beta - \alpha^{\mathsf{T}}X^{\mathsf{T}}RY\beta, \quad i\in[m], \\ & t \ge (C^{\mathsf{T}}X)_j\alpha - \alpha^{\mathsf{T}}X^{\mathsf{T}}CY\beta, \quad j\in[n], \\ & \alpha\in\Delta_s, \beta\in\Delta_t, \\ & \alpha_k \in [\tfrac{i_k-1}{p}, \tfrac{i_k+1}{p}], \quad k\in[s], \\ & \beta_l \in [\tfrac{j_l-1}{q}, \tfrac{j_l+1}{q}], \quad l\in[t].\end{aligned}$$

It is a linear program with $m+n+3s+3t+2$ constraints and $s+t+1$ variables, which can be solved in $\mathrm{poly}(m,n,s,t)$ time [32].

In total, the time complexity of the whole algorithm is given by

$$\mathrm{poly}(m,n,s,t) + \mathrm{poly}(m,n,s,t)\left(\frac{e}{\sqrt{\epsilon/2}}\right)^{s+t} = \mathrm{poly}(m,n,s,t)\left(\frac{e}{\sqrt{\epsilon/2}}\right)^{s+t}.$$

□

## 5. Applications to the Search-and-Mix Methods

In this section, we show how to apply the optimal mixing problem to make an instance-optimal enhancement to the search-and-mix methods in the literature. For another use case, we can also use the optimal mixing problem to assemble the different outputs of various algorithms for approximate NE in the literature. See Appendix B for some illustrative empirical results.

As is summarized by [12], in the literature, polynomial-time algorithms for approximate NE follow a *search-and-mix* method. Such methods can be divided into two phases. In the search phase, an algorithm computes several strategies of each player in polynomial time. In the mixing phase, the algorithm then make convex combinations of the selected strategies into several strategy profiles and outputs the profile with the minimum $f$ value. An illustration is presented in Figure 2.

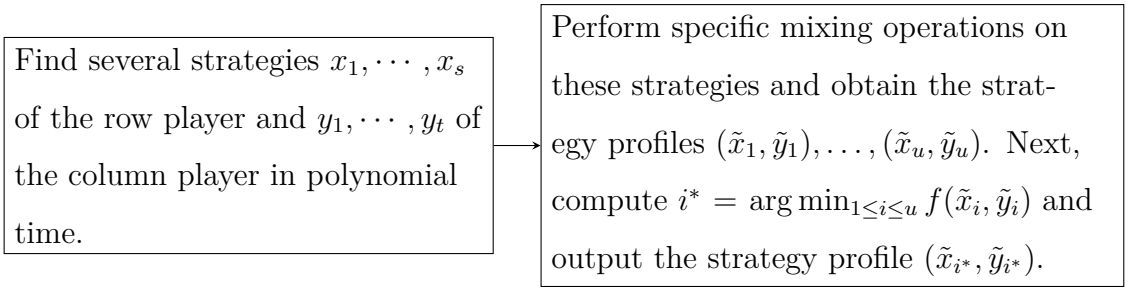


Figure 2: Procedure of the search-and-mix method in the literature

However, the mixing phases in the literature are *ad hoc*, since the mixing coefficients are selected specifically for corresponding search phase with

certain properties. A typical example is as follows.

**Example 5.1** (BBM algorithm [15])**.**

- Search phase*: Compute an NE* $(x^*, y^*)$ *of the zero-sum game* $(R - C, C - R)$[3]*. Let* $g_1 = f_R(x^*, y^*)$ *and* $g_2 = f_C(x^*, y^*)$*. By symmetry, assume without loss of generality that* $g_1 \geq g_2$*. Then compute* $r_1 \in \mathrm{br}_R(y^*)$ *and* $b_2 \in \mathrm{br}_C(r_1)$*.*

- Mixing phase*: Mix strategies in the search phase and obtain strategy profiles* $(x^*, y^*)$ *and* $(r_1, (1-\delta_2)y^* + \delta_2 b_2)$*, where* $\delta_2 = (1-g_1)/(2-g_1)$*. Output the one with the smaller* $f$ *value.*

*Note that the mixing coefficient* $\delta_2$ *is chosen specifically for this search phase to produce an optimal approximation bound of* 0.38*. If we choose* $\delta_2$ *to be other values, for example,* $1/2$*, then it is not hard to show that the approximation guarantee will only be* 0.5*.*

Now, we relate the optimal mixing problem to the search-and-mix methods in the literature. The traditional ad hoc designed mixing phases focus too much on the worst case and not useful in practice. However, from the perspective of our work, the mixing phase is essentially an optimal mixing problem. We can use the approximate and exact optimal mixing problem to design new mixing phases for the search-and-mix methods, which computes the *instance-optimal* mixing coefficients. The new procedure is presented in Figure 3.

[3]Computing NE in zero-sum games can be modeled by a linear program and thus can be solved in polynomial time. See, e.g., [24].

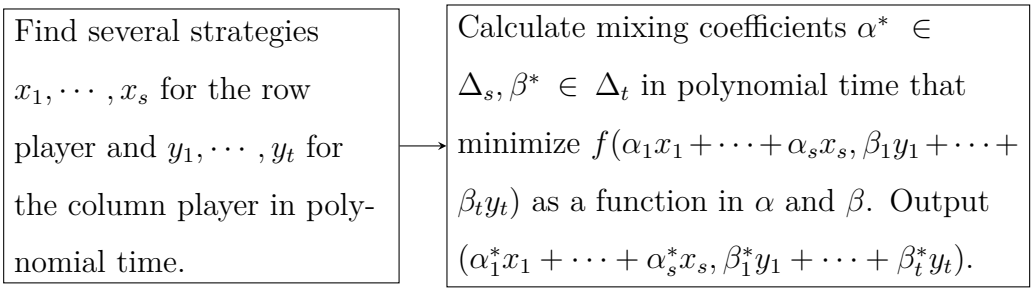


Figure 3: The new procedure for the search-and-mix method

Our exact algorithm for the optimal mixing problem can cover the need of *all* mixing phases in the literature. Moreover, our approximation algorithm can be used for most future mixing phases. For any constant number of strategies in the search phase, our approximation algorithm is an FPTAS. Beyond that, when $s, t = O(\log n)$, our approximation algorithm is a PTAS and when $s, t = \text{poly}(\log n)$, our approximation algorithm is a QPTAS. They can all be used in the new polynomial-time procedure of the search-and-mix methods.

To conclude this section, as an example, we show how to design a new mixing phase for Example 5.1 using the optimal mixing problem.

**Example 5.2** (New mixing phase for the BBM algorithm)**.** *Recall that in the search phase we obtain $x^*, r_1$ for the row player and $y^*, b_2$ for the column player. Then, the new mixing phase is to input payoff matrices $R, C$, row player's mixed strategies $x^*, r_1$, column player's mixed strategies $y^*, b_2$, solve the optimal $(2,2)$-mixing problem to obtain $\alpha^*, \beta^*$ and then output $(\alpha^* x^* + (1-\alpha^*) r_1, \beta^* y^* + (1-\beta^*) b_2)$.*

## 6. Conclusion and Discussion

In this paper, we study the optimal mixing problem of approximate Nash equilibrium computation in bimatrix games. We develop algorithms for the exact and approximate optimal mixing problems. These algorithms can be used to enhance and integrate arbitrary existing constant approximate NE algorithms, offering a powerful tool for the design of approximating NE algorithms. Moreover, these algorithms allow us to explore the implications of support restrictions on approximate NE, and provide the first upper bound on this problem.

### *6.1. Difficulties in Studying Exact Optimal Mixing Algorithms*

In Theorem 1.1, we stop at $(2, 3)$ for the exact optimal mixing problem. However, there is a fair reason for this. In general, the optimal mixing problem takes the form of a quadratically constrained quadratic program (QCQP). Recall that the objective function in Definition 3.1 can be expanded as follows:

$$\begin{aligned}\max\{\, &\max\{R(\beta_1 y_1 + \cdots + \beta_t y_t)\} - (\alpha_1 x_1 + \cdots + \alpha_s x_s)^{\mathsf{T}} R(\beta_1 y_1 + \cdots + \beta_t y_t),\\ &\max\{C^{\mathsf{T}}(\alpha_1 x_1 + \cdots + \alpha_s x_s)\} - (\alpha_1 x_1 + \cdots + \alpha_s x_s)^{\mathsf{T}} C(\beta_1 y_1 + \cdots + \beta_t y_t)\}.\end{aligned}$$

By rewriting the maximum operators as constraints, we can see that the optimal mixing problem is indeed a QCQP:

$$
\begin{aligned}
\min_{\alpha,\beta,z_1,z_2,w} \quad & w \\
\text{s.t.} \quad & \sum_{i=1}^{s} \alpha_i = 1, \quad \alpha_i \geq 0, \quad \sum_{j=1}^{t} \beta_j = 1, \quad \beta_j \geq 0, \\
& x = \sum_{i=1}^{s} \alpha_i x_i, \quad y = \sum_{j=1}^{t} \beta_j y_j, \\
& z_1 \geq R_i^{\mathsf{T}} y \quad \text{for all } i, \\
& z_2 \geq x^{\mathsf{T}} C_j \quad \text{for all } j, \\
& w \geq z_1 - x^{\mathsf{T}} R y, \quad w \geq z_2 - x^{\mathsf{T}} C y.
\end{aligned}
$$

As a QCQP, the optimal mixing problem is an optimization problem. By using the KKT conditions, the optimization problem can be reduced to a system of polynomial equations and inequalities:

$$
\begin{aligned}
& f_i(x_1, \ldots, x_s, y_1, \ldots, y_t) = 0, \quad i = 1, \ldots, \\
& g_j(x_1, \ldots, x_s, y_1, \ldots, y_t) \leq 0, \quad j = 1, \ldots,
\end{aligned}
$$

where $f_i$ and $g_j$ are polynomials. The exact optimal mixing problem is to find a solution to this system of polynomial equations and inequalities. Such a system is called an *algebraic variety* in the context of algebraic geometry. Thus, the exact solution is an element belonging to an algebraic variety.

For the case of $(2, 3)$, this paper shows that the solution can be reduced to a solution of a univariate quintic equation, which can be solved radically. However, for the case of $(3, 3)$, it is not even clear how to reduce the problem to a univariate equation. This phenomenon is not unique in NE computation. For example, the exact NE of a 3-player game is also an algebraic variety, and it is still not clear how to solve it using a Turing machine [43].

The difficulties in studying the exact optimal mixing problem or NE lie essentially in the real algebraic nature of the problem. In computability theory, the Blum-Shub-Smale machine [44], or BSS machine, is a model of computation intended to describe computations over the real numbers. Indeed, using the BSS machine, it is possible to solve the optimal mixing problem. However, the BSS machine surpasses the computability of the Turing machine, as it can represent uncountable sets, while the Turing machine can only represent countable sets. Thus, the exact optimal mixing problem is probably not computable by a Turing machine.

### *6.2. Future Directions*

This work also brings up several open questions:

- **How does the existence of NE benefit the computation of NE?** In Table 1, the upper bound of approximate NE under restrictions is far greater than the upper bound of approximate NE without restrictions. This suggests that the existence of NE is crucial for the computation of approximate NE. The optimal mixing problem provides a new methodology to study the computational complexity of approximate NE without the existence guarantee. We hope it can be used to understand the benefit of the NE existence to approximate NE computation.

- **How to distinguish the optimization problem, function problem, and the decision problem of approximate NE?** Note that the literature of approximate and exact NE is composed of three parts: the optimization problem, the function problem, and the decision problem. Their relations are very intricate but important in understanding

the underlying structure of NE computation. However, such differences are often ignored in the literature. Thus, it is important to distinguish them and understand their relations.

- **What is the relationship between approximate NE and QCQP?** In our optimization perspective, the approximate NE computation problem is a special case of the quadratically constrained quadratic program (QCQP) problem. We utilize the QCQP structure to develop algorithms for the optimal mixing problem. On the other hand, QCQP is known to be NP-hard [45] and has no constant approximation algorithm unless NP = P [46], while the constant approximate NE computation has quasi-polynomial time algorithms. It suggests that there is some interesting connection between the approximate NE computation and a subclass of QCQP problem.

- **How to extend our approach to compute approximate NE with other restrictions?** Note that the core component of our approach is the utilization of the bilinear structure of $f$ and grid covering. Thus, by writing other approximate NE restrictions, like maximum social welfare, as optimization problems with bilinear term, we can extend our approach to compute approximate NE with other restrictions.

## CRediT authorship contribution statement

**Xiaotie Deng:** Supervision, Project administration, Writing - Review & Editing, Funding acquisition. **Dongchen Li:** Conceptualization, Methodology, Formal analysis, Writing - Original Draft, Writing - Review & Editing,

Investigation, Software, Funding acquisition. **Hanyu Li:** Conceptualization, Methodology, Formal analysis, Writing - Original Draft, Writing - Review & Editing, Visualization, Investigation.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

No data was used for the research described in the article.


## Acknowledgements

This work is supported by Natural Science Foundation of China (Grant No. 62172012) and Natural Science Foundation of China (Grant No. 6212290003).


## Appendix A. Missing Proofs and Algorithms

*Appendix A.1. Proof of Lemma 3.1*

In case 1, take any $x \in S$. Then $g_2(x) \leq M_2 \leq m_1 \leq g_1(x)$. Therefore, $\min_S g(x) = \min_S g_1(x) = m_1$. The minimum is obtained exactly in $\{x \in S : g_1(x) = m_1\}$, namely $g_1^{-1}(m_1)$.

Case 2 is symmetric to case 1, so we omit it.

For case 3, we consider which set function $g$ reaches the minimum. Suppose that $x$ is a minimum, and $g_1(x) \neq g_2(x)$, then suppose without loss of

generality that $g_1(x) > g_2(x)$. By the continuity of $g_1, g_2$, $g_1 > g_2$ in some neighborhood of $x$. Therefore, $g = g_1$ in this neighborhood, so $x$ must be a local minimum for $g_1$. The problem becomes solving the first-order condition for optimization with linear constraints. Thus $x$ must also be a KKT point of $g_1$ and satisfy the KKT condition given in the statement.

*Appendix A.2. Proof of Proposition 3.1*

We prove the statement in the order of 1, 2, 3, 5, 4, and 6.

*Proof of Statement 1 and 2.* For statement 1, we note that notation $\dim(S) = m$ means that the smallest affine space containing it has dimension $m$. Therefore, the space can be expressed by the solution of $(n - m)$ linear equations, say $u_i^\mathsf{T} x = v_i, i \in [n - m]$. Moreover, the remaining constraints cannot contain any equation like $u^\mathsf{T} x = v$; otherwise, $\dim(S) \leq m - 1$, which violates the definition of the affine hull.

For statement 2, by the definition of parallel, $d \parallel S$ if and only if there exists a line segment defined by $x_0, x_1 \in S$ such that $x_1 = x_0 + \delta d$, where $\delta$ is a nonzero constant. Since $x_0, x_1 \in S$, we have for every $i \in [n - m]$, $u_i^\mathsf{T} x_0 = v_i$ and $u_i^\mathsf{T}(x_0 + \delta d) = v_i$, so $u_i^\mathsf{T} d = 0$. □

To prove statements 3, 5 and 4, we need the representation theorem for polytopes. We say a halfspace (inequality) $a^\mathsf{T} x \leq b$ is *facet-defining* (for polytope $P$) if $P \cap \{x \in \mathbb{R}^n : a^\mathsf{T} x \geq b\}$ defines a facet of $P$.

**Theorem Appendix A.1** (Representation theorem for polytopes, Theorem 2.15 in [41])**.** *A subset $P \subseteq \mathbb{R}^n$ is a polytope if and only if it can be described as a bounded intersection of facet-defining halfspaces, one for each facet, and*

*of the affine hull of $P$. Moreover, the facet-defining inequalities are uniquely determined (if we write them as $a_i^\mathsf{T} x \leq 1$), and none of them can be deleted.*

*Proof of Statement 3 and 5.* By Theorem Appendix A.1, we can write $S$ in the form

$$\mathrm{aff}(S) \cap \bigcap_{i \in \tilde{W}} \left\{ x \in \mathbb{R}^n : \tilde{a}_i^\mathsf{T} x \leq 1 \right\}.$$

Here, inequalities $\tilde{a}_i^\mathsf{T} x \leq 1$ are facet-defining. Since they are unique and cannot be deleted, each inequality $\tilde{a}_i^\mathsf{T} x \leq 1$ corresponds to a constraint $a_j^\mathsf{T} x \leq b_j$ with $a_j = b_j \tilde{a}_i$. For each $i \in \tilde{W}$, pick such a $j$. Then we have already selected an index subset $W$ of $[k]$. By statement 1, $\mathrm{aff}(S)$ can be written in the form of $\left\{ x \in \mathbb{R}^n : u_i^\mathsf{T} x = v_i, \forall i \in [n-m] \right\}$. Now we have proved statement 3.

For statement 5, since constraint $a_j^\mathsf{T} x \leq b_j$, $j \in W$ is fact-defining, again by Theorem Appendix A.1, $S'_j = \mathrm{aff}(S) \cap \left\{ x \in \mathbb{R}^n : a_j^\mathsf{T} x \leq b_j \right\}$ exactly represents a facet and *vice versa*. □

Now we prove statement 4.

*Proof of statement 4.* We first prove that

$$S_1 := \left\{ x \in S : \forall k \in W, a_k^\mathsf{T} x < b_k \right\} \subseteq S^\circ.$$

Pick $x \in S_1$. By the continuity of $a_k^\mathsf{T} x$, there exists a small neighborhood $U$ of $x$ such that $U \cap S \subseteq S_1$. Therefore, $x$ is an interior point of $S$. Since $x$ is arbitrary, $S_1 \subseteq S^\circ$ holds.

Second, we show that

$$S_2 := \left\{ x \in S : \exists k \in W, a_k^\mathsf{T} x = b_k \right\} \subseteq \partial S.$$

Note that by the construction of statement 3, all constraints indexed in $W$ in $S$ can not be deleted. It then guarantees that for every $k \in W$, there exists $x_0 \in \mathbb{R}^n \setminus S$ such that $a_k^\mathsf{T} x_0 > b_k$ and $a_j^\mathsf{T} x_0 \leq b_j$ holds for each $j \in W \setminus \{k\}$. Suppose $x_1 \in S_2$. Then $a_k^\mathsf{T} x_1 = b_k$ for some $k \in W$. Consider the direction $d = x_0 - x_1$. Notice that the set $\left\{x \in \mathbb{R}^n : a_k^\mathsf{T} x \geq b_k \text{ and } \forall j \in W \setminus \{k\}, a_j^\mathsf{T} x \leq b_j\right\}$ is convex. Since $x_1, x_0$ are both in it, the line segment defined by $x_1, x_0$ also lies in it. Therefore, for arbitrarily small $\epsilon > 0$,

$$a_k^\mathsf{T}\left(x_1 + \epsilon(x_0 - x_1)\right) = (1-\epsilon)\underbrace{a_k^\mathsf{T} x_1}_{=b_k} + \epsilon \underbrace{a_k^\mathsf{T} x_0}_{>b_k} > b_k.$$

Hence $x_1 + \epsilon(x_0 - x_1) \notin S$ and thus $x_1 \in \partial S$. Since $x_1$ is arbitrary, $S_2 \subset \partial S$ holds.

Now we combine these two results. Clearly $S_1 \cap S_2 = \varnothing$ and $S_1 \cup S_2 = S = \partial S \cup S^\circ$. So we must have $S_1 = S^\circ$ and $S_2 = \partial S$. □

Finally, statement 6 is the direct corollary of a result on face lattice.

**Definition Appendix A.1.** *A* graded lattice *is a finite partially ordered set* $(S, \leq)$ *if it shares* all *the following properties.*

- *It has a unique minimal element* $\hat{0}$ *and a unique maximal element* $\hat{1}$.
- *Every maximal chain has the same length.*
- *Every two elements* $x, y \in S$ *have a unique minimal upper bound in* $S$*, called the* join $x \vee y$*, and every two elements* $x, y \in S$ *have a unique maximal lower bound in* $S$*, called the* meet $x \wedge y$.

*For a graded lattice, the minimal elements of* $S \setminus \hat{0}$ *are called* atoms*, and the maximal elements of* $S \setminus \hat{1}$ *are called* coatoms.

*A lattice is* atomic *if every element is a join $x = a_1 \vee \cdots \vee a_k$ of $k \geq 0$ of atoms. Similarly, a lattice is* coatomic *if every element is a meet of coatoms.*

**Theorem Appendix A.2** (Proposition 2.3 and Theorem 2.7 in [41])**.** *Let $P$ be a convex polytope. Consider the set of all faces $L(P)$, partially ordered by inclusion.*

1. *Set $L(P)$ is a graded lattice of length $\dim(P) + 1$. The meet operation is exactly the intersection of sets.*
2. *The face lattice $L(P)$ is both atomic and coatomic.*
3. *The faces of $F$ are exactly the faces of $P$ that are contained in $F$.*

*Proof of statement 6.* By Theorem Appendix A.2, $L(S)$ is a graded lattice. Suppose $F$ is a face in $L(S)$. Then since $L(S)$ is coatomic, $F$ is the meet (i.e., intersection) of coatoms, i.e., facets. □

*Appendix A.3. Proof of Corollary 3.1*

We prove the corollary by discussing all possible cases that achieve the minimum. Since $g(x) = \max\{g_1(x), g_2(x)\}$, we partition the domain into three parts according to whether $g_1(x)$ is greater than, smaller than or equal to $g_2(x)$.

*Proof of Statement 1.* By symmetry, we only need to consider the case of

$$S_1 := \left\{x \in S : g_1(x) > g_2(x), \exists \lambda \geq 0, \nabla g_1(x) + \lambda^\intercal U = 0 \text{ and } \forall i \in [m], \lambda_i(U_i x - V_i) = 0\right\}.$$

It suffices to show that for every $x \in S_1$, either $\nabla g_1(x) = 0$ or $x \in \partial S$.

For a given $x \in S_1$, if $\nabla g_1(x) = 0$, then $\lambda = 0$ is a solution for the KKT conditions given in Theorem 12.1 in [40]. Otherwise, since $\nabla g_1(x) =$

$-\lambda^\mathsf{T} U \neq 0$, there must exist $i$ such that $\lambda_i \neq 0$. Therefore, we must have $U_i x = V_i$. By definition, every $x \in S$ satisfies $U_i x \leq V_i$. By our assumption on $U$, $U_i \neq 0$. So there is a vector $d \in \mathbb{R}^n$ such that $U_i d > 0$. For any $\epsilon > 0$, $U_i(x + \epsilon d) > V_i$. Hence $x + \epsilon d \notin S$, i.e., $x \in \partial S$. □

To prove the rest statements, we need the following claim.

**Claim Appendix A.1.** *For any face $T$ of $S$, if $T$ is parallel to $e_i$, we have: for any $x \in T \cap S_1$, if the minimum of $f$ can be obtained at $x$, then either $x$ is contained in a facet not parallel to $e_i$ or $\partial g_1(x)/\partial x_i = 0$.*

*Proof.* Since $x \in S_1$, by the KKT condition given in $S_1$, there exists $\lambda$ such that $\nabla g_1(x) = -\lambda^\mathsf{T} U$. Thus $\partial g_1(x)/\partial x_i = \nabla g_1(x)^\mathsf{T} e_i = -\lambda^\mathsf{T} U e_i$. Note that by the definition of parallel, there exists a line $l$ such that $l \subseteq \text{aff}(T)$ and $l \parallel e_i$. Therefore, for any $x \in T$, there exists a line $l_x \subseteq \text{aff}(T)$ such that $x \in l_x$ and $l_x \parallel e_i$. Define $\tilde{l}_x := l_x \cap T$.

If $x$ is not an endpoint of the line segment $\tilde{l}_x$, then $\pm e_i$ are both feasible directions for $x$, namely for any sufficiently small $\epsilon > 0$, $x \pm \epsilon e_i \in T$. We show that in this case, $x$ must satisfy $\partial g_1(x)/\partial x_i = 0$. The KKT condition implies that for any $j \in [n]$, we either have $\lambda_j = 0$ or $U_j x = V_j$. If $U_j x = V_j$, since $x \pm \epsilon e_i \in S$, $U_j(x \pm \epsilon e_i) \leq V_j = U_j x$, which means $U_j e_i = 0$. Therefore, either $\lambda_j = 0$ or $U_j e_i = 0$, we have $\lambda^\mathsf{T} U e_i = \sum_j \lambda_j U_j e_i = 0$. Thus $\partial g_1(x)/\partial x_i = -\lambda^\mathsf{T} U e_i = 0$ as desired.

To finish our proof, it suffices to show that if $x \in S_1 \cap T$ is an endpoint of $\tilde{l}_x$, then either $x$ is contained in a facet $N$ of $S$ not parallel to $e_i$ or $\partial g_1(x)/\partial x_i = 0$. We prove it by induction on $\dim(T)$.

Suppose $\dim(T) = 1$. By the definition of polytopes, $T$ must be a one-

dimensional bounded and closed convex set, i.e., a line segment. In this case, $e_i \parallel T$, so for any $x_0 \in T$, $\tilde{l}_{x_0}$ is exactly $T$. Thus, $x_0$ is the endpoint of $\tilde{l}_{x_0}$ if and only if $\{x_0\}$ is a face of $T$. By Theorem Appendix A.2, $\{x_0\}$ is a face of $S$. Then by statement 6 in Proposition 3.1, $\{x_0\}$ is the intersection of several facets of $S$. Since $S = \left\{x \in \mathbb{R}^n : U_i^\mathsf{T} x \leq V_i, i \in [m]\right\}$, by statement 5 in Proposition 3.1, the facets of $S$ can be expressed as $S_i' = \left\{x \in S : U_i^\mathsf{T} x = V_i\right\}$, $i \in W$ for some index subset $W$. Thus there exists a nonempty subset of $W$, denoted by $I$, such that $\{x_0\} = \bigcap_{i\in I} S_i'$ and $x_0 \notin S_j'$ for every $j \in W \setminus I$. Note that by assumption $\dim(S) = n$, we have $\mathrm{aff}(S) = \mathbb{R}^n$. By statement 3 in Proposition 3.1, $S = \left\{x \in \mathbb{R}^n : U_i^\mathsf{T} x \leq V_i, \forall i \in W\right\}$. Then we have

$$\begin{aligned}
\{x_0\} =& \bigcap_{i\in I} S_i' \\
=& \bigcap_{i\in I} \left(S \cap \left\{x \in \mathbb{R}^n : U_i^\mathsf{T} x = V_i\right\}\right) \\
=& S \cap \bigcap_{i\in I} \left(\left\{x \in \mathbb{R}^n : U_i^\mathsf{T} x = V_i\right\}\right) \\
=& \left\{x \in \mathbb{R}^n : U_i^\mathsf{T} x = V_i, i \in I, U_j^\mathsf{T} x < V_j, j \in W \setminus I\right\}.
\end{aligned}$$

Now we show that there exists a facet $S_i'$, $i \in I$ not parallel to $e_i$. Suppose on the contrary that for any $i \in I$, $S_i'$ is parallel to $e_i$, then we have $U_i^\mathsf{T} e_i = 0$ by the definition of parallel. Thus for any $k \in \mathbb{R}$, we have $U_i^\mathsf{T}(x_0 + ke_i) = V_i$ for every $i \in I$. Note that by continuity there exists a sufficiently small $\epsilon > 0$ such that for every $j \in W \setminus I$, $U_j^\mathsf{T}(x + \epsilon e_i) < V_j$. Thus $x_0 + \epsilon e_i$ is also contained in the set $\left\{x \in \mathbb{R}^n : U_i^\mathsf{T} x = V_i, i \in I, U_j^\mathsf{T} x < V_j, j \in W \setminus I\right\} = \{x_0\}$, a contradiction. So we finish the proof of the case $\dim(T) = 1$.

Now we suppose that the result holds on every $h$-dimensional face with $h = m - 1 \leq n - 1$, and let $\dim(T) = m$. Note that for any $x \in T^\circ$, there

exists $\epsilon > 0$ such that every $d \parallel \text{aff}(T)$ satisfies $x + \epsilon d \in T$. So $x$ must be an interior point of $\tilde{l}_x$ and thus not be an endpoint of $\tilde{l}_x$. We have assumed that $x$ is the endpoint of $\tilde{l}_x$, so this is not the case. We must have $x \in \partial T$. By statement 4 and 5 in Proposition 3.1, $x$ must be contained in a face $T' \subseteq \partial T$ of $T$ with $\dim(T') = m - 1$. By Theorem Appendix A.2, $T'$ is also a face of $S$. If $T' \parallel e_i$, then line $l_x \subseteq \text{aff}(T')$ with $x \in l_x$. Let $\tilde{l}'_x = l_x \cap T'$. By the same argument, $\tilde{l}'_x$ is a line segment. If $x$ is not an endpoint of this segment, then we have proved that $\partial g_1(x)/\partial x_i = 0$ as desired. If $x$ is an endpoint, then by the induction hypothesis either $x$ is contained in a facet $N$ of $S$ not parallel to $e_i$ or $\partial g_1(x)/\partial x_i = 0$. Thus the induction holds. If face $T'$ is not parallel to $e_i$, then we show that $T'$ is contained in some facet $N$ of $S$ not parallel to $e_i$. Note that since face $T'$ is not parallel to $e_i$, $l_x \cap T' = \{x\}$. If all facets $S'$ of $S$ containing $T'$ are parallel to $e_i$, by the same argument on the case of $\dim(T) = 1$, for sufficiently small $\epsilon > 0$, $x + \epsilon e_i \in T'$. However, clearly $x + \epsilon e_i \in l_x$ and thus $x + \epsilon e_i \in l_x \cap T'$, which leads to a contradiction. □

Now we can continue the main proof.

*Proof of Statement 2.* For statement 2, it suffices to show that for every $x \in \partial S \cap S_1$, either $x \in \partial S_N$ or both $x \in \partial S_P$ and $\partial g_1(x)/\partial x_i = 0$ hold. Equivalently, we show that for every $x \in P \cap S_1$, where $P$ is a facet parallel to $e_i$, either there exists a facet $N$ which is not parallel to $e_i$ such that $x \in N$, or $\partial g_1(x)/\partial x_i = 0$. This immediately follows by taking $m = n - 1$ in Claim Appendix A.1. □

*Proof of Statement 3.* We first show that any face $T$ of $S$ must have the form

$$T_{I_1,I_2} := \prod_{i\in[n]} S_i.$$

Here, $S_i = \{m_i\}$ for every $i \in I_1$, $S_i = \{M_i\}$ for every $i \in I_2$ and $S_i = [m_i, M_i]$ for every $i \in [n] \setminus (I_1 \cup I_2)$, where $I_1, I_2 \subseteq [n]$ and $I_1 \cap I_2 = \varnothing$.

By applying statement 3 in Proposition 3.1, $S$ can be written into $\{x \in \mathbb{R}^n : x_i \leq M_i, -x_i \leq -m_i, \forall i \in [n]\}$. Therefore, by statement 5 in Proposition 3.1, the facets of $S$ is given by $S \cap \{x \in \mathbb{R}^n : x_i = m_i\}$ or $S \cap \{x \in \mathbb{R}^n : x_i = M_i\}$. By statement 6 in Proposition 3.1, any face $T$ of $S$ can be expressed as the intersection of several facets. Thus there exist index subsets $I_1, I_2$ such that

$$T = S \cap \bigcap_{i\in I_1} \{x \in \mathbb{R}^n : x_i = m_i\} \cap \bigcap_{i\in I_2} \{x \in \mathbb{R}^n : x_i = M_i\}.$$

If $I_1 \cap I_2 = \varnothing$, then exactly $T = T_{I_1,I_2}$; otherwise, if $i \in I_1 \cap I_2$, by $m_i \neq M_i$, $T = \varnothing$. Now, it suffices to show that any $T_{I_1,I_2}$ is a face of $S$. For any $T_{I_1,I_2}$, consider $a = -\sum_{i\in I_1} e_i + \sum_{i\in I_2} e_i, b = -\sum_{i\in I_1} m_i + \sum_{i\in I_2} M_i$. On the one hand, for any $x \in S$, we have $m_i \leq x^\mathsf{T} e_i = x_i \leq M_i$, so $a^\mathsf{T} x = \sum_{i\in I_1}(-x_i) + \sum_{i\in I_2} x_i \leq -\sum_{i\in I_1} m_i + \sum_{i\in I_2} M_i = b$. On the other hand, we can see that the equality holds if and only if $x_i = m_i$ for every $i \in I_1$ and $x_i = M_i$ for every $i \in I_2$. This set is exactly given by $T_{I_1,I_2}$, so $T_{I_1,I_2}$ is the face of $S$ determined by $a, b$, and we finish our proof.

With the clear description of all the faces of $S$ by $T_{I_1,I_2}$, we can easily see that a face $T_{I_1,I_2}$ is a single point if and only if $I_1 \cup I_2 = [n]$. Also, for any face $T$ that is not a single point, there exists $i \in [n]$ such that $i \notin I_1 \cup I_2$. So, for any $y = (y_1, ..., y_n)$ in $T$, $\prod_{j\neq i}\{y_j\} \times [m_i, M_i] \subseteq T$, which defines a line from

$(y_1, \ldots, y_{i-1}, m_i, y_{i+1}, \ldots, y_n)$ to $(y_1, \ldots, y_{i-1}, M_i, y_{i+1}, \ldots, y_n)$ parallel to $e_i$. Therefore, by the definition of parallel, $e_i \parallel T$. So every face $T$ of $S$ is parallel to some $e_i$.

Suppose $x \in \partial S$. We show that either $x$ is a single point face, or $x$ belongs to the interior of some face $T$ parallel to some $e_i$. Note that $x$ must belong to some face $T$ of $S$. We prove it by induction on the dimension of $T$. When $\dim(T) = 0$, $T = \{x\}$ is a single point face. For $\dim(T) = n$, we only need to consider the case that $x \notin T^\circ$. Immediately, $x \in \partial T$, so by Proposition 3.1 and Theorem Appendix A.2, $x$ belongs to some face of $S$ with lower dimension. The result then follows by the induction hypothesis.

Thus, either $x$ is a single point face given by $\{x \in \mathbb{R}^n : \forall i, x_i \in \{m_i, M_i\}\}$, or $x$ belongs to the interior of a face $T$ parallel to some $e_i$. We now apply the result in Claim Appendix A.1. Suppose $x$ is not a single point face that attains the minimum of $f$. If $g_1(x) > g_2(x)$, then $\partial g_1(x)/\partial x_i = 0$; if $g_1(x) < g_2(x)$, then $\partial g_2(x)/\partial x_i = 0$. So, $x$ must be contained in the set $S^+$ given in this statement. □

*Appendix A.4. The $(2, m)$-Separation Algorithm*

This problem can be restated as a famous problem in computational geometry called *envelope problem*, which is a special case of *half-plane intersection problem*. The half-plane intersection problem can be solved with the plane sweep method in time $O(n \log n)$ with $n$ breakpoints, see, e.g., Section 4.2 of [33]. For completeness, we restate the full algorithm here.

Specifically, suppose we are given two series $\{a_i\}_{i=1}^k, \{b_i\}_{i=1}^k$. We want to compute the breakpoints of function $h(x) = \max_{i \in [k]}\{a_i x + b_i\}$ in the interval $[0, 1]$ and the value of $h$ on these points. We present a method based on ideas

from computational geometry.

First, we turn the case into $a_1 < a_2 < ... < a_k$. To do so, reorder functions $\{a_i x + b_i\}_i$ so that $a_1 \leq a_2 \leq ... \leq a_k$. Then we check all contiguous pairs $(a_i, a_{i+1})$. If $a_i = a_{i+1}$, then we delete the function with smaller $b_i$, since it is strictly smaller that the other one. By this procedure, we obtain $a_1 < a_2 < ... < a_k$ in time $O(k \log k)$.

Let us use a list $w$ to memorize the breakpoints and a list $t$ to memorize the value of $h(x)$ on these points. Define $h_s(x) = \max_{i \in [s]}\{a_i x + b_i\}$. We use a recursion method to find the breakpoints by gradually updating the set of breakpoints from $h_1(x)$ to $h_k(x) = h(x)$. For the beginning, since $h_1(x) = a_1 x + b_1$, we can initialize list $w$ with $w(0) = 0, w(1) = 1$ and list $t$ with $t(0) = b_1, t(1) = a_1 + b_1$. Then we consider how to update from $h_s(x)$ to $h_{s+1}(x)$.

By assumption $a_1 < a_2 < ... < a_s < a_{s+1}$, function $\Delta h_s(x) = h_s(x) - a_{s+1}x - b_{s+1}$ is continuous on $[0, 1]$ and decreasing on every linear piece. So $\Delta h_s$ is decreasing on $[0, 1]$ and has at most one zero point. Therefore, $h_s(x)$ has at most one intersection point with $a_{s+1}x + b_{s+1}$. If such point exists, say $x^*$, then we have $h_s(x) \leq a_{s+1}x + b_{s+1}$ if and only if $x \geq x^*$. So, we only need to add $x^*$ into list $w$ and delete all the points in list $w$ which belong to $[x^*, 1)$. Similarly, we add $a_{s+1}x^* + b_{s+1}$ into the list $t$, update the value corresponding to 1 with $a_{s+1} + b_{s+1}$ and delete all values between them. The geometric illustration of such a procedure is given in Figure A.4.

To find such $x^*$, we use a binary search on index $t$ to locate the proper line $a_t x + b_t$ forming the intersection point $x^*$. Such a search costs only logarithm time of the number of lines.

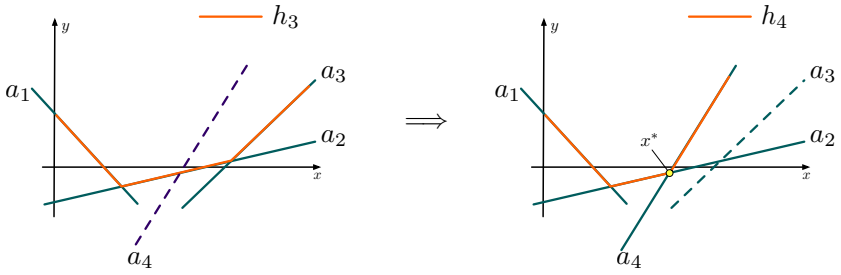


Figure A.4: Illustration of the update procedure from $h_3$ (orange line on the left) to $h_4$ (orange line on the right). We try to add term $a_3x + b_3$ (dashed line on the left) into the max operator in $h_3$. $a_4x + b_4$ intersects $h_3$ at $a_2x + b_2$. Thus $x^*$ (yellow dot) is calculated and the breakpoints larger than $x^*$ are deleted. Equivalently, $a_3x + b_3$ is removed (dashed line on the right).

Now we analyze the time complexity of this algorithm. There are in total $k$ rounds of binary searches, with the $i$th round using time $O(\log i)$. In total, the time complexity is $O\left(\sum_{i=1}^{k} \log i\right) = O(k \log k)$. We collect the above arguments into the following proposition.

**Proposition Appendix A.1.** *There exists an algorithm that outputs all the breakpoints of $h(x)$ and their corresponding function values in time $O(k \log k)$.*

*Appendix A.5. The $(3, m)$-Separation Algorithm*

Note that since $t = 3$, $\beta$ can be represented by two free variables, i.e., $\beta = (x, y, 1 - x - y)$. Then the polytope $P_i$ in Definition 3.2 is actually a polygon on the plane. The $(3, m)$-separation algorithm needs to find a clockwise enumeration of vertices of $P_i$. This problem, again, can be stated by the half-plane intersection. For completeness, we present the algorithm here.

In fact, a proper application of the $(2, m)$-separation algorithm will give us

the desired algorithm. A key observation is that the boundary of a polygon can be expressed as a union of four parts: left boundary, right boundary, upper semi-boundary and lower semi-boundary. If we write all constraints of $P_i$ in the form $l_j := \tilde{a}_j x + \tilde{b}_j y + \tilde{c}_j \geq 0, j \in [k]$, then each constraint belongs to exactly one part of the boundary:

1. When $\tilde{b}_j = 0$, $l_j = 0$ is a candidate of the left (right) boundary if $\tilde{a}_j > 0$ ($< 0$).

2. When $\tilde{b}_j \neq 0$, $l_j = 0$ is a candidate of the upper (lower) semi-boundary if $\tilde{b}_j < 0$ ($> 0$).

In the second case, we write the boundary into the form $y = \tilde{a}x + \tilde{b}$. Then we can apply Proposition Appendix A.1 on the upper (lower) semi-boundary to obtain ordered vertices in time $O(k \log k)$. Next, we combine the two semi-boundaries to obtain the leftmost and rightmost vertices.

Now we determine the vertical boundaries. The left (right) boundary, if exists, has the maximum (minimum) $-\tilde{c}_j/\tilde{a}_j$, which can be found in $O(k)$ time. If the left (right) boundary does not rule out the leftmost (rightmost) vertex, then there is no left (right) boundary. Otherwise, by a binary search on vertices of the two semi-boundaries, we can find two segments adjacent to the left (right) boundary in $O(\log k)$ time. An illustration of this procedure is presented in Figure A.5.

We collect the above arguments into the following proposition.

**Proposition Appendix A.2.** *There exists an algorithm that outputs all vertices in a clockwise order of the polygon $P_i$ in time $O(k \log k)$.*

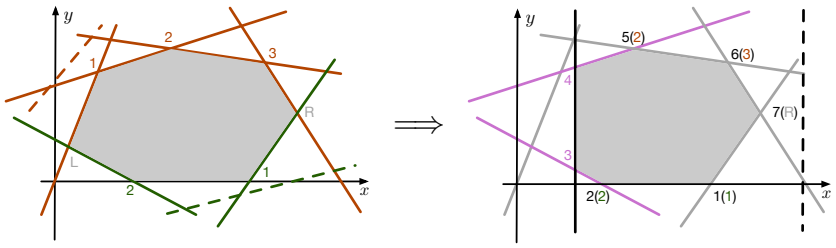


Figure A.5: Illustration of the procedure computing ordered vertices (vertical line cases). On the left, the lower semi-boundary and the upper one is colored with green and orange, respectively. The vertices are labeled clockwise on each semi-boundary. On the right, we try to add vertical lines. The black dashed line does not change the structure at all, so it is omitted. The black solid line will change the structure, and the new labels of vertices are computed (and in the bracket is the original labels).

*Appendix A.6. The Optimal* $(2,2)$*-Mixing Algorithm*

We first give some notations Let $F_R(\alpha,\beta) = f_R(\alpha x_1+(1-\alpha)x_2, \beta y_1+(1-\beta)y_2)$. Define $F_C(\alpha,\beta)$ similarly. Then let $F(\alpha,\beta) = \max\{F_R(\alpha,\beta), F_C(\alpha,\beta)\}$. The goal of the optimal $(2,2)$-mixing algorithm is to calculate the minimum of $F$ on square $\mathcal{A} = [0,1]\times[0,1]$. Now we state the algorithm.

Applying the $(2,m)$-separation algorithm in Appendix A.4, we can construct a mesh grid of $(\alpha,\beta)$ so that on each grid, both $F_R$ and $F_C$ are linear in $\alpha$ and $\beta$ respectively. Then both $F_R$ and $F_C$ have the form $x_1 + x_2\alpha + x_3\beta + x_4\alpha\beta$, where $x_i$'s are constants determined by $F_R$ or $F_C$ values on four vertices of the grid. Our next step is then to give a method computing the minimum point of $F(\alpha,\beta)$ on each grid.

On each grid, by statement 3 in Corollary 3.1, it suffices to minimize $F = \max\{F_R, F_C\}$ over:

1. points with $\partial F_k(\alpha,\beta)/\partial\alpha = 0$ or $\partial F_k(\alpha,\beta)/\partial\beta = 0$, $k \in \{R, C\}$,

2. the four vertices of the grid, and

3. points with $F_R = F_C$.

- (Case 1) Equations $\partial F_k(\alpha, \beta)/\partial\alpha = 0$ and $\partial F_k(\alpha, \beta)/\partial\beta = 0$ have the form that $\alpha$ or $\beta$ takes a fixed value.[4] Hence the problem becomes computing the minimum of two univariate linear functions, which is easy to solve.

- (Case 2) We just need to enumerate the value of $F$ on the four vertices.

- (Case 3) By solving the equation $F_R(\alpha, \beta) = F_C(\alpha, \beta)$, we obtain an expression of $\beta$ given by a linear fraction of $\alpha$. If the denominator linear function of $\alpha$ is zero, then we can solve it just like in case 1. Otherwise, by substituting the expression of $\beta$ into the expression of $F = F_R$, we convert this problem into finding the minimum of a function $g(\alpha)$ with the form $(a_2\alpha^2+a_1\alpha+a_0)/(b_1\alpha+b_0)$. This can be done by calculating its values at two boundary points and points with zero derivatives. Note that $g'(\alpha) = 0$ is equivalent to a quadratic equation in $\alpha$, which has at most two solutions. So in this case we can test at most four points to find the minimum.

We collect the above arguments into the following proposition:

**Proposition Appendix A.3.** *There exists an algorithm finding the minimum point of $F(\alpha, \beta)$ on any grid in $O(1)$ time.*

[4]When the coefficient of $\alpha$ (or $\beta$) is zero, all or none of $\alpha$ (or $\beta$) solve the equation.

With these results above, we can efficiently calculate the minimum point of $f$ on each grid where both $F_R$ and $F_C$ are linear in $\alpha$ and $\beta$ respectively. Note that the numbers of breakpoints of $\alpha$ and $\beta$ are at most $m$ and $n$ respectively, so there are at most $mn$ grids. On each grid the minimization procedure takes $O(1)$ time, implying a total $O(mn)$ time on $\mathcal{A}$. Thus time complexity of calculating the minimum of $f$ on $\mathcal{A}$ is $O(\max\{m,n\}\log\max\{m,n\}) + O(mn) = O(mn)$. We summarize it as the following theorem.

**Theorem Appendix A.3.** *Give any strategies $x_1, x_2 \in \Delta_m$ and $y_1, y_2 \in \Delta_n$, let*

$$F(\alpha,\beta) = f(\alpha x_1 + (1-\alpha)x_2, \beta y_1 + (1-\beta)y_2), \quad \alpha,\beta \in [0,1].$$

*Then there exists an algorithm finding the minimum point of $F(\alpha,\beta)$ in time $O(mn)$.*

*Appendix A.7. The Optimal $(2,3)$-Mixing Algorithm*

We begin by some notations. Let

$$\begin{aligned} F_R(\alpha,\beta,\gamma) =& \max\{R(\gamma y_1 + (1-\gamma)y_2)\} - \\ & (\alpha x_1 + \beta x_2 + (1-\alpha-\beta)x_3)^{\mathsf{T}} R(\gamma y_1 + (1-\gamma)y_2), \\ F_C(\alpha,\beta,\gamma) =& \max\left\{C^{\mathsf{T}}(\alpha x_1 + \beta x_2 + (1-\alpha-\beta)x_3)\right\} - \\ & (\alpha x_1 + \beta x_2 + (1-\alpha-\beta)x_3)^{\mathsf{T}} C(\gamma y_1 + (1-\gamma)y_2). \end{aligned}$$

Define $F(\alpha,\beta,\gamma) = \max\{F_R(\alpha,\beta,\gamma), F_C(\alpha,\beta,\gamma)\}$. Then the algorithm in this part minimizes $F$ on the prism $\mathcal{A} = \{(\alpha,\beta,\gamma) \in [0,1]^3 : \alpha+\beta \le 1\}$.

Using $(2,m)$-separation algorithm in Appendix A.4 and $(2,m)$-separation

algorithm in Appendix A.7, we can obtain the linear region[5] of function $F$. Our next step is then to minimize $F$ on each linear region, in which both $F_R$ and $F_C$ have form $a_0\alpha\gamma + a_1\beta\gamma + a_2\gamma + a_3\alpha + a_4\beta + a_5$. Note that every linear region $S$ is given by the Cartesian product of a polygon $P$ and an interval $I$. Thus by statement 2 in Corollary 3.1, the minimum of $F$ must be obtained when:

1. $(\alpha, \beta)$ belongs to side surfaces of $S$ and
   (a) either there exists $k \in \{R, C\}$ such that $\partial F_k/\partial\gamma = 0$, or
   (b) $(\alpha, \beta)$ is in the intersection of side surfaces and top/bottom surfaces.
2. $(\alpha, \beta)$ belongs to top/bottom surfaces of $S$ and
   (a) there exists $k \in \{R, C\}$ such that either $\partial F_k/\partial\alpha = 0$ or $\partial F_k/\partial\beta = 0$, or
   (b) $(\alpha, \beta)$ is in the intersection of side surfaces and top/bottom surfaces.
3. $F_R(\alpha, \beta) = F_C(\alpha, \beta)$.
4. $\nabla F_R(\alpha, \beta) = 0$ or $\nabla F_C(\alpha, \beta) = 0$.

For case 1b and 2b, note that the boundary is formed by $O(m)$ line segments. Furthermore, $F_R$ and $F_C$ are linear on each segment. Thus it suffices to check their intersection and two endpoints on each segment.

[5]To shorten statements, we say region $X$ is a *linear region* of function $F$ if $F$ is linear in every variable on $X$.

For case 2a and case 4, the equation of zero derivative gives a linear equation on $\gamma$. Then $\gamma$ takes a fixed value. Now we have to minimize $F$ over the polygon $P$ of $(\alpha, \beta)$. We can use statement 2 in Corollary 3.1 again, and minimize $F$ at points on the boundary of $P$, with $F_R = F_C$, and with zero partial derivative in $\alpha$ or $\beta$. The case of the boundary is similar to case 1b and 2b, consuming time $O(m)$. The rest cases are similar to discussions in Proposition Appendix A.3: We can turn the problem into minimizing a univariate function $g$. The only difference here is the domain $J$ of $g$. By the same calculation in the proof of Proposition Appendix A.3, it can be shown that $J$ is a segment (or line) $\tilde{J}$ parallel or perpendicular to $\alpha = 0$ when we ignore the restriction of $P$. Domain $J$ then can be determined by searching the intersection points of $\tilde{J}$ and the boundary of $P$ in $O(m)$ time.

For case 1a, the equation of zero derivative gives a linear equation on $(\alpha, \beta)$. Then the equation produces a line $l$ on $(\alpha, \beta)$. Now, the feasible set of $(\alpha, \beta)$ is a segment $L$ determined by the intersection of $l$ and $P$. Similar to $J$, we can compute two endpoints of $L$ in $O(m)$ time. Note that by a suitable linear transformation from $(\alpha, \beta)$ to $(\alpha', \beta')$, the equation of $l$ becomes $\alpha' = 0$. Then on $L \times I$, $F$ becomes a function of $(\beta', \gamma)$ being linear in $\beta'$ and $\gamma$, respectively. Now we can apply Proposition Appendix A.3 to minimize $F$ on $L \times I$ in $O(1)$ time.

For case 3, by $F_R = F_C$, we obtain an expression of $\gamma$ given by the fraction of linear functions in $\alpha$ and $\beta$. A special case is that the denominator equals zero. We can deal with this case in the same way as case 1a. Otherwise, by substituting this expression into $F_R$, it suffices to minimize a function $h(\alpha, \beta)$ with the form $(c_0 + c_1\alpha + c_2\beta + c_3\alpha^2 + c_4\alpha\beta + c_5\beta^2)/(d_0 + d_1\alpha + d_2\beta)$ on a

given linear region.

When $d_1 = d_2 = 0$, this is to solve quadratic programming on a polygon with $O(m)$ sides. The minimum is taken either on the sides or at interior points with zero derivatives. Since it has only two variables, we can cancel one of the variables via the linear equation of a side. Then the minimization on the side is equivalent to minimizing a univariate quadratic function on a segment. On the other hand, the zero-derivative condition is exactly two linear equations with two variables. In both situations, the calculation can be completed within $O(m)$ time.

Otherwise, we substitute the denominator with $\theta$, and the expression is transformed into

$$G_1(\alpha, \theta) := \frac{e_0 + e_1\alpha + e_2\alpha^2}{\theta} + e_3 + e_4\alpha + e_5\theta.$$

First, we consider the minimum about $\theta$. By the property of hyperbolic function, the minimum can only be obtained at the boundary points or at $\theta = \pm\sqrt{(e_0 + e_1\alpha + e_2\alpha^2)/e_5}$ (if exists). Since $\theta$ is linear in $(\alpha, \beta)$ and the domain of $(\alpha, \beta)$ is a polygon $P$, the domain of $\theta$ is a interval given by $[M_{\min}(\alpha), M_{\max}(\alpha)]$, where $M_{\min}, M_{\max}$ are piecewise linear functions with $O(m)$ pieces. By considering vertices of $P$ in order, we can calculate linear pieces of $M_{\min}$ and $M_{\max}$ in $O(m)$ time, denoted by $I_i^{\min}$ and $I_j^{\max}$, respectively. So we only need to consider $O(m)$ cases that $\theta$ takes $M_{\min}(\alpha)$ on $\alpha \in I_i^{\min}$, $M_{\max}(\alpha)$ on $\alpha \in I_j^{\max}$, or $\pm\sqrt{(e_0 + e_1\alpha + e_2\alpha^2)/e_5}$ when $(e_0 + e_1\alpha + e_2\alpha^2)\, e_5 \geq 0$. In each case, it suffices to find the minimum of either

$$\frac{e_0 + e_1\alpha + e_2\alpha^2}{t(\alpha)} + e_3 + e_4\alpha + e_5 t(\alpha), t \in \{M_{\min}, M_{\max}\} \text{ or}$$

$$e_3 + e_4\alpha \pm 2\sqrt{e_5\left(e_0 + e_1\alpha + e_2\alpha^2\right)},$$

where $\alpha$ belongs to a certain interval. Each case can be solved by calculating points on the boundary and points with zero derivatives in $O(m)$ time.

We conclude the discussion above with the following proposition.

**Proposition Appendix A.4.** *There exists an algorithm finding the minimum point of $F(\alpha, \beta, \gamma)$ on linear region $S = P \times I$ in $O(m)$ time, where $P$ is a polygon of $(\alpha, \beta)$ formed by $O(m)$ linear constraints and $I$ is a closed interval of $\gamma$.*

Now we come back to function $F(\alpha, \beta, \gamma)$. Using Proposition Appendix A.1 and Proposition Appendix A.2, we can split the domain of $F$ into $O(mn)$ linear regions in time $O(n \log n + m^2 \log m)$. Then on each region, we can use Proposition Appendix A.4 to compute the minimum value of $F$ in time $O(m)$. The total time complexity is then $O\left(m^2(n + \log m) + n \log n\right)$. We summarize it into the following theorem.

**Theorem Appendix A.4.** *Give any $x_1, x_2, x_3 \in \Delta_m$ and $y_1, y_2 \in \Delta_n$, let*

$$F(\alpha, \beta, \gamma) = f\left(\alpha x_1 + \beta x_2 + (1 - \alpha - \beta)x_3, \gamma y_1 + (1 - \gamma)y_2\right),$$

*where $\alpha, \beta, \gamma, \alpha + \beta \in [0, 1]$. Then there exists an algorithm finding the minimum point of $F(\alpha, \beta, \gamma)$ in time $O\left(m^2(n + \log m) + n \log n\right)$.*

*Appendix A.8. The Approximate Optimal Algorithms and Proof of Theorem 4.1*

In this part, we present the full algorithm for the $\epsilon$-optimal $(s, t)$-mixing problem and prove Theorem 4.1. The algorithm is shown in Algorithm 4.

## Appendix B. Using Optimal Mixing Algorithms as an Assembling Tool for Approximate NE

Suppose we have a finite set of algorithms $\mathcal{A} = \{A_1, \ldots, A_k\}$ for computing approximate Nash equilibria in bimatrix games. Then, we can use (approximate) optimal mixing algorithms to assemble the outputs of these algorithms to obtain a new algorithm as Algorithm 6.

**Algorithm 6** Assembled Approximate NE Algorithm

**Input:** A bimatrix game $(R, C)$.

**Output:** An approximate Nash equilibrium $(x^*, y^*)$.

1: Run each algorithm $A_i \in \mathcal{A}$ on $(R, C)$ to obtain an approximate NE $(x_i, y_i)$.

2: Solve the optimal mixing problem with the set of strategies $\{x_1, \ldots, x_k\}$ and $\{y_1, \ldots, y_k\}$, resulting in an strategy profile $(x^*, y^*)$.

3: **return** $(x^*, y^*)$.

To illustrate the effectiveness of this assembling tool in practice, we conduct some small experiments.

1. **Payoff Matrices:** We generated two random $3 \times 3$ matrices, $R$ and $C$, where each entry is drawn from a uniform distribution over the interval $[0, 1]$.

2. **Strategies:** For each experiment, we randomly generated three pure strategy vectors $x_1, x_2, x_3$ and $y_1, y_2, y_3$ for the two players, each of dimension 3.

3. **Optimization Method:** We used the Sequential Least Squares Programming (SLSQP) algorithm to solve the constrained optimization

problem. An initial guess was set to a uniform distribution, $\alpha_i = \beta_i = \frac{1}{3}$ for all $i$. The solution provides nearly optimal weights $\alpha^*$ and $\beta^*$, which are used to compute the mixed strategies.

The experimental results are presented in Table B.2. We highlight the ones where the optimal mixing $f$-value is smaller than the $f$-values of all the strategies.

| Trial | Strategy 1 $f$-value | Strategy 2 $f$-value | Strategy 3 $f$-value | Optimal Mixing $f$-value |
|---|---|---|---|---|
| 1 | 0.127 | 0.095 | 0.284 | **0.095** |
| 2 | 0.187 | 0.198 | 0.213 | **0.086** |
| 3 | 0.118 | 0.144 | 0.205 | **0.087** |
| 4 | 0.256 | 0.121 | 0.220 | **0.085** |
| 5 | 0.139 | 0.174 | 0.136 | **0.111** |
| 6 | 0.120 | 0.197 | 0.258 | **0.076** |
| 7 | 0.126 | 0.258 | 0.139 | **0.087** |
| 8 | 0.183 | 0.112 | 0.141 | **0.092** |
| 9 | 0.204 | 0.232 | 0.202 | **0.086** |
| 10 | 0.302 | 0.175 | 0.125 | **0.109** |

Table B.2: Trial results for different input strategies and optimal mixing $f$-values

As is shown in Table B.2, the optimal mixing $f$-value is smaller than the $f$-values of all the strategies in most of the trials. Although this experiments are only for illustrative purposes, it shows that the using of optimal mixing algorithms can indeed improve the approximation in practice. Further

comprehensive experiments are needed to investigate the performance of the assembled algorithm in more practical scenarios.

## Appendix C. Definitions in Discrete Geometry

Below are the definitions of several concepts in discrete geometry. The concepts below are either from [47] or basic concepts in linear algebra. We append the location of the concepts from [47] for further interests.

1. (Affine Space, P1, Section 1.1) An affine space is a displacement of a vector space. It has the form of $v + V = \{v + x : x \in V\}$, where $v$ is a vector and $V$ is a vector space. Equivalently, an affine space can be expressed as $\left\{x \in \mathbb{R}^n : u_i^\mathsf{T} x = v_i, i \in [m]\right\}$ for some $u_i \in \mathbb{R}^n \setminus \{0\}$ and $v_i \in \mathbb{R}$, $i \in [m]$. The dimension of an affine space $V + v$ is defined to be the dimension of $V$.

2. (Affine Hull, P1, Section 1.1) The affine hull of a set $S \in \mathbb{R}^n$, denoted by $\text{aff}(S)$, is the minimal affine space containing it.

3. (Dimension, P83, Section 5.2) The dimension of a set $S \in \mathbb{R}^n$, denoted by $\dim(S)$, is given by the dimension of its affine hull.

4. (Hyperplane, P3, Section 1.1) In any linear space $H$, a hyperplane is an affine subspace whose dimension is one less than that of $H$.

5. (Half-space, P3, Section 1.1) In $\mathbb{R}^n$, a half-space is the set $\{x \in \mathbb{R}^n : a^\mathsf{T} x \leq b\}$ or $\{x \in \mathbb{R}^n : a^\mathsf{T} x < b\}$, where $a$ is a nonzero vector in $\mathbb{R}^n$ and $b \in \mathbb{R}$.

6. (Polytope, P82, Section 5.2) A convex polytope $S$ is defined as a bounded set that is the intersection of finitely many half-spaces, namely $S = \{x \in \mathbb{R}^n : Ax \leq b\}$, where $A \in \mathbb{R}^{k \times n}$ has no zero rows and $b \in \mathbb{R}^k$.

7. (Face, P86, Section 5.3) A face of a polytope $S$ is defined by the set $\{x \in S : \forall y \in S, a^\mathsf{T} x \leq a^\mathsf{T} y\}$ for a certain $a \in \mathbb{R}^n$. Note that by setting $a = 0$, we have $S$ itself as a face. By definition, every face of a polytope is also a polytope.

8. (Facet, P87, Section 5.3) A facet of polytope $S$ is a face of dimension exactly $\dim(S) - 1$.

9. (Boundary) The boundary of a face $S$ is defined as the set of points $x \in S$ such that for any $\epsilon > 0$, there exists $y \in \text{aff}(S) \setminus S$ satisfying $\|y - x\| < \epsilon$. We denote the boundary of $S$ as $\partial S$. The interior of $S$, denoted as $S^\circ$, is defined as $S \setminus \partial S$.

10. (Line and segment) In $\mathbb{R}^n$, a line (segment) is a set of the form $\{y \in \mathbb{R}^n : y = td + b, t \in I\}$, where $d \in \mathbb{R}^n$ is a nonzero vector, $b \in \mathbb{R}^n$, and $I = \mathbb{R}$ ($I = [u, v]$). It represents a one-dimensional affine space. The vector $d$ is called the direction of the line.

11. (Parallel lines) Two lines (segments) are said to be parallel if their directions $d_1$ and $d_2$ satisfy $d_1 = kd_2$ for some nonzero real number $k$. The relation of being parallel is an equivalence class, and two lines are equivalent if and only if they share a proportional direction. Hence, we can also represent a line (segment) using its direction vector, which we will use directly in reference to a line below.

12. (Parallel between lines and polytopes) For an affine space $A \subseteq \mathbb{R}^n$, we say that a vector $d$ is parallel to $A$ if $A$ contains a line parallel to $d$. Equivalently, if $A = \left\{x \in \mathbb{R}^n : u_i^\mathsf{T} x = v_i, i \in [m]\right\}$, then $d \parallel A$ if and only if $u_i^\mathsf{T} d = 0$ holds

for each $i \in [m]$. For a vector $d$ and a polytope $P \subseteq \mathbb{R}^n$, we say that $d$ is parallel to $P$ if $d \parallel \text{aff}(P)$.

## References


[1] N. Megiddo, C. H. Papadimitriou, On total functions, existence theorems and computational complexity, Theoretical Computer Science 81 (2) (1991) 317–324. doi:10.1016/0304-3975(91)90200-L.

[2] X. Chen, X. Deng, S.-H. Teng, Settling the complexity of computing two-player Nash equilibria, J. ACM 56 (3) (2009) 14:1–14:57. doi:10.1145/1516512.1516516.

[3] C. Daskalakis, P. W. Goldberg, C. H. Papadimitriou, The Complexity of Computing a Nash Equilibrium, SIAM Journal on Computing 39 (1) (2009) 195–259. doi:10.1137/070699652.

[4] J. Hu, M. P. Wellman, Nash Q-learning for general-sum stochastic games, Journal of machine learning research 4 (Nov) (2003) 1039–1069.

[5] C. Jin, Q. Liu, Y. Wang, T. Yu, V-Learning–A Simple, Efficient, Decentralized Algorithm for Multiagent RL, in: ICLR 2022 Workshop on Gamification and Multiagent Solutions, 2022.

[6] Q. Liu, T. Yu, Y. Bai, C. Jin, A sharp analysis of model-based reinforcement learning with self-play, in: International Conference on Machine Learning, PMLR, 2021, pp. 7001–7010.

[7] P. Muller, S. Omidshafiei, M. Rowland, K. Tuyls, J. Pérolat, S. Liu, D. Hennes, L. Marris, M. Lanctot, E. Hughes, A Generalized Training Approach for Multiagent Learning, in: ICLR, ICLR, 2020, pp. 1–35.

[8] OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, et al., Dota 2 with Large Scale Deep Reinforcement Learning (Dec. 2019). arXiv:1912.06680.

[9] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, et al., Grandmaster level in StarCraft II using multi-agent reinforcement learning, Nature 575 (7782) (2019) 350–354. doi:10.1038/s41586-019-1724-z.

[10] A. Deligkas, M. Fasoulakis, E. Markakis, A Polynomial-Time Algorithm for 1/3-Approximate Nash Equilibria in Bimatrix Games, in: S. Chechik, G. Navarro, E. Rotenberg, G. Herman (Eds.), 30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, Vol. 244 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Berlin/Potsdam, Germany, 2022, pp. 41:1–41:14. doi:10.4230/LIPIcs.ESA.2022.41.

[11] H. Tsaknakis, P. G. Spirakis, An Optimization Approach for Approximate Nash Equilibria, in: X. Deng, F. C. Graham (Eds.), Internet and Network Economics, Third International Workshop, WINE

2007, December 12-14, Proceedings, Vol. 4858 of Lecture Notes in Computer Science, Springer, San Diego, CA, USA, 2007, pp. 42–56. doi:10.1007/978-3-540-77105-0_8.

[12] H. Li, W. Huang, Z. Duan, D. H. Mguni, K. Shao, J. Wang, X. Deng, A survey on algorithms for Nash equilibria in finite normal-form games, Computer Science Review 51 (2024) 100613. doi:10.1016/j.cosrev.2023.100613.

[13] J. Nash, Non-Cooperative Games, Annals of Mathematics 54 (2) (1951) 286–295. arXiv:1969529.

[14] C. Daskalakis, A. Mehta, C. Papadimitriou, Progress in approximate nash equilibria, in: Proceedings of the 8th ACM Conference on Electronic Commerce, ACM, San Diego California USA, 2007, pp. 355–358. doi:10.1145/1250910.1250962.

[15] H. Bosse, J. Byrka, E. Markakis, New Algorithms for Approximate Nash Equilibria in Bimatrix Games, in: X. Deng, F. C. Graham (Eds.), Internet and Network Economics, Third International Workshop, WINE 2007, December 12-14, Proceedings, Vol. 4858 of Lecture Notes in Computer Science, Springer, San Diego, CA, USA, 2007, pp. 17–29. doi:10.1007/978-3-540-77105-0_6.

[16] A. Czumaj, A. Deligkas, M. Fasoulakis, J. Fearnley, M. Jurdzinski, R. Savani, Distributed Methods for Computing Approximate Equilibria, in: Y. Cai, A. Vetta (Eds.), Web and Internet Economics - 12th International Conference, WINE 2016, December 11-14, Proceedings,

Vol. 10123 of Lecture Notes in Computer Science, Springer, Montreal, Canada, 2016, pp. 15–28. doi:10.1007/978-3-662-54110-4_2.

[17] Z. Chen, X. Deng, W. Huang, H. Li, Y. Li, On tightness of the Tsaknakis-Spirakis algorithm for approximate Nash equilibrium, in: Algorithmic Game Theory: 14th International Symposium, SAGT 2021, Aarhus, Denmark, September 21–24, 2021, Proceedings 14, Springer, 2021, pp. 97–111.

[18] J. Fearnley, T. P. Igwe, R. Savani, An Empirical Study of Finding Approximate Equilibria in Bimatrix Games, in: E. Bampis (Ed.), Experimental Algorithms, Vol. 9125, Springer International Publishing, Cham, 2015, pp. 339–351. doi:10.1007/978-3-319-20086-6_26.

[19] H. Tsaknakis, P. G. Spirakis, D. Kanoulas, Performance Evaluation of a Descent Algorithm for Bi-matrix Games, in: C. Papadimitriou, S. Zhang (Eds.), Internet and Network Economics, Vol. 5385, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 222–230. doi:10.1007/978-3-540-92185-1_29.

[20] V. Conitzer, T. Sandholm, New complexity results about Nash equilibria, Games Econ. Behav. 63 (2) (2008) 621–641. doi:10.1016/j.geb.2008.02.015.

[21] I. Gilboa, E. Zemel, Nash and correlated equilibria: Some complexity considerations, Games and Economic Behavior 1 (1) (1989) 80–93. doi:10.1016/0899-8256(89)90006-7.

[22] R. J. Lipton, E. Markakis, A. Mehta, Playing large games using simple strategies, in: Proceedings of the 4th ACM Conference on Electronic Commerce, 2003, pp. 36–41.

[23] X. Deng, D. Li, H. Li, The Search-and-Mix Paradigm in Approximate Nash Equilibrium Algorithms (Oct. 2023). arXiv:2310.08066, doi:10.48550/arXiv.2310.08066.

[24] N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani, Algorithmic Game Theory, Cambridge university press, 2007.

[25] A. Rubinstein, Settling the Complexity of Computing Approximate Two-Player Nash Equilibria, in: 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS), 2016, pp. 258–265. doi:10.1109/FOCS.2016.35.

[26] C. E. Lemke, J. T. Howson, Jr., Equilibrium Points of Bimatrix Games, Journal of the Society for Industrial and Applied Mathematics 12 (2) (1964) 413–423. doi:10.1137/0112033.

[27] C. H. Papadimitriou, On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence, J. Comput. Syst. Sci. 48 (3) (1994) 498–532. doi:10.1016/S0022-0000(05)80063-7.

[28] R. Mehta, Constant rank bimatrix games are PPAD-hard, in: Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, ACM, New York New York, 2014, pp. 545–554. doi:10.1145/2591796.2591835.

[29] P. K. Kothari, R. Mehta, Sum-of-squares meets Nash: Lower bounds for finding any equilibrium, in: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, ACM, Los Angeles CA USA, 2018, pp. 1241–1248. doi:10.1145/3188745.3188892.

[30] C. Daskalakis, A. Mehta, C. H. Papadimitriou, A Note on Approximate Nash Equilibria, in: P. G. Spirakis, M. Mavronicolas, S. C. Kontogiannis (Eds.), Internet and Network Economics, Second International Workshop, WINE 2006, December 15-17, Proceedings, Vol. 4286 of Lecture Notes in Computer Science, Springer, Patras, Greece, 2006, pp. 297–306. doi:10.1007/11944874_27.

[31] S. C. Kontogiannis, P. N. Panagopoulou, P. G. Spirakis, Polynomial Algorithms for Approximating Nash Equilibria of Bimatrix Games, in: P. G. Spirakis, M. Mavronicolas, S. C. Kontogiannis (Eds.), Internet and Network Economics, Second International Workshop, WINE 2006, December 15-17, Proceedings, Vol. 4286 of Lecture Notes in Computer Science, Springer, Patras, Greece, 2006, pp. 286–296. doi:10.1007/11944874_26.

[32] L. G. Khachiyan, Polynomial algorithms in linear programming, USSR Computational Mathematics and Mathematical Physics 20 (1) (1980) 53–72. doi:10.1016/0041-5553(80)90061-0.

[33] d. B. Mark, C. Otfried, v. K. Marc, O. Mark, Computational Geometry Algorithms and Applications, Spinger, 2008.

[34] P. McMullen, The maximum numbers of faces of a convex

polytope, Mathematika 17 (2) (1970) 179–184. doi:10.1112/S0025579300002850.

[35] M. E. Dyer, The Complexity of Vertex Enumeration Methods, Mathematics of Operations Research 8 (3) (1983) 381–402. doi:10.1287/moor.8.3.381.

[36] D. Avis, K. Fukuda, A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra, in: Proceedings of the Seventh Annual Symposium on Computational Geometry, 1991, pp. 98–104.

[37] C. L. Assad, G. Morales, J. Arica, VERTEX ENUMERATION OF POLYHEDRA, Pesquisa Operacional 42 (2022). doi:10.1590/0101-7438.2022.042.00254570.

[38] L. Khachiyan, E. Boros, K. Borys, V. Gurvich, K. Elbassioni, Generating all vertices of a polyhedron is hard, Twentieth Anniversary Volume: Discrete & Computational Geometry (2009) 1–17.

[39] L. Khachiyan, Transversal hypergraphs and families of polyhedral cones, Advances in Convex Analysis and Global Optimization: Honoring the Memory of C. Caratheodory (1873–1950) (2001) 105–118.

[40] J. Nocedal, S. J. Wright, Numerical Optimization, Springer, New York, NY, USA, 1999. doi:10.1007/b98874.

[41] G. M. Ziegler, Lectures on Polytopes, Springer New York, New York, NY, 1995.

[42] M. B. Cohen, Y. T. Lee, Z. Song, Solving linear programs in the current matrix multiplication time, Journal of the ACM (JACM) 68 (1) (2021) 1–39.

[43] K. Etessami, M. Yannakakis, On the Complexity of Nash Equilibria and Other Fixed Points, SIAM Journal on Computing 39 (6) (2010) 2531–2597. doi:10.1137/080720826.

[44] L. Blum, M. Shub, S. Smale, On a theory of computation and complexity over the real numbers: $NP$- completeness, recursive functions and universal machines, Bulletin of the American Mathematical Society 21 (1) (1989) 1–46. doi:10.1090/S0273-0979-1989-15750-9.

[45] S. Sahni, Computationally Related Problems, SIAM Journal on Computing 3 (4) (1974) 262–279. doi:10.1137/0203021.

[46] M. Bellare, P. Rogaway, The complexity of approximating a nonlinear program, Mathematical Programming 69 (1-3) (1995) 429–441. doi:10.1007/BF01585569.

[47] J. Matousek, Lectures on Discrete Geometry, Vol. 212, Springer Science and Business Media, New York, NY, 2013.