Thesis

Algorithms and Iterative Methods for Infinite Games

Global Optimization in Game Theory and Beyond

Tomáš Votroubek

2026

Artificial Intelligence Center
Faculty of Electrical Engineering
Czech Technical University in Prague

Acknowledgments

I based this thesis on papers written with Tomáš Kroupa and Sara Vannucci. I would like to thank the organizers of the Barrande fellowship programme for the opportunity to study at LAAS-CNRS, and to Didier Henrion for his guidance during my short research stay.

Contents

1. Introduction

In practice, the inherent complexity of game-theoretical concepts often necessitates simplifying assumptions to model interactions between agents. When it is possible to map our problem to one of the well-studied classes of games, established solution concepts can usually be practically computed. Interesting interactions in the real world care little about game theory, however, and they appear arbitrarily far from any of the convenient classes of games. In scenarios where the reductions introduced by such mapping would be too drastic, we may still gain some insight into players' behavior with ad hoc heuristics, such as by discretizing the action spaces. While these approaches can be useful, they may lack robustness and guarantees.

In this thesis, we explore methods capable of providing guarantees or certificates on the quality of their solutions across a broad range of games. Consequently, we explore methods for global optimization, which appears as a very common sub-task in game theory. The ability to check global optimality is, for example, useful in answering whether an agent in an interaction is behaving rationally.

Finding global optima of functions over a set in the absence of any assumptions is a hard problem; even approximating such solutions is hard. Nevertheless, there exist classes of problems where global optimization is feasible. We demonstrate how Lasserre's hierarchies and the Spatial Branch and Bound method can be effectively applied to solve tasks in game theory and robotics. Our research indicates that the practicality of global methods remains largely unexplored despite the potential.

1.1. Contributions and Thesis Structure

This thesis covers results published under the supervision of Tomáš Kroupa on the solutions of games with infinite action spaces. It focuses on a generically applicable algorithm called *Multiple Oracle*, and shows how the powerful underlying global optimisation techniques extend to applications in robotics. Finally, it outlines the future direction for our research.

Nash Equilibria in Separable Network Games We provide an infinite-dimensional linear program and a Moment–Sum-of-Squares hierarchy for Nash equilibria in polynomial separable network games in Section 3.2. See Theorem 5, and Examples 3 and 4.

Our infinite-dimensional linear program for continuous network games is a generalisation of the linear program by Cai et al. [13] originally applicable to games with finite action spaces.

Our code at `https://github.com/votroto/PolyNets.jl` solves polynomial separable network games using Moment–Sum-of-Squares hierarchies.

Epsilon Equilibria in Continuous Games In Section 3.3, we extend the Double Oracle algorithm to multi-player general-sum continuous games, prove the convergence to ϵ-equilibria in the Wasserstein metric under exact best-response oracles, and empirically evaluate it in Appendix B on a wide library of continuous games.

Our program for multi-player general-sum continuous games is a generalisation of the Double Oracle algorithm by McMahan, Gordon, and Blum [55], originally designed for zero-sum matrix games, and of its later generalisation by Adam et al. [2] to two-player zero-sum continuous games.

Relevant Julia source code is published at `https://github.com/votroto/Quack.jl`.

Optimal Inverse Kinematics In Chapter 4, we develop a bilinear lifting scheme for polynomial inverse kinematics with possible applications to manipulator design, enabling the computation of globally optimal solutions for generic manipulators with up to 10 degrees of freedom.

Our scheme substantially improves upon the results of Trutman et al. [90], who demonstrated that inverse kinematics with 7 degrees of freedom are solvable using a method based on algebraic geometry.

Relevant Julia source code is published at `https://github.com/votroto/IK.jl`.

List of Continuous Games Appendix A contains our curated and standardised library of continuous-game benchmarks.

2. Methods of Global Optimization

The analysis of competitive interactions between agents often leads to challenging mathematical optimization problems [18]. Game theory traditionally assumes that agents are motivated solely by the goal of minimizing convex cost functions [10]. This assumption is primarily a mathematical convenience to ensure the tractability of optimization problems and the existence of well-known solution concepts. When this assumption is violated, computing various stable states of interactions becomes difficult, and even determining their existence can be computationally hard [21]. Similarly, when agents can choose from an infinite number of distinct actions, finding actions that result in favorable outcomes reduces to global optimization over nonconvex functions.

Nevertheless, there are combinations of reasonable solution concepts and classes of games that are both tractable and widely applicable. Allowing agents to play randomized strategies guarantees the existence of stable points but still poses the challenge of global optimization. In this chapter, we introduce two prominent approaches to global optimization: Lasserre's hierarchies and the spatial Branch and Bound method. In later chapters, these methods will enable us to solve games with infinite action spaces and nonconvex cost functions. Furthermore, we will demonstrate their utility in the context of robotics.

Example applications of Lasserre hierarchies include network-game equilibria in Section 3.2.2 and best-response *oracles* in Examples 2 and 3. In most of the examples in Appendix B, best responses are computed using spatial Branch and Bound, which is also applied to inverse kinematics in Chapter 4. Note that our inverse kinematics formulation follows prior work of Trutman et al. [90], which is based on Lasserre hierarchies.

2.1. Lasserre Hierarchies

The moment-SOS hierarchies are a basis of methods for constrained polynomial optimisation. More specifically, the methods are able to find the minimum of a polynomial over a compact set defined by polynomial inequalities. In some instances, they can even recover some minimisers as well as a certificate of global optimality. All of this is achievable under some mild assumptions on the structure of the feasible set, such as having a bound on the norm of the feasible elements. Under such constraints, each level of the hierarchy corresponds to a dual[1] pair of semidefinite programs which provide a monotonically nondecreasing sequence of lower bounds that generically converge [61] to the optimum. The current downsides of the hierarchies are that: the bounds on the convergence are not yet well understood; semidefinite solvers are not as well optimised for this task; and the recovery of minimisers of sufficient precision may be practically challenging.

[1] Strong duality holds if the original constraint set has an interior point, or if either program is strictly feasible.

Let $\mathbf{g} = (g_1, \dots, g_k)$ be a tuple of polynomials. A *basic semialgebraic set* $\mathcal{S}(\mathbf{g}) = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{g}(\mathbf{x}) \geq 0\}$ is a finite intersection of superlevel sets defined by $\mathbf{g}$. Polynomial Optimization Problems aim to find the global minimum of a polynomial f over the set $K = \mathcal{S}(\mathbf{g})$:

$$\min_{\mathbf{x} \in K} \quad f(\mathbf{x}). \tag{POP}$$

Two complementary techniques form the basis of polynomial optimization methods called the Moment-SOS hierarchies [48]. The first approach involves maximizing a lower-bound α on the condition that $f - \alpha$ is non-negative on K:

$$\begin{aligned} \max_{\alpha \in \mathbb{R}} \quad & \alpha \\ \text{s.t.} \quad & f(\mathbf{x}) - \alpha \in \mathcal{P}(K) \end{aligned} \tag{2.1}$$

where $\mathcal{P}(K)$ is the set of polynomials nonnegative on K. The second approach is based on the problem of finding a measure which minimizes the Lebesgue integral of f:

$$\min_{\mu \in \mathcal{M}(K)} \quad \int_K f \, \mathrm{d}\mu \tag{2.2}$$

where $\mathcal{M}(K)$ is the set of regular Borel probability measures on K. Problems 2.1 and 2.2 are convex reformulations of POP, and their optimal values agree exactly. The practical versions of these problems are semidefinite programs which are their strenghtenings and relaxations respectively.

2.1.1. Nonnegative Polynomials Using Sums of Squares

As we represent polynomials by their coefficients, we need sufficient conditions on the coefficients so that we can algorithmically decide the question of nonnegativity on sets. To do this, we use *preorders* and *quadratic modules*.

Just as ideals preserve the set of zeros, preorders preserve positivity on sets. Observe that if a polynomial can be expressed as a sum of squares[2] (SOS), it is necessarily nonnegative everywhere. Further notice that any sum or product of polynomials nonnegative on a set is still nonnegative on that set. A *preorder* generated by a polynomial tuple $\mathbf{g}$ is the set of all sums of all combinations of the g_i weighted by SOS polynomials. Schmüdgen proved [80, Corollary 3] that polynomials strictly positive on a compact $\mathcal{S}(\mathbf{g})$ are members of the preorder generated by $\mathbf{g}$. Furthermore, Putinar's refinement [72] of the theorem states, that for $\mathcal{S}(\mathbf{g})$ with an algebraic certificate of compactness, the cross products between the elements of $\mathbf{g}$ can be ignored in the decomposition. That is, polynomials strictly positive on $\mathcal{S}(\mathbf{g})$ are members of the *quadratic module* $\mathcal{Q}(\mathbf{g})$ generated by $\mathbf{g}$:

$$\mathcal{Q}(\mathbf{g}) = \left\{ s_0 + \sum_{i=1}^{k} g_i s_i \;\middle|\; s_0, s_1, \dots, s_k \text{ are SOS polynomials} \right\}. \tag{2.3}$$

[2] Given $p \in \mathbb{R}[x]$, if there exist $s_1, \dots, s_k \in \mathbb{R}[x]$ such that $p = \sum_{i=1}^{k} s_i^2$, then $p(\mathbf{x}) \geq 0 \; \forall x \in \mathbb{R}$.

Definition 1 (Archimedean Quadratic Module)**.** Following Laurent [51, Definition 3.18], we say that $\mathcal{Q}(\mathbf{g})$ is *Archimedean*, if the superlevel set of one of the $g_i(\mathbf{x})$ is compact, or if there exists $a > 0$ such that $a - \|\mathbf{x}\|^2 \in \mathcal{Q}(\mathbf{g})$.

Theorem 1 (Putinar's Positivstellensatz [71])**.** *Let $\mathcal{Q}(\mathbf{g})$ be Archimedean. If a polynomial p satisfies $p(\mathbf{x}) > 0\ \forall \mathbf{x} \in \mathcal{S}(\mathbf{g}) \subseteq \mathbb{R}^n$, then $p \in \mathcal{Q}(\mathbf{g})$.*

We remark that the Archimedean condition is very mild. Many sets important for applications are in fact the solution sets of finitely many polynomial inequalities where the bounds on the variables are known. In particular, such a semi-algebraic set is not necessarily convex or connected.

With a multi-index $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_m) \in \mathbb{N}^m$ we use the standard notation $\mathbf{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} \cdots x_m^{\alpha_m}$ to define monomials. For any $d \in \mathbb{N}$, we define the set of multi-indices truncated to degree d,

$$\mathbb{N}^m_d = \{\boldsymbol{\alpha} \in \mathbb{N}^m \mid \alpha_1 + \cdots + \alpha_m \leq d\}.$$

To formulate the sum of squares condition using semidefinite constraints, we define the basis of the linear space of polynomials with degree at most d as

$$[\mathbf{x}]_d = (\mathbf{x}^{\boldsymbol{\alpha}})_{\boldsymbol{\alpha} \in \mathbb{N}^m_d}.$$

Any polynomial $p(\mathbf{x})$ with the coefficient vector $\mathbf{c} = (c_{\boldsymbol{\alpha}})_{\boldsymbol{\alpha} \in \mathbb{N}^m_d}$ can then be expressed as

$$p(\mathbf{x}) = \sum_{|\boldsymbol{\alpha}| \leq d} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}} = \mathbf{c}^\intercal [\mathbf{x}]_d.$$

A polynomial $p(\mathbf{x})$ of degree at most $2d$ is SOS if, and only if, there exists a positive semidefinite matrix G such that $p(\mathbf{x}) = [\mathbf{x}]_d^\intercal G [\mathbf{x}]_d$ [48, Proposition 2.1]. We can therefore formulate the problem of checking whether such $p(\mathbf{x})$ is SOS as the semidefinite feasibility problem for a matrix G indexed by $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$:

$$\begin{aligned} \sum_{\boldsymbol{\beta}+\boldsymbol{\gamma}=\boldsymbol{\alpha}} G_{\boldsymbol{\beta}\boldsymbol{\gamma}} &= c_{\boldsymbol{\alpha}} \quad \forall \boldsymbol{\alpha}, |\boldsymbol{\alpha}| \leq 2d, \\ G &\succeq 0. \end{aligned}$$

Membership of p in $Q(\mathbf{g})$ can then be formulated as a single semidefinite program by applying such conditions to each polynomial variable in Theorem 1 [48, Chapter 2.4.2].

2.1.2. Existence of Representing Measures

To manipulate measures algorithmically, we represent them as functionals $l : \mathbb{R}[\mathbf{x}] \to \mathbb{R}$. Every linear functional l can be encoded by its *moment sequence* $\mathbf{y} = (l(x^{\boldsymbol{\alpha}}))_{\boldsymbol{\alpha}}$. The space of such functionals is the dual space $\mathbb{R}[\mathbf{x}]'$, and for $f(\mathbf{x}) = \sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$ we can write $l(f) = \sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} y_{\boldsymbol{\alpha}}$.

Given a compact K and $f \in \mathbb{R}[\mathbf{x}]$, if l in nonnegative on $\mathcal{P}(K)$ then $l(f)$ has the form:

$$l(f) = \int_K f \, d\mu,$$

where the unique $\mu \in \mathcal{M}(K)$ is called the *representing measure* of l [71].

However, regular Borel measures are a subset of the space of linear functionals. As before, we need a systematic way to carve out just those linear functionals that represent integration with respect to a measure over a semialgebraic set K. This is a version of the *K-moment problem*; i.e. the characterization of those real sequences $\mathbf{y}$ which have a representing measure.

Define $\mathcal{M}_d(K)$ as the cone of moments up to degree d which have a representing measure on K. Similarly, define $\mathcal{P}_d(K)$ as the cone of polynomials up to degree d which are nonnegative on K. If K is compact [51], the *moment cone* $\mathcal{M}_d(K)$ is dual to $\mathcal{P}_d(K)$:

$$\mathcal{M}_d(K) = \mathcal{P}_d(K)' = \left\{ l \in \mathbb{R}[\mathbf{x}]_d' \,\middle|\, l(p) \geq 0 \quad \forall p \in \mathcal{P}_d(K) \right\}.$$

Membership of a sequence $\mathbf{y}$ in $\mathcal{M}_d(K)$ can be encoded as the constraint on the eigenvalues of infinitely many *moment* and *localizing* matrices. Let $d \in \mathbb{N}$ and $\mathbf{y} = (y_\alpha)_{\alpha \in \mathbb{N}^m}$ be a real sequence. The *moment matrix of order d* is the matrix $M_d(\mathbf{y})$ with rows indexed by $\boldsymbol{\alpha} \in \mathbb{N}^m_d$ and columns indexed by $\boldsymbol{\beta} \in \mathbb{N}^m_d$ such that

$$(M_d(\mathbf{y}))_{\alpha\beta} = y_{\alpha+\beta}.$$

Each polynomial g_i defining the semialgebraic set $\mathcal{S}(\mathbf{g})$ can be written as

$$g_i(\mathbf{x}) = \sum_\gamma c_\gamma \mathbf{x}^\gamma$$

for some real coefficients $(c_\gamma)_{\gamma \in \mathbb{N}^m}$, of which only finitely many are nonzero. Then, the *localizing matrix of order d* is the matrix $M_d(g_i, \mathbf{y})$ with rows indexed by $\boldsymbol{\alpha} \in \mathbb{N}^m_d$ and columns indexed by $\boldsymbol{\beta} \in \mathbb{N}^m_d$, where

$$(M_d(g_i, \mathbf{y}))_{\alpha\beta} = \sum_\gamma c_\gamma y_{\alpha+\beta+\gamma}.$$

Theorem 2 (Lasserre [48, Theorem 2.44(b)]). *Let* $\mathbf{g}$ *be a tuple such that* $\mathcal{Q}(\mathbf{g})$ *is Archimedean. The moment sequence* $\mathbf{y}$ *with* $y_{\mathbf{0}} = 1$ *has a representing Borel measure supported in* $\mathcal{S}(\mathbf{g})$ *if, and only if,*

$$\begin{aligned} M_d(\mathbf{y}) &\succeq 0 \quad \forall d \in \mathbb{N}, \\ M_d(g_i, \mathbf{y}) &\succeq 0 \quad \forall d \in \mathbb{N},\ \forall i \in \{1, \dots, k\}. \end{aligned} \tag{2.4}$$

where $A \succeq 0$ *denotes that a real symmetric matrix* A *is positive semidefinite.*

2.1.3. Hierarchy of Semidefinite Relaxations

The final step is to put bounds on the sizes of moment sequences and degrees of polynomials using the Lasserre's hierarchies [48, Chapter 6]. Each relaxation in the hierarchy is a convex semidefinite program that approximates the solution of the original program.

We consider the quadratic module $\mathcal{Q}_h(\mathbf{g})$ truncated to a degree h:

$$\mathcal{Q}_h(\mathbf{g}) = \left\{ s_0 + \sum_{i=1}^{k} g_i s_i \,\middle|\, \begin{array}{l} \mathbf{s} \text{ are SOS polynomials,} \\ \deg(g_i s_i) \leq h, \forall i \in 1, \dots, k \\ \deg(s_0) \leq h \end{array} \right\}$$

Instead of testing polynomial positivity by checking its membership in $Q(\mathbf{g})$ the h-th level of the hierarchy tests for membership in $Q_h(\mathbf{g})$. Similarly, instead of requiring that the constraints hold for all moments, the h-th program in a hierarchy only constrains the moment and localizing matrices of order h. Moreover, we consider only truncated moment sequences $\hat{\mathbf{y}}$, which are just finite-dimensional real vectors. Under our initial Archimedean assumption, the optimization problems are mutually strongly-dual. Each level of the hierarchy is then a semidefinite program which under-estimates the optimum. By going to higher levels, the original problem can be approximated arbitrarily well.

2.1.4. Recovery of Global Minimizers

It is possible to certify the convergence of the Moment hierarchy and to extract the minimizers in a robust way [43]. On the other hand, neither task is directly possible from the side of SOS polynomials, except via duality.

Theorem 3 (Lasserre [49])**.** *The sequence* $\mathbf{y}$ *of* $2t$ *moments has a representing measure* μ *on a semialgebraic set* $S(\mathbf{g})$*, if and only if:*

$$\begin{aligned} M_t(\mathbf{y}) &\succeq 0, \\ M_{t-v}(g_i\mathbf{y}) &\succeq 0 \quad \forall g_i \in \mathbf{g}, \\ \operatorname{rank} M_t(\mathbf{y}) &= \operatorname{rank} M_{t-v}(\mathbf{y}), \end{aligned}$$

where $v = \left\lceil \frac{1}{2} \deg(\mathbf{g}) \right\rceil$*. Furthermore* μ *is* r*-atomic, with* $r = \operatorname{rank} M_d(\mathbf{y})$*.*

Laurent [51, Section 6.7] describes a technique to extract solutions from a truncated sequence of moments[3]. Rank-r moment matrices are convex combinations of r rank-1 matrices corresponding to Dirac measures supported on the minimizers. The minimizers can be computed using a linear algebra algorithm: Given an optimal $M_t(\mathbf{y}^\star)$, first, obtain its decomposition $M_t(\mathbf{y}^\star) = LL'$. Secondly, Convert L to column-echelon form, and for each variable x_i, construct the square multiplication matrix N_i by extracting from L the columns corresponding to the pivots, and the rows corresponding to the pivot monomials multiplied by x_i. Finally, compute the eigenvectors of a random convex combination of the multiplication matrices to get their common eigenvectors $\mathbf{v}$. The optimal measure is then supported on the set:

$$\left\{ \left(v_j^\top N_1 v_j, \dots \right) \mid j \in 1, \dots, r \right\}.$$

The convergence rate of the hierarchy is not yet well understood, and it may not be practically possible to reach levels high enough to certify convergence. Regardless of whether a truncated sequence describes a measure or a generic linear functional, we may try the first-order moments as candidate minimizers. The hierarchies yield a nondecreasing sequence of lower bounds on the optimal value of the original problem, while any feasible evaluation of the original problem provides an upper bound. If the first-order moments are feasible and the evaluation of the original problem equals the value of the relaxation, then we have a lower bound equal to the upper bound and therefore a proof of optimality.

[3] *GloptiPoly* extracts solutions using this technique [36]

In other cases, it may still be possible to glean some insight into the set of minimizers through sublevel sets of the Christoffel-Darboux polynomial [50].

2.2. Spatial Branch and Bound

Spatial Branch and Bound (BB) is a method to find global optima of non-linear mathematical programs by means of splitting the feasible region in a tree-like pattern and constructing locally-valid cuts from its constraints. BB maintains upper and lower bounds on the global optimum, and the gap between them serves as a proof on how suboptimal the current solution could be. When the gap is smaller than a user-specified threshold, the algorithm stops.

Each step of the algorithm is a branch associated with a set of local bounds on the variables. At each step, the method constructs a convex relaxation for each constraint separately; the relaxed constraints are then solved, e.g. using linear programming. Solutions of the relaxations are lower-bounds on the optimum within the branch and any of its child branches. The worst-case lower-bound in any currently active branch is then a lower bound on the global optimum of the original problem. To improve the lower-bounds, a variable is selected to branch on and its bounds are split into several sub-regions. At the same time, any admissible point for the original problem is an upper-bound on the global optimum. While the program could branch deep enough for the local relaxation to yield an admissible point, it is usual to use heuristics to accelerate the search for feasible points. Any branch where the relaxed sub-problem gives a value worse than an admissible solution can be pruned away. Fig. 2.1 illustrates how BB uses progressively tighter locally-valid cuts to improve the approximation of a concave feasible region defined by quadratic inequalities.

In the rest of the paper, we will show that BB can be performant and scale to large problem sizes, on top of being widely applicable. Even in cases where established techniques exist, such as Lasserre's hierarchies applied to polynomial optimization, BB methods can be competitive. In fact, recent results [22] indicate that considering random instances of some problems results in polynomial time complexity for BB methods.

2.2.1. Convex Relaxations

During the bounding phase of BB, the algorithm constructs a relaxation of the original problem, transforming it into a form that is practically solvable. This likely means relaxing the original problem into a convex one. This process relies on the ability to create reasonable relaxations of the constraints. The literature provides tight convex relaxations for common nonconvex terms, including bilinear and fractional terms, exponentiation, and wholly convex or concave functions. Although tight relaxations may not be generically available, the existing set of relaxations is sufficient to address all algebraic functions [82].

Algebraic functions can be decomposed into a hierarchy of binary operations, each corresponding to one of the aforementioned functions. Consequently, all algebraic functions can be solved using BB by introducing auxiliary variables and appropriate constraints. An implementation of such an algorithm is currently available in the Couenne solver [53]. As an example, to address the multilinear constraint $xyz = 0$, we could introduce a lifting variable

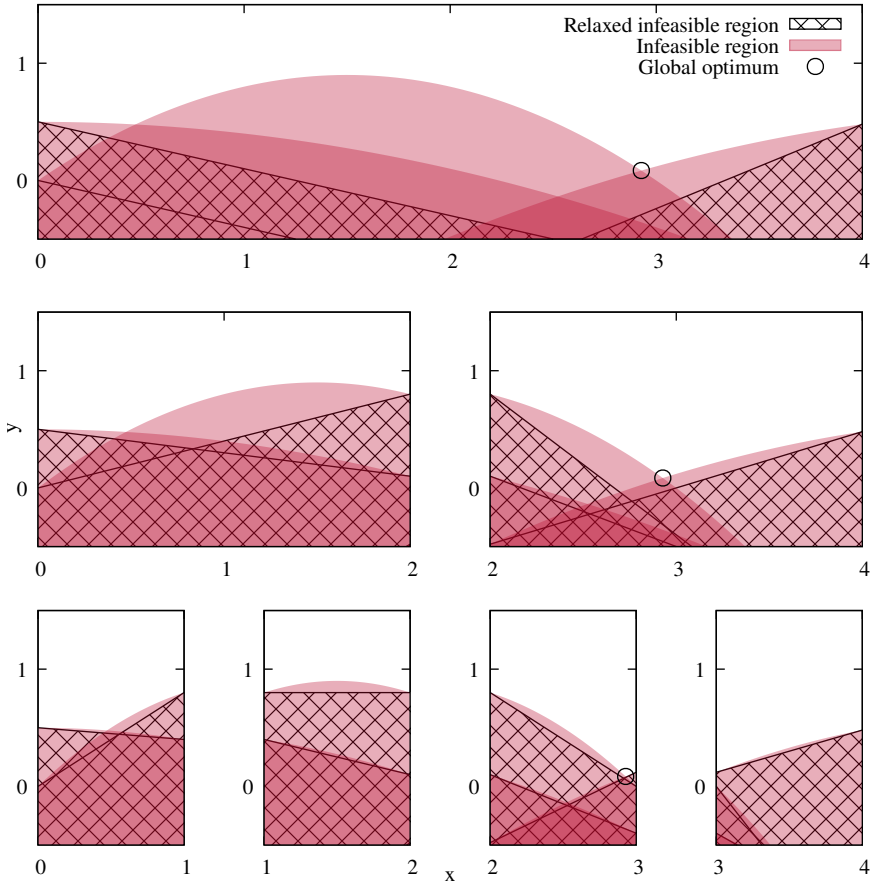


Figure 2.1.: An example of a Spatial Branch and Bound tree of a non-convex program minimizing the variable y over $x \in [0, 4]$ such that it lies above the indicated quadratic constraints. The infeasible region is shaded in red, and the locally-valid relaxations via secant cuts are hatched in black.

$\alpha = xy$ and rewrite the original constraint as $\alpha z = 0$. The result are two constraints for which tight convex relaxations exist.

Symbolic reformulations increase the dimensionality of the problem, which is a significant concern given that the worst-case complexity of the branch-and-bound algorithm is exponential in the number of variables. However, in Chapter 4, we will demonstrate that these transformations can be competitive when compared to Lasserre's hierarchies, particularly for inverse kinematics of generic redundant manipulators. The polynomial problems arising from inverse kinematics constraints involve more than fourteen variables and degrees greater than four.

2.2.2. Global Optimality Certificates

In comparison to the obvious certificate of global optimality obtained when a feasible solution is found by an outer approximation method such as the Lasserre's hierarchies described in Section 2.1, proving the optimality for the spatial Branch and Bound is less direct. While solvers could return a certificate proving that the solution found is globally optimal within a predefined margin of error, this would reveal the cuts taken by the solver. In the rest of this manuscript, we will focus on the Gurobi solver, which uses a closed-source variant of the spatial BB method and does not return such certificates. The found optimal solution can be used to confirm Gurobi's final primal bound. The log output contains the BB tree information, but it seems impossible to completely reconstruct Gurobi's BB tree or view the specific added cuts. Admittedly, using Gurobi means a certain trade-off between certifying the global optimality and efficiency of the computations. In our experiments, we verified the solutions of BB metods by comparing them with Lasserre's hierarchies where possible.

2.2.3. Finding Feasible Solutions

Getting feasible solutions as early as possible in the BB tree is essential for performance. Using an NLP solver on the problem is an obvious heuristic for finding feasible solutions, and is often performed internally by solvers. However, reformulating problems such that they can be solved by BB methods, e.g. by means of introducing auxiliary variables, can make result in a more complex program and hide some of the original structure. As a result, an NLP solver executed before any reformulation may have an easier time finding a feasible solutions even if the BB solver attempts the same.

Additionally, there may already exist domain-specific methods for finding heuristic solutions with significantly lower complexity than BB methods. For example, in robotics, there exist methods that work in real-time. These methods could be useful both for warm-starting and as runtime heuristics for BB.

3. Applications in Continuous Games

Continuous games are multiplayer games in which strategy sets are compact and utility functions are continuous. Many application domains have a continuum of actions, such as the allocation of time, resources, location in space [40], or parameters of classifiers [96]. This also includes several games modeling cybersecurity scenarios [63, 96, 77].

Continuous games have equilibria in mixed strategies by Glicksberg's generalization of Nash's theorem [30], but these equilibria are typically very hard to characterize and compute. We point out the main difficulties in the development of algorithms and numerical methods for continuous games:

- The equilibrium can be any tuple of mixed strategies with infinite supports [42] or almost any tuple of finitely-supported mixed strategies [75].
- Bounds on the size of supports of equilibrium strategies are known only for particular classes of continuous games [84].
- Some important games have only mixed equilibria; for example, certain variants of Colonel Blotto games [31].
- Finding a mixed strategy equilibrium involves locating its support, which lies inside a continuum of points.

To the best of our knowledge, algorithms and numerical methods exist only for very special classes of continuous games. In particular, two-player zero-sum polynomial games can be solved using sum-of-squares optimization based on a sequence of semidefinite relaxations of the original problem [67, 46]. Başar and Olsder [5] have written a detailed analysis of equilibria for some families of games with a particular shape of utility functions (e.g., games of timing, bell-shaped kernels). Fictitious play, one of the principal learning methods for finite games, was recently extended to continuous action spaces and applied to Blotto games [29]. The dynamics of fictitious play have been analyzed only under further restrictive assumptions in the continuous setting [38]. No-regret learning, studied by Mertikopoulos and Zhou [56], can be applied to finding pure equilibria or to mixed strategy learning in finite games. Convergence guarantees for algorithms in the distributed environment solving convex-concave games and some generalizations thereof have been developed by Mertikopoulos et al. [57]. In a similar setting, Chasnov et al. [14] provides convergence guarantees to a neighborhood of a stable Nash equilibrium for gradient-based learning algorithms.

3.1. Continuous Games and Nash Equilibria

The player set is denoted by $N = \{1, \dots, n\}$. Each player $i \in N$ selects a pure strategy x_i from a nonempty compact set $X_i \subseteq \mathbb{R}^{d_i}$, where d_i is a positive integer. A pure strategy profile is the n-tuple $\mathbf{x} = (x_1, \dots, x_n) \in \mathbf{X}$, where $\mathbf{X} = X_1 \times \cdots \times X_n$. We assume that each utility function $u_i : \mathbf{X} \to \mathbb{R}$ is continuous. A *continuous game* is the tuple

$$\mathcal{G} = \langle N, (X_i)_{i \in N}, (u_i)_{i \in N} \rangle. \tag{3.1}$$

A mixed strategy of player i can be an arbitrary Borel probability measure $\mu_i \in \mathcal{M}(X_i)$. Note that this general framework allows us to consider mixed strategies whose support is uncountably infinite, which is inevitable for the coherent treatment of games over compact strategy sets.

Let $\mathbf{M} = \mathcal{M}(X_1) \times \cdots \times \mathcal{M}(X_n)$ and let $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n) \in \mathbf{M}$ be a profile of mixed strategies. For the sake of brevity, we use $\boldsymbol{\mu}$ in place of the corresponding product probability measure $\mu_1 \times \cdots \times \mu_n$ in the definitions below involving integrals. The expected utility of player $i \in N$ is the function $U_i : \mathbf{M} \to \mathbb{R}$

$$U_i(\boldsymbol{\mu}) = \int_{\mathbf{X}} u_i \, \mathrm{d}\boldsymbol{\mu}$$

This definition yields a function which can be effectively evaluated only in special cases, such as when the support of each μ_i is finite.

We define the index notation $\cdot_{-i}$ to be the tuple of all players but i, such that

$$\mathbf{X}_{-i} = \bigtimes_{\substack{j \in N \\ j \neq i}} X_j \quad \text{and} \quad \mathbf{M}_{-i} = \bigtimes_{\substack{j \in N \\ j \neq i}} \mathcal{M}(X_j).$$

For every $\boldsymbol{\mu} \in \mathbf{M}$ and each player $i \in N$, by $\boldsymbol{\mu}_{-i}$ we denote the restriction of $\boldsymbol{\mu}$ onto $\mathbf{M}_{-i}$. The value of expected utility of player i in a strategy profile $(x_i, \boldsymbol{\mu}_{-i}) \in X_i \times \mathbf{M}_{-i}$ is denoted by

$$U_i(x_i, \boldsymbol{\mu}_{-i}) = \int_{\mathbf{X}_{-i}} u_i(x_i, .) \; \mathrm{d}\boldsymbol{\mu}_{-i}. \tag{3.2}$$

Definition 2 (Nash [59]). A Nash equilibrium is a strategy profile $\boldsymbol{\mu}^\star \in \mathbf{M}$ satisfying

$$U_i(\boldsymbol{\mu}^\star) = \max_{x_i \in X_i} U_i(x_i, \boldsymbol{\mu}^\star_{-i}), \qquad \forall i \in N.$$

That is, each player's strategy is a best response against the strategies of the others.

Theorem 4 (Glicksberg [30]). *Every continuous game has a mixed strategy Nash equilibrium.*

3.2. Nash Equilibria in Separable Network Games

Separable network games [9, 13] are multiplayer strategic games in which players interact only with adjacent players in a simple undirected graph. The utility of each player results

from the aggregation of utilities in the corresponding general-sum two-player games. We extend this model to infinite *network games* whose strategy sets are compact subsets of the Euclidean space. We show that *Nash Equilibria* (NE) of a zero-sum continuous network game can be characterized as optimal solutions to a specific infinite-dimensional linear optimization problem. In particular, when the utility functions are multivariate polynomials, this optimization formulation enables us to approximate the equilibria using a hierarchy of semidefinite relaxations based on the Moment-SOS hierarchies discussed in Section 2.1.

3.2.1. Continuous Network Games

Separable network games can capture situations in which players distribute their capacities or resources in a continuous way across multiple targets to achieve control over them. The targets can be battlefields, inspection sites, exit points etc.

In the class of games considered in this chapter, each player $i \in N$ is identified with a node of an undirected graph (N, E) without loops. The presence of an edge $\{i, j\} \in E$ means that players i and j engage in a two-person general-sum strategic game in which the utility functions of players i and j are functions $u_{ij} : X_i \times X_j \to \mathbb{R}$ and $u_{ji} : X_j \times X_i \to \mathbb{R}$.

We will always assume that the functions u_{ij} and u_{ji} are at least integrable, ensuring that every expected utility appearing below is correctly defined. Typically, a player i participates in multiple pairwise games, in which case, the same strategy strategy $x_i \in X_i$ is implemented in all two-person games played with the adjacent players in the graph (N, E). Thus, the aggregated utility of player i is computed as

$$u_i(\mathbf{x}) = \sum_{\substack{j \in N \\ \{i,j\} \in E}} u_{ij}(x_i, x_j), \qquad \mathbf{x} \in \mathbf{X}. \tag{3.3}$$

The resulting n-person game with compact strategy sets is called a *separable network game*, or "network game" for short. We say that $\mathcal{G}$ is

- a *polymatrix game* when all strategy sets X_i are finite;
- *zero-sum* if $\sum_{i \in N} u_i(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathbf{X}$;
- *continuous* if each utility function u_i is continuous.

A class of network games often studied in the literature is the class of zero-sum polymatrix games. The "global" zero-sum assumption and finite strategy sets make such games computationally tractable [13, 26, 4, 78, 12].

Example 1. Let $N = \{1, 2, 3, 4\}$ and the graph (N, E) be the cycle on Fig. 3.1. This is the smallest example of a network game such that each player interacts only with a strict subset of other players. For example, the utility function of player 3 is $u_3(x_1, x_2, x_3, x_4) = u_{31}(x_3, x_1) + u_{34}(x_3, x_4)$. The zero-sum condition for this game can be formulated as $u_3 = -u_1 - u_2 - u_4$.

As a direct consequence of Glicksberg's theorem 4, every continuous network game $\mathcal{G}$ has a NE $\boldsymbol{\mu}^\star \in \mathbf{M}$ in mixed strategies. We will show that the NE in a zero-sum continuous

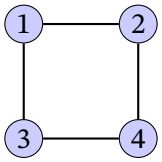


Figure 3.1.: A network game with 4 players and 4 pairwise games

network game can be characterized using an optimization-based approach. Specifically, we consider the following infinite-dimensional linear optimization problem, which generalizes *Linear Program 1* of Cai et al. [13]:

$$\begin{aligned}
\underset{\boldsymbol{\mu},\,\mathbf{w}}{\text{minimize}} \quad & \sum_{i\in N} w_i && && \text{(3.4a)}\\
\text{subject to} \quad & w_i \geq U_i(x_i, \boldsymbol{\mu}_{-i}) && \forall i \in N \;\; \forall x_i \in X_i, && \text{(3.4b)}\\
& \boldsymbol{\mu} = (\mu_1, \dots, \mu_n) \in \mathbf{M}, && && \text{(3.4c)}\\
& \mathbf{w} = (w_1, \dots, w_n) \in \mathbb{R}^n && && \text{(3.4d)}
\end{aligned}$$

Theorem 5. *For every profile of mixed strategies $\boldsymbol{\mu}^\star \in \mathcal{P}$ in a zero-sum continuous network game $\mathcal{G}$, the following are equivalent:*

1. *$\boldsymbol{\mu}^\star$ is a Nash equilibrium of game $\mathcal{G}$.*
2. *$(\boldsymbol{\mu}^\star, \mathbf{w}^\star)$ is an optimal solution to (3.4), where*

$$w_i^\star = \max_{x_i \in X_i} U_i(x_i, \boldsymbol{\mu}_{-i}^\star), \qquad i \in N. \tag{3.5}$$

Proof. Firstly, observe that the expression $\max_{\nu_i \in P_i} U_i(\nu_i, \boldsymbol{\mu}_{-i})$ is maximized by a ν_i supported on any of U_i's maximizers. Therefore, given a feasible $(\boldsymbol{\mu}, \mathbf{w})$, the constraint (3.4b) implies

$$w_i \geq \max_{x_i \in X_i} U_i(x_i, \boldsymbol{\mu}_{-i}) = \max_{\nu_i \in P_i} U_i(\nu_i, \boldsymbol{\mu}_{-i}) \geq U_i(\mu_i, \boldsymbol{\mu}_{-i}) \quad \forall i \in N. \tag{3.6}$$

Note that the maxima in inequalities (3.6) exist by compactness of X_i and the continuity of the functions $U_i(\cdot, \boldsymbol{\mu}_{-i}) : X_i \to \mathbb{R}$. Applying inequalities (3.6) in the objective, together with the zero-sum property, gives:

$$\sum_{i\in N} w_i \geq \sum_{i\in N} U_i(\mu_i, \boldsymbol{\mu}_{-i}) = 0. \tag{3.7}$$

By Theorem 4, there exists an NE $\boldsymbol{\mu}^\star$ such that $U_i(\boldsymbol{\mu}^\star) = \max_{x_i \in X_i} U_i(x_i, \boldsymbol{\mu}_{-i}^\star)$ $\forall i \in N$. Let $\boldsymbol{\mu}$ be an NE and $\mathbf{w}_i^\star$ be as in condition (3.5), then the inequalities in (3.6) become equalities, and the lower bound of zero is achieved in (3.7).

We have shown that NE correspond to optimal solutions. Conversely, consider any optimal solution $(\boldsymbol{\mu}^\star, \mathbf{w}^\star)$. Notice that $w_i^\star$ must be tight upper bounds on $U_i(\cdot, \boldsymbol{\mu}_{-i}^\star)$. By the zero-sum

property, $\sum_{i\in N} U_i(\boldsymbol{\mu}) = 0$ for any $\boldsymbol{\mu}$. If $\boldsymbol{\mu}^\star$ were not an NE, then there exists a player i such that $\max_{x_i\in X_i} U_i(x_i, \boldsymbol{\mu}_{-i}) > U_i(\boldsymbol{\mu})$. This implies that

$$0 = \sum_{i\in N} U_i(\boldsymbol{\mu}) < \sum_{i\in N} \max_{x_i\in X_i} U_i(x_i, \boldsymbol{\mu}_{-i}) = \sum_{i\in N} w_i,$$

i.e., the objective value is not optimal. □

The Borel probability measures and continuous functions in Problem 3.4 do not generally allow for effective computation. Still, there are cases where the solution can be computed. For example, in the case of zero-sum polymatrix games, Problem 3.4 becomes a linear program [13]. Similarly, if we restrict the model to polynomial functions and semialgebraic sets, the solution can be computed using the tools of polynomial optimization discussed in Section 2.1.

3.2.2. Polynomial Network Games

In this section, we make the following assumptions about the strategy spaces and the utility functions of a network game $\mathcal{G}$ defined by (3.1).

1. Every compact strategy space $X_i \subseteq \mathbb{R}^{m_i}$ is a *basic semi-algebraic set* $\mathcal{S}(\mathbf{g}_i)$ where each g_{ik} is a real polynomial in m_i variables and $K_i \neq \emptyset$ is a finite index set. We also assume the standard archimedean certificate on compactness 1 on the sets.

2. In every two-player game involving players $i, j \in N$, the utility functions u_{ij} and u_{ji} of player i and j, respectively, are real polynomials in $m_i + m_j$ indeterminates. Therefore, each utility function u_i given by (3.3) is a real polynomial in $m_1 + \cdots + m_n$ indeterminates.

3. The game $\mathcal{G}$ is zero-sum.

These three assumptions define a *polynomial zero-sum network game* $\mathcal{G}$. Glicksberg's theorem (4) can be significantly strengthened for polynomial games — each polynomial game has NE in which every player mixes only among finitely many pure strategies [91, 84].

The solutions for polynomial zero-sum network games can be solved using Lasserre's hierarchies as described in Section 2.1. Specifically, we relax the constraints on measures and polynomial inequalities of Program 3.4. Instead of requiring that the mixed strategies of players lie in the cone of probability measures, we apply constraints on truncated sequences of moments. Similarly, we replace the inequalities (3.4b) with the constraint that $w_i - U_i(\cdot, \boldsymbol{\mu}_{-i})$ lies in the truncated quadratic module.

The h-th level of the hierarchy is the following semidefinite program:

$$\begin{aligned}
\underset{\hat{\mathbf{y}}_1,\ldots,\hat{\mathbf{y}}_n,\mathbf{w}}{\text{minimize}} \quad & w_1 + \cdots + w_n & \\
\text{subject to} \quad & w_i - \bar{u}_i(\hat{\mathbf{y}}_{-i}) \in Q_{2h}(X_i) & \forall i \in N \\
& \hat{\mathbf{y}}_i = (y_{i,\boldsymbol{\alpha}})_{\boldsymbol{\alpha}\in\mathbb{N}^{m_i}_{2h}} \text{ and } y_{i,\mathbf{0}} = 1 & \forall i \in N \\
& M_h(\hat{\mathbf{y}}_i) \succeq 0 & \forall i \in N \\
& M_h(g_{ik}, \hat{\mathbf{y}}_i) \succeq 0 & \forall k \in K_i,\ \forall i \in N \\
& \mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n &
\end{aligned} \tag{3.8}$$

We then iteratively solve each level of the hierarchy until all sequences $\hat{\mathbf{y}}_i$ have a representing atomic measure, which we extract using the algorithm of Lasserre [48, Algorithm 6.9]. With the strategies of other players fixed, the best response condition is equivalent to certifying optimality for a polynomial over a semialgebraic set.

3.2.3. Experiments with Polynomial Network Games

In the remaining part of this paper we show several examples of polynomial games and recovered their equilibria by the method explained in the Section 3.2.2. We implemented the hierarchy of programs (3.8) in Julia using JuMP [24] and the SumOfSquares.jl package [94]. We solved the semidefinite programs using Mosek on a laptop with a 3.1GHz dual-core CPU.

Example 2 (Game A.3.19)**.** Consider a two-player zero-sum polynomial game played on the $[-1, 1]$ interval. The utility functions are

$$u_1(x_1, x_2) = 2x_1x_2^2 - x_1^2 - x_2 = -u_2(x_1, x_2).$$

Since polynomial network games are a generalization of polynomial games, we can use the same framework to solve them. A pure NE is achieved when Player 1 plays $x_1 \approx 0.4$ and Player 2 plays $x_2 \approx 0.63$. The optimal payoffs to each player are $\mathbf{w} \approx (-0.47, 0.47)$. This problem was solved at order 4 of the hierarchy. Due to the low number of variables the problem required only 44 constraints and it was solved within 10 ms.

Parrilo [68] demonstrated that the hierarchy is not necessary for two-player zero-sum games with interval strategy sets. In this case, Lukács' theorem [89, p. 1.21.1] states that there exists a Putinar decomposition in terms of rank-1 multipliers such that the term degrees do not exceed that of the input polynomial. The $z \in [-1, 1]$ strategy set would be encoded as $1 + z \geq 0 \wedge 1 - z \geq 0$ for inputs with odd degrees and $1 - z^2 \geq 0$ for even degrees. Both encodings satisfy the Archimedean property as the product of the defining polynomials is positive on a compact set. However, semidefinite solvers usually cannot guarantee rank-1 solutions.

Example 3 (Game A.3.28)**.** We consider a three-player polynomial network game with a complete graph, where each strategy set is the interval $[-1, 1]$ and the utility functions are

$$\begin{aligned}
u_1(x_1, x_2, x_3) &= -2x_1x_2^2 - 2x_1^2 + 5x_1x_2 - 4x_1x_3 - x_2 - 2x_3, \\
u_2(x_1, x_2, x_3) &= 2x_1x_2^2 - 2x_2x_3^2 - 2x_1^2 - 5x_1x_2 - 2x_2^2 + 5x_2x_3 + x_2, \\
u_3(x_1, x_2, x_3) &= 2x_2x_3^2 + 4x_1^2 + 4x_1x_3 + 2x_2^2 - 5x_2x_3 + 2x_3.
\end{aligned}$$

The game has the following partially-mixed Nash equilibrium:

$$\begin{aligned}\mu_1^\star &\approx \delta_{-0.06},\\ \mu_2^\star &\approx \delta_{0.35},\\ \mu_3^\star &\approx 0.72\delta_1 + 0.28\delta_{-1}.\end{aligned}$$

The corresponding utilities are $\mathbf{w} \approx (-1.22, 0.26, 0.97)$. This problem converged in around 10 ms at order 4, resulting in 96 constraints.

Example 4 (Game A.3.29)**.** We consider a polynomial security game with attackers $N_A = \{1, 2\}$, defenders $N_D = \{3, 4\}$. Player 3 protects targets t_1 and t_2, while player 4 defends t_3, t_4. The capacities of attackers are $a_1 = 0.5$, $a_2 = 0.8$, and the defenders' capacities are $d_3 = d_4 = 1$. The efficiency functions are:

$$\begin{aligned}e_1(x, y) &= 0.7x^2(1 - y^2)\\ e_2(x, y) &= (1.4x - 0.7x^2)(y^2 - 2y + 1)\\ e_3(x, y) &= (1.8x - 0.9x^2)(1 - y^2)\\ e_4(x, y) &= 0.9x^2(y^2 - 2y + 1)\end{aligned}$$

Attackers in security games can attack any target, but each defender can only protect targets within their jurisdiction. The interactions can be modelled as a bipartite graph of pairwise general-sum games over targets to which the players allocate resources up to their total capacity. Each target is associated with a resiliency function e_t. The total utility of a defender j is

$$u_j(x) = \sum_{i \in N_A} \sum_{t \in T_j} e_t(x_i, x_j),$$

that is, the sum of resiliency evaluations over all attackers N_A and all targets within their jurisdiction T_j; the total utility of an attacker i is

$$u_i(x) = a_i - \sum_{j \in N_D} \sum_{t \in T_j} e_t(x_i, x_j).$$

Thus the games are globally constant sum.

Our algorithm returned the following partially-mixed strategy profile at order 6:

$$\begin{aligned}\mu_1^\star &\approx \delta_{(0.29,0,0,0.21)},\\ \mu_2^\star &\approx \delta_{(0.36,0.13,0.04,0.24)},\\ \mu_3^\star &\approx 0.46\delta_{(0.33,0.67)} + 0.54\delta_{(0.69,0.31)},\\ \mu_4^\star &\approx 0.02\delta_{(0.46,0.54)} + 0.06\delta_{(0.08,0.92)} + 0.92\delta_{(0.91,0.09)}\end{aligned}$$

The payoffs to each player are $\mathbf{w} \approx (0.1, 0.3, -0.17, -0.23)$. While the payoffs converged at order 4 already and took only 3 seconds to solve, the corresponding strategies could not be extracted until order 6, which took 6 minutes to solve. Due to the high dimensionality of the utility functions and the fast growth of the hierarchy, the resulting SDP contained over 25 thousand constraints.

3.3. Epsilon Equilibria in Continuous Games

In this section we present an algorithm that is much more widely applicable at the cost of relaxing the requirement on the exactness of solutions from the previous section. We focus on the computation and approximation of mixed strategy equilibria for the whole class of multiplayer general-sum continuous games. To solve such generic games, we vastly extend the scope of applicability of the Double Oracle algorithm (DO), initially designed and proved to converge only for finite two-player zero-sum games [55]. Specifically, we propose an iterative strategy generation technique, which splits the original problem into the master problem with only a finite subset of strategies being considered, and the subproblem in which an oracle finds the best response of each player. This simple method is guaranteed to recover an approximate equilibrium in finitely many iterations.

Further, we argue that the Wasserstein distance (the earth mover's distance) represents a very natural metric on the space of mixed strategies. Our main result is the proof of convergence of this algorithm in the Wasserstein distance to an equilibrium of the original continuous game.

There are relatively few methods for computing or approximating equilibria of continuous games, which narrows down the continuous games for which the exact solution can be found and used as a benchmark in our experiments. We demonstrate the computational performance of our method on selected examples of games appearing in the literature and on randomly generated games.

[1] extended DO from finite games and proved that it converges for all two-player zero-sum continuous games. The algorithm is based on the iterative solution of finite subgames and their subsequent extension with best responses. Despite its simplicity, this method was successfully adapted to large extensive-form games [8], Bayesian games [52], and security domains with complex policy spaces [95].

3.3.1. Subgames and Epsilon-Equilibria

Consider a game $\mathcal{G} = \langle N, (X_i)_{i\in N}, (u_i)_{i\in N}\rangle$ and nonempty compact subsets $Y_i \subseteq X_i$ for each player i from the player set N. When each u_i is restricted to $Y_1 \times \cdots \times Y_n$, the continuous game $\langle N, (Y_i)_{i\in N}, (u_i)_{i\in N}\rangle$ is called the *subgame* of $\mathcal{G}$.

The *support* of a mixed strategy μ is the compact set defined by

$$\operatorname{spt}\mu = \bigcap\{K \subseteq X \mid K \text{ compact}, \mu(K) = 1\}.$$

In this section, we will construct mixed strategies with finite supports. The support of Dirac measure δ_x is the singleton $\operatorname{spt}\delta_x = \{x\}$, where $x \in X_i$. In general, when the support $\operatorname{spt}\mu$ of a mixed strategy μ is finite, it means that $\mu(x) > 0$ for all $x \in \operatorname{spt}\mu$ and $\sum_{x\in\operatorname{spt}\mu}\mu(x) = 1$. For clarity, we emphasize that a finitely-supported mixed strategy μ_i of player i should be interpreted as the function $\mu : X_i \to [0,1]$ vanishing outside $\operatorname{spt}\mu_i$, and not as a vector of probabilities with a fixed dimension. This is because only the former viewpoint enables us to consider the distance between *any* pair of pure strategies in X_i, which makes it possible to compute a distance between mixed strategies with *arbitrary* supports.

Proposition 1. *For each player $i \in N$ and any mixed strategy profile $\boldsymbol{\mu}_{-i} \in \mathbf{M}_{-i}$, there exists a pure strategy $x_i \in X_i$ such that*

$$\max_{\nu \in \mathcal{M}(X_i)} U_i(\nu, \boldsymbol{\mu}_{-i}) = \max_{x \in X_i} U_i(x, \boldsymbol{\mu}_{-i}) = U_i(x_i, \boldsymbol{\mu}_i).$$

Glicksberg's theorem (4) states that every continuous game has an equilibrium. The following useful characterization is a consequence of Proposition 1: A profile $\boldsymbol{\mu}^*$ is an equilibrium if, and only if,

$$U_i(x_i, \boldsymbol{\mu}^*_{-i}) \leq U_i(\boldsymbol{\mu}^*), \qquad \text{for every } i \in N \text{ and all } x_i \in X_i.$$

For any $\epsilon \geq 0$, an *ϵ-equilibrium* is a mixed strategy profile $\boldsymbol{\mu}^* \in \mathbf{M}$ such that

$$U_i(x_i, \boldsymbol{\mu}^*_{-i}) - U_i(\boldsymbol{\mu}^*) \leq \epsilon, \qquad \text{for every } i \in N \text{ and } x_i \in X_i.$$

This implies that for every $p_i \in \mathcal{M}(X_i)$, the inequality $U_i(p_i, \boldsymbol{\mu}^*_{-i}) - U_i(\boldsymbol{\mu}^*) \leq \epsilon$ also holds.

We introduce the vector notation

$$\mathbf{U}(\boldsymbol{\mu}) = (U_1(\boldsymbol{\mu}), \dots, U_n(\boldsymbol{\mu})).$$

Furthermore, we define

$$\mathbf{U}(\mathbf{x}, \boldsymbol{\mu}) = (U_1(x_1, \boldsymbol{\mu}_{-1}), \dots, U_n(x_n, \boldsymbol{\mu}_{-n})),$$

for any $\mathbf{x} \in \mathbf{X}$ and $\boldsymbol{\mu} \in \mathbf{M}$. Let $\boldsymbol{\epsilon}$ be the vector $(\epsilon, \dots, \epsilon)$. Using this vector notation, a mixed strategy profile $\boldsymbol{\mu}^* \in \mathbf{M}$ is an ϵ-equilibrium if, and only if,

$$\mathbf{U}(\mathbf{x}, \boldsymbol{\mu}^*) - \mathbf{U}(\boldsymbol{\mu}^*) \leq \boldsymbol{\epsilon}, \qquad \forall \mathbf{x} \in \mathbf{X}.$$

3.3.2. Convergence of Mixed Strategies

We consider an arbitrary metric ρ_i on the compact strategy space $X_i \subseteq \mathbb{R}^{m_i}$ of each player $i \in N$. This enables us to quantify a distance between pure strategies $x, y \in X_i$ by the number $\rho_i(x, y) \geq 0$. Consequently, we can define the Wasserstein distance d_W on $\mathcal{M}(X_i)$, which is compatible with the metric of the underlying strategy space X_i in the sense that

$$d_W(\delta_x, \delta_y) = \rho_i(x, y), \qquad \forall x, y \in X_i.$$

The preservation of distance from X_i to $\mathcal{M}(X_i)$ is a very natural property, since the space of pure strategies X_i is embedded in $\mathcal{M}(X_i)$ via the correspondence $x \mapsto \delta_x$ mapping the pure strategy x to the Dirac measure δ_x.

The Wasserstein distance originated from optimal transport theory. It is nowadays highly instrumental in solving many problems of computer science. Specifically, the *Wasserstein distance* of mixed strategies $\mu, \nu \in \mathcal{M}(X_i)$ is

$$d_W(\mu, \nu) = \inf_{\xi} \int_{X_i^2} \rho_i(x, y) \, \mathrm{d}\xi(x, y),$$

where the infimum is over all Borel probability measures ξ on X_i^2 whose one-dimensional marginals are μ and ν:

$$\xi(A \times X_i) = \mu(A),$$
$$\xi(X_i \times A) = \nu(A),$$

for all Borel subsets $A \subseteq X_i$.

The dependence of d_W on the metric ρ_i is understood. The function d_W is a metric on $\mathcal{M}(X_i)$. Since X_i is compact, it has necessarily bounded diameter. This implies that the convergence in $(\mathcal{M}(X_i), d_W)$ coincides with the weak convergence [65, Corollary 2.2.2]. Specifically, the following two assertions are equivalent for any sequence $(\mu^j)_{j\in\mathbb{N}}$ in $\mathcal{M}(X_i)$:

1. (μ^j) converges to μ in $(\mathcal{M}(X_i), d_W)$.
2. (μ^j) *weakly converges* to μ, which means by the definition that

$$\lim_j \int_{X_i} f \, \mathrm{d}\mu^j = \int_{X_i} f \, \mathrm{d}\mu,$$

for every continuous function $f : X_i \to \mathbb{R}$.

The metric space $(\mathcal{M}(X_i), d_W)$ is compact by [65, Proposition 2.2.3]. Consequently, the joint strategy space $\mathbf{M}$ is compact in a product metric as well, and any sequence $(\boldsymbol{\mu}^j)_{j\in\mathbb{N}}$ in $\mathbf{M}$ has an accumulation point. Equivalently, the previous assertion can be formulated as follows.

Proposition 2. *Any sequence of mixed strategy profiles $(\boldsymbol{\mu}^j)_{j\in\mathbb{N}}$ in $\mathbf{M}$ contains a weakly convergent subsequence.*

The function U_i is continuous on $\mathbf{M}$ by the definition of weak convergence. This implies that if $(\boldsymbol{\mu}^j)_{j\in\mathbb{N}}$ weakly converges to $\boldsymbol{\mu}$ in $\mathbf{M}$, then the corresponding values of expected utility goes to $U_i(\boldsymbol{\mu})$, that is, $\lim_j U_i(\boldsymbol{\mu}^j) = U_i(\boldsymbol{\mu})$. By compactness of $\mathcal{M}(X_i)$ and continuity of U_i, all maxima and maximizers appearing in the paper exist. This implies that the optimal value of utility function in response to the mixed strategies of other players is attained for some pure strategy. Proposition 1 is the precise formulation of this useful fact.

In the rest of this section, we will discuss the problem of computing and approximating the Wasserstein distance $d_W(\mu, \nu)$ of any pair of mixed strategies $\mu, \nu \in \mathcal{M}(X_i)$ for player i. In general, this is a difficult infinite-dimensional optimization problem [70]. In our setting it suffices to evaluate $d_W(\mu, \nu)$ only for mixed strategies with finite supports. In particular, it follows immediately from the definition of d_W that

$$d_W(\mu, \delta_y) = \sum_{x \in \operatorname{spt}\mu} \rho_i(x, y)\mu(x),$$

for every finitely-supported mixed strategy $\mu \in \mathcal{M}(X_i)$ and any $y \in X_i$. If mixed strategies μ and ν have finite supports, then computing $d_W(\mu, \nu)$ becomes the linear programming

problem

$$\begin{aligned}
\min_{\xi(x,y)} \quad & \sum_{x\in\operatorname{spt}\mu}\sum_{y\in\operatorname{spt}\nu}\rho_i(x,y)\xi(x,y) \\
& \xi(x,y)\geq 0 \qquad \forall (x,y)\in \operatorname{spt}\mu\times\operatorname{spt}\nu \\
& \sum_{x\in\operatorname{spt}\mu}\sum_{y\in\operatorname{spt}\nu}\xi(x,y)=1, \\
& \sum_{y\in\operatorname{spt}\nu}\xi(x,y)=\mu(x) \qquad \forall x\in\operatorname{spt}\mu \\
& \sum_{x\in\operatorname{spt}\mu}\xi(x,y)=\nu(y) \qquad \forall y\in\operatorname{spt}\nu
\end{aligned} \tag{3.9}$$

with variables $\xi(x,y)$ indexed by $(x,y)\in\operatorname{spt}\mu\times\operatorname{spt}\nu$.

3.3.3. The Multiple Oracle algorithm and its Correctness

We propose the Multiple Oracle algorithm (MO) as an iterative strategy-generation technique for (approximately) solving any continuous game $\mathcal{G}=\langle N,(X_i)_{i\in N},(u_i)_{i\in N}\rangle$. We recall that the *best response set* of player $i\in N$ with respect to a mixed strategy profile $\boldsymbol{\mu}_{-i}\in\mathbf{M}_{-i}$ is

$$\beta_i(\boldsymbol{\mu}_{-i})=\arg\max_{x\in X_i} U_i(x,\boldsymbol{\mu}_{-i}).$$

The set $\beta_i(\boldsymbol{\mu}_{-i})$ is always nonempty by Proposition 1. We assume that every player uses an oracle to recover at least one best response strategy, which means that the player is able to solve the corresponding optimization problem to global optimality. In Section 3.3.6 we will see the instances of games for which this is possible.

Algorithm MO The Multiple Oracle algorithm

function ITERATE($\mathcal{G}$, $\mathbf{X}$)
  Find an equilibrium $\boldsymbol{\mu}$ of the subgame $\langle N,\mathbf{X},(u_i)_{i\in N}\rangle$
  $\forall i\in N$: Pick a best response $x_i\in\beta_i(\boldsymbol{\mu}_{-i})$
  $\forall i\in N$: Extend strategy set $Y_i\leftarrow X_i\cup\{x_i\}$
  return $\mathbf{Y},\mathbf{x},\boldsymbol{\mu}$
end function
function MULTIPLE-ORACLE($\mathcal{G}$, $\mathbf{X}$, ϵ)
  repeat
    $\mathbf{X},\mathbf{x},\boldsymbol{\mu}\leftarrow$ iterate($\mathcal{G},\mathbf{X}$)
  until $\forall i\in N$: $U(\mathbf{x}_i,\boldsymbol{\mu}_{-i})-U(\boldsymbol{\mu})\leq\epsilon$
  return $\boldsymbol{\mu}$
end function

MO proceeds as follows: In every iteration j, finite strategy sets X_i^j are constructed for each player $i\in N$ and the corresponding finite subgame of $\mathcal{G}$ is solved. Let $\boldsymbol{\mu}^j$ be its equilibrium. Then, an arbitrary best response strategy x_i^{j+1} with respect to $\boldsymbol{\mu}_{-i}^j$ is added to each strategy set X_i^j. Those steps are repeated until

$$U_i(x_i^{j+1},\boldsymbol{\mu}_{-i}^j)-U_i(\boldsymbol{\mu}^j)\leq\epsilon \qquad \forall i\in N. \tag{3.10}$$

First we discuss basic properties of the algorithm. At each step j, we have the inequality

$$\mathbf{U}(\mathbf{x}^{j+1}, \boldsymbol{\mu}^j) \geq \mathbf{U}(\boldsymbol{\mu}^j). \tag{3.11}$$

Indeed, for each player $i \in N$, we get

$$U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^j) = \max_{x \in X_i} U_i(x, \boldsymbol{\mu}_{-i}^j) \geq \max_{x \in X_i^j} U_i(x, \boldsymbol{\mu}_{-i}^j) = U_i(\boldsymbol{\mu}^j).$$

We note that (3.10) is necessarily different from the stopping condition of DO. Namely, the latter condition [55], tailored to two-player zero-sum games, is

$$U_1(x_1^{j+1}, \mu_2^j) - U_1(x_2^{j+1}, \mu_1^j) \leq \epsilon. \tag{3.12}$$

When $\mathcal{G}$ is a two-player zero-sum continuous game, it follows immediately from (3.11) that (3.12) implies (3.10).

We prove correctness of MO — the eventual output $\boldsymbol{\mu}^j$ is an ϵ-equilibrium of the original game $\mathcal{G}$.

Lemma 1. *The strategy profile $\boldsymbol{\mu}^j$ is an ϵ-equilibrium of $\mathcal{G}$, whenever MO terminates at step j.*

Proof. Let $\mathbf{x} \in \mathbf{X}$. We get

$$\mathbf{U}(\mathbf{x}, \boldsymbol{\mu}^j) - \mathbf{U}(\boldsymbol{\mu}^j) = \mathbf{U}(\mathbf{x}, \boldsymbol{\mu}^j) - \mathbf{U}(\mathbf{x}^{j+1}, \boldsymbol{\mu}^j) + \mathbf{U}(\mathbf{x}^{j+1}, \boldsymbol{\mu}^j) - \mathbf{U}(\boldsymbol{\mu}^j).$$

Then $\mathbf{U}(\mathbf{x}, \boldsymbol{\mu}^j) - \mathbf{U}(\mathbf{x}^{j+1}, \boldsymbol{\mu}^j) \leq \mathbf{0}$, since $\mathbf{x}^{j+1}$ is the profile of best response strategies. Consequently, we obtain

$$\mathbf{U}(\mathbf{x}, \boldsymbol{\mu}^j) - \mathbf{U}(\boldsymbol{\mu}^j) \leq \mathbf{U}(\mathbf{x}^{j+1}, \boldsymbol{\mu}^j) - \mathbf{U}(\boldsymbol{\mu}^j) \leq \boldsymbol{\epsilon},$$

where the last inequality is just the terminating condition (3.10). Therefore, $\boldsymbol{\mu}^j$ is an ϵ-equilibrium of $\mathcal{G}$. □

If the set $\mathbf{X}^j = X_1^j \times \cdots \times X_n^j$ cannot be further inflated, MO terminates.

Lemma 2. *Assume that $\mathbf{X}^j = \mathbf{X}^{j+1}$ at step j of MO. Then*

$$\mathbf{U}(\mathbf{x}^{j+1}, \boldsymbol{\mu}^j) = \mathbf{U}(\boldsymbol{\mu}^j).$$

Proof. The condition $\mathbf{X}^j = \mathbf{X}^{j+1}$ implies $\mathbf{x}^{j+1} \in \mathbf{X}^j$. For each player $i \in N$,

$$U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^j) = \max_{x \in X_i^j} U_i(x, \boldsymbol{\mu}_{-i}^j) = U_i(\boldsymbol{\mu}^j).$$

□

The original Double Oracle algorithm for two-player zero-sum finite games [55] makes it possible to find exact equilibria. The following proposition generalizes this result to the setting of multiplayer general-sum finite games.

Proposition 3. *If $\mathcal{G}$ is a finite game and $\epsilon = 0$, then MO recovers an equilibrium of $\mathcal{G}$ in finitely-many steps.*

Proof. Let $\mathcal{G}$ be finite and $\epsilon = 0$. Since each X_i is finite, there exists an iteration j in which $\mathbf{X}^{j+1} = \mathbf{X}^j$. Then the terminating condition of MO is satisfied with $\epsilon = 0$ (Lemma 2) and $\boldsymbol{\mu}^j$ is an equilibrium of $\mathcal{G}$ (Lemma 1). □

Theorems 6 and 7 extend the convergence theorem from [1] and provides an alternative proof of Glicksberg's theorem [30] about existence of equilibria in continuous games. Indeed, the sequence $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots$ has at least one accumulation point by Proposition 2. Further, the consequence of Theorem 7 is that a finitely-supported ϵ-equilibrium of $\mathcal{G}$ can always be found in finitely-many steps, although game $\mathcal{G}$ may have no finitely-supported equilibria.

Theorem 6. *Let $\mathcal{G}$ be a continuous game and $\epsilon = 0$. If MO stops at step j, then $\boldsymbol{\mu}^j$ is an equilibrium of $\mathcal{G}$. Otherwise, any accumulation point of $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots$ is an equilibrium of $\mathcal{G}$.*

Proof. If MO terminates at step j, then Lemma 1 implies that $\boldsymbol{\mu}^j$ is an equilibrium of $\mathcal{G}$. In the opposite case, the algorithm generates a sequence of mixed strategy profiles $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots$ Consider any weakly convergent subsequence of this sequence — at least one such subsequence exists by Proposition 2. Without loss of generality, such a subsequence will be denoted by the same indices as the original sequence. Therefore, for each player $i \in N$, there exists some $\mu_i^\star \in \mathcal{M}(X_i)$, such that the sequence $\mu_i^1, \mu_i^2, \ldots$ weakly converges to $\mu_i^\star$. We need to show that $\boldsymbol{\mu}^\star = (\mu_1^\star, \ldots, \mu_n^\star) \in \mathbf{M}$ is an equilibrium of $\mathcal{G}$.

For all $i \in N$, define

$$Y_i = \bigcup_{j=1}^{\infty} X_i^j.$$

First, assume that $x \in Y_i$. Then there exists j_0 with $x \in X_i^j$ for each $j \geq j_0$. Hence the inequality $U_i(\boldsymbol{\mu}^j) \geq U_i(x, \boldsymbol{\mu}_{-i}^j)$, for each $j \geq j_0$, since $\boldsymbol{\mu}^j$ is an equilibrium of the corresponding finite subgame. Therefore,

$$U_i(\boldsymbol{\mu}^\star) = \lim_j U_i(\boldsymbol{\mu}^j) \geq \lim_j U_i(x, \boldsymbol{\mu}_{-i}^j) = U_i(x, \boldsymbol{\mu}_{-i}^\star).$$

Further, by continuity of U_i,

$$U_i(\boldsymbol{\mu}^\star) \geq U_i(x, \boldsymbol{\mu}_{-i}^\star) \quad \text{for all } x \in \mathrm{cl}(Y_i). \tag{3.13}$$

Now, consider an arbitrary $x \in X_i \setminus \mathrm{cl}(Y_i)$. The definition of x_i^{j+1} yields $U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^j) \geq U_i(x, \boldsymbol{\mu}_{-i}^j)$ for each j. This implies, by continuity,

$$\lim_j U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^\star) \geq \lim_j U_i(x, \boldsymbol{\mu}_{-i}^j) = U_i(x, \boldsymbol{\mu}_{-i}^\star). \tag{3.14}$$

Since $x_i^{j+1} \in X_i^{j+1}$, compactness of X_i provides a convergent subsequence (denoted by the same indices) such that $x' = \lim_j x_i^j \in \mathrm{cl}(Y_i)$. Then (3.13) gives

$$U_i(\boldsymbol{\mu}^\star) \geq U_i(x', \boldsymbol{\mu}_{-i}^\star) = \lim_j U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^\star). \tag{3.15}$$

Combining (3.14) and (3.15) shows that $U_i(\boldsymbol{\mu}^\star) \geq U_i(x, \boldsymbol{\mu}_{-i}^\star)$. □

Theorem 7. *Let $\mathcal{G}$ be a continuous game and $\epsilon > 0$. Then MO terminates at some step j and $\boldsymbol{\mu}^j$ is an ϵ-equilibrium of $\mathcal{G}$.*

Proof. If MO terminates at step j with a positive ϵ, then Lemma 1 implies that $\boldsymbol{\mu}^j$ is an ϵ-equilibrium of $\mathcal{G}$. Otherwise MO produces a sequence $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots$ and we can repeat the analysis as in Item 1 for convergent subsequences of $(\boldsymbol{\mu}^j)$ and $(\mathbf{x}^j)$, which are denoted by the same indices. Define $x' = \lim_j x_i^j$. Then

$$U_i(\boldsymbol{\mu}^\star) \geq U_i(x', \boldsymbol{\mu}_{-i}^\star) = \lim_j U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^j). \tag{3.16}$$

At every step j we have $U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^j) \geq U_i(\boldsymbol{\mu}^j)$ for each $i \in N$ by (3.11). For this reason $\lim_j U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^j) \geq U_i(\boldsymbol{\mu}^\star)$. Putting together the last inequality with (3.16), we get

$$\lim_j (U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^j) - U_i(\boldsymbol{\mu}^j)) = 0 \tag{3.17}$$

for each $i \in N$. This equality means that MO stops at some step j and $\boldsymbol{\mu}^j$ is an ϵ-equilibrium by Lemma 1. □

MO generates the sequence of equilibria $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots$ in increasingly larger subgames of $\mathcal{G}$. The sequence itself may fail to converge weakly in $\mathbf{M}$ even for a two-player zero-sum continuous game; see Example 1 from [1]. In fact, Theorems 6 and 7 guarantee convergence only to an accumulation point. We recall that this is a typical feature of some globally convergent methods not only in infinite-dimensional spaces [37, Theorem 2.2], but also in Euclidean spaces. For example, a gradient method generates the sequence such that only its accumulation points are guaranteed to be stationary points [6, Proposition 1.2.1]. One necessary condition for the weak convergence of $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots$ is easy to formulate using the stopping condition of MO.

Proposition 4. *If the sequence $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots$ generated by MO converges weakly to an equilibrium $\boldsymbol{\mu}^\star$, then $\lim_j \big(U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^j) - U_i(\boldsymbol{\mu}^j)\big) = 0$, for each $i \in N$.*

Proof. Using (3.11) and the triangle inequality,

$$\begin{aligned} U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^j) - U_i(\boldsymbol{\mu}^j) &= \left|U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^j) - U_i(\boldsymbol{\mu}^j)\right| \\ &\leq \left|U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^j) - U_i(\boldsymbol{\mu}^\star)\right| + \left|U_i(\boldsymbol{\mu}^\star) - U_i(\boldsymbol{\mu}^j)\right|. \end{aligned}$$

As $j \to \infty$, the first summand goes to zero by (3.17) and the second by the assumption. Hence the conclusion. □

In our numerical experiments (see Section 3.3.6), we compute the difference

$$U_i(x_i^{j+1}, \boldsymbol{\mu}_{-i}^j) - U_i(\boldsymbol{\mu}^j) \tag{3.18}$$

at each step j and check if such differences are diminishing with j increasing. This provides a simple heuristics to detect the quality of approximation and convergence. Another option is to calculate the Wasserstein distance

$$d_W(\boldsymbol{\mu}^j, \boldsymbol{\mu}^{j+1}), \tag{3.19}$$

which can be done with a linear program (3.9). If the sequence $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots$ converges weakly, then $d_W(\boldsymbol{\mu}^j, \boldsymbol{\mu}^{j+1}) \to 0$. We include the values (3.19) in the results of some numerical experiments and observe that they are decreasing to zero quickly. Figure 3.9 shows, that neither (3.18) nor (3.19) are monotone sequences.

MO is a meta-algorithm parameterized by

1. the algorithm for computing equilibria of subgames (the *master problem*) and
2. the optimization method for computing the best response (the *oracles*).

We detail this setup for each example in the next section.

The choice of computational methods should reflect the properties of a continuous game, since the efficiency of methods for solving the master problem and subproblem is the decisive factor for the overall performance and precision of MO. For example, polymatrix games are solvable in polynomial time [13], whereas finding even an approximate Nash equilibrium of a finite general-sum game is a very hard problem [19]. As for the solution of the oracles, the best response computation can be based on global solvers for special classes of utility functions.

3.3.4. Master Problem Implementations

Finding NE in zero-sum finite games reduces to either the classical linear program, or its aforementioned extension by Cai et al. [13] in case of polymatrix games. The implementation of the master problem corresponding to general-sum matrix sub-games is more complex. Various methods exist to solve general-sum games, including the global Newton method, simplical subdivision, or even straightforward strategy enumeration. These methods, however, may have unpredictable runtimes and do not always guarantee convergence.

In our experiments on general-sum matrix games, we use the path following algorithm implemented by the *gambit-logit* utility which is a part of the Gambit library, along with a version of a multilinear formulation of NE (3.20) described by Sturmfels [87] and improved by Fischer and Gupte [27]. We also use the classical linear programming implementation where appropriate.

Continuing with the established notation, let $w_i \in \mathbb{R}$ be the bound on player i's utility, and let $\boldsymbol{\mu}$ be probability simplices defining the mixed strategies of players. The multilinear feasibility program in $\mathbf{w}$ and $\boldsymbol{\mu}$ for finding NE in general-sum games is

$$\begin{aligned} \sum_{i \in N} U_i(\boldsymbol{\mu}) &\geq \sum_{i \in N} w_i \\ U_i(x, \boldsymbol{\mu}_{-i}) &\leq w_i && \forall i \in N, x \in X_i \\ \sum_{x \in X_i} \mu_i(x) &= 1 && \forall i \in N \\ 0 \leq \mu_i(x) &\leq 1 && \forall i \in N, x \in X_i \end{aligned} \tag{3.20}$$

As the utility functions are defined by tensors and strategy spaces are finite, the definitions of expected utility simplify into weighted sums. Specifically, the best response function

$U_i(x, \boldsymbol{\mu}_{-i})$ reduces to the vector

$$\sum_{\mathbf{x}_{-i} \in \mathbf{X}_{-i}} U_i[x, \mathbf{x}_{-i}] \prod_{j \in N_{-i}} \mu_j[x_j];$$

while the expected utility $U_i(\boldsymbol{\mu})$ becomes

$$\sum_{\mathbf{x} \in \mathbf{X}} U_i[\mathbf{x}] \prod_{i \in N} \mu_i[x_i].$$

As mentioned in Section 2.2, algebraic functions can be decomposed into binary operations for which there exist known tight convex relaxations. Specifically, the multilinear program 3.20 can be reformulated as a non-convex quadratically-constrained quadratic program, which can then be solved by spatial Branch and Bound solvers.

An advantage of the multilinear program, is that it can be modified to search for specific solutions. Constraining the search to symmetric equilibria can be done trivially by adding variables μ_0 and equating $\mu_0 = \mu_i \; \forall i \in N$. We use this modification to exploit the symmetry of Blotto games and speed up the search in Experiment 5. Additionally, as 3.20 is a feasibility program, it can be trivially modified to search for equilibria with the smallest ℓ_0 norm.

Formulation 3.20 worked acceptably well with *Gurobi*, and frequently outperformed other methods. To demonstrate the relative performance of different master-problem implementations, we first compared the runtimes between the bilinear formulation and the *gambit-logit* algorithm on two-player general-sum matrix games with normally distributed payoffs. Each algorithm was executed 100 times per game size with a 20-second time limit. The measured runtimes include only the time attributable to the solvers without any modelling overhead. Fig. 3.2 demonstrates that we could not replicate the consistency achieved by Fischer and Gupte [27]. Although the bilinear program exhibits lower median runtimes (shown as darkest plot regions) compared to *gambit-logit* with default settings, its runtime has a multimodal distribution with a significantly higher standard deviation (5.15 s vs. 0.22 s for 30×30 games) leading to frequent timeouts. The *gambit-logit* program always finished successfully, while the success rate of the bilinear formulation was steadily decreasing, completing 89% of the 30-action games and only 59% of 40-action games. Mean runtimes on 20-action games were: 0.81 s (bilinear) and 0.18 s (*gambit-logit*)[1].

In our experience, matrices of normally-distributed values tend to be poor anologues for the subgames generated by MO. To provide an alternative benchmark, we discretized the strategy spaces of two different test functions. Payoff matrices were generated by evaluating the Michalewicz function for Player 1 and McCormick function for Player 2 on a Cartesian product of randomly sampled actions uniformly distributed over $[0, \pi]$, i.e the strategy space of each player for a game of size k is $X_i^{(k)} \overset{\text{i.i.d.}}{\sim} \mathcal{U}([0, \pi])^k$, and the utilities are:

$$u_1(x_1, x_2) = -\sum_{i=1}^{2} \sin(x_i) \sin^{2m}\left(\frac{i x_i^2}{\pi}\right)$$

$$u_2(x_1, x_2) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - \frac{3x_1 + 5x_2 + 2}{2}$$

[1] We omit statistics such as the standard deviation, as they can be misleading due to the multimodal runtimes of some solvers.

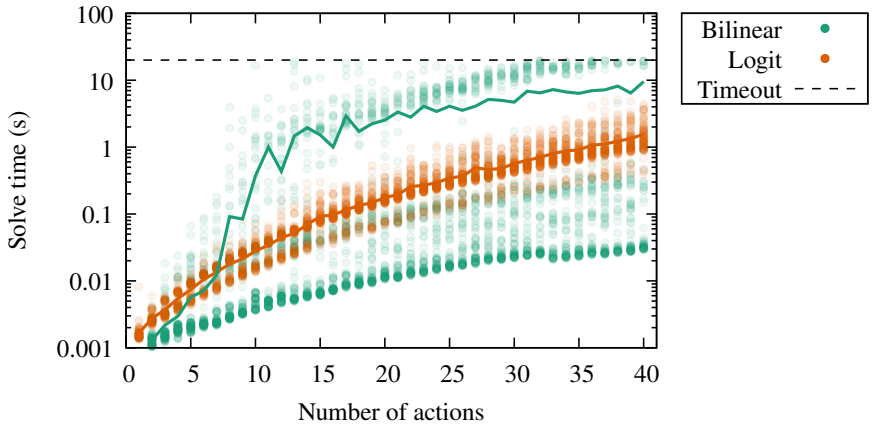


Figure 3.2.: Runtime comparison between the bilinear formulation and the *gambit-logit* algorithm on two-player general-sum matrix games with normally distributed payoffs. Mean runtimes of successful runs are overlaid as thick lines.

The results are shown in Fig. 3.3, where the multilinear formulation was more consistent; however, in actual runs of the algorithm, the master-problem still occasionally times out when impelemnted in this way. It appears that consistent performance relies on optimizations specific to the *Baron* solver, which we could not run sufficiently extensive benchmarks due to a lack of licence.

Finally, solvers may be able to take advantage of the structure of a game formulated as the multilinear program. Fig. 3.4 shows that *Gurobi* is significantly faster on zero-sum compared to general-sum games. The performance of the *gambit-logit* utility, on the other hand, is consistent across our tests. Each algorithm was executed 100 times per game size with a 20-second time limit. For 20-action games, mean runtimes were: 0.6 ms (bilinear program), 0.02 ms (linear program), and 159.7 ms (*gambit-logit*).

In our testing, the *Gambit logit* utility was the fastest on average and most reliable method for solving finite general-sum games, which agrees with the benchmarks done by Fischer and Gupte [27], where it was the second fastest method.

3.3.5. Oracle Implementations

We show a comparison of oracle implementations. The following table shows the runtime attributable to the oracle computation for various examples and solvers including: *Couenne*, *Gurobi*, *Gurobi* with *GRBaddgenconstrPoly* applicable to univariate polynomials, the Sum-of-squares hierarchy using *Mosek*, *SCIP*, and the *PolyJuMP* KKT optimizer using *HomotopyContinuations*. The examples are split into groups of polynomial games over intervals, polynomial games over semialgebraic sets, and games with general utilities based on standard optimization test functions.

All solvers were executed sequentially in each iteration of MO, using the value returned by *Gurobi* as the reference solution, allowing us to verify that all global solvers agree on the

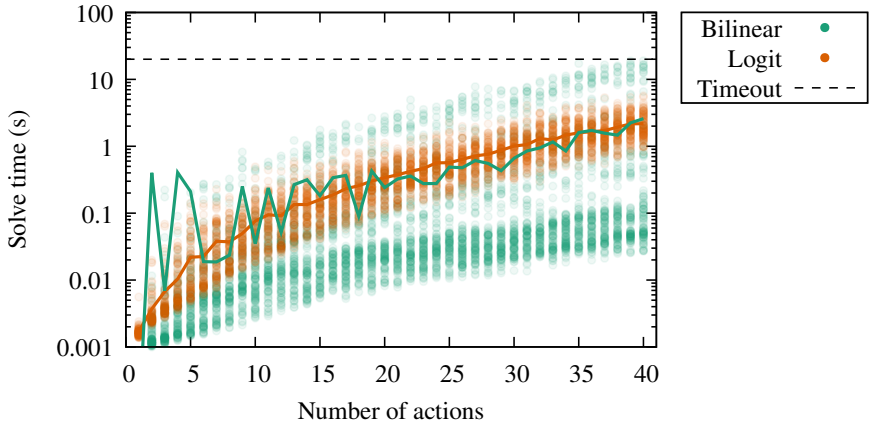


Figure 3.3.: Runtime comparison between the bilinear formulation and the *gambit-logit* algorithm on two-player general-sum matrix games generated by sampling test functions. Solid lines represent mean runtimes.

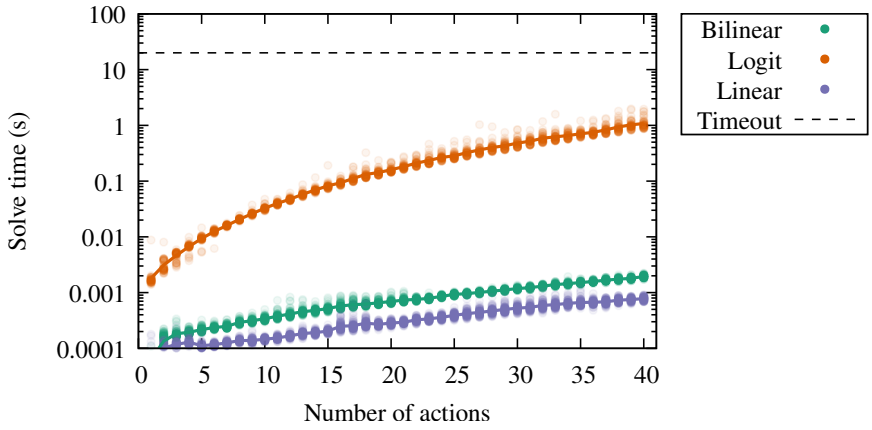


Figure 3.4.: Runtime comparison of the bilinear program, linear program, and the *gambit-logit* algorithm for solving two-player zero-sum matrix games with normally distributed payoffs. Thick lines indicate mean runtimes of successful executions.

best-response values. The oracle problem was passed via *AMPL* to all global solvers with the exception of *Gurobi*, which accepts nonlinear input from *JuMP* natively. The following tests were executed on *AMD Ryzen 5 4600G* and the listed times for each example are the sums of runtimes reported by the solvers in milliseconds.

Example	Couenne	Gurobi (Poly)		Ipopt	SOS	SCIP	KKT
A.3.18	409.5	1.4	11.4	146.5	6.2	67.0	24.4
A.3.19	587.2	2.1	15.7	205.7	5.4	114.1	28.0
A.3.20	819.5	4.1	26.6	287.4	9.6	152.1	39.4
A.3.21	404.3	1.3	8.2	142.0	3.6	71.9	19.7
A.3.22	1211.6	72.0	14.9	260.3	9.5	177.7	43.3
A.3.24	473.5	1.9	8.1	175.6	2.5	80.4	16.0
A.2.7	347.4	24.4		122.7	26.3	65.8	635.3
A.2.8	487.6	2.0		178.3	12.4	79.9	1707.5
A.2.11	84.6	7.4		28.5		15.0	
A.4.1		22.5				199.4	
A.4.2	273.3	10.1				45.5	
A.4.3		90.5				319.6	

The results of this brief test supports our choice to use *Gurobi* as the default oracle. The hierarchies are competitive on polynomial problems, while the FOSS solver *SCIP* is the only other solver robust and generic enough to solve all the listed examples. This result is consitent with our experience. *Ipopt* returned values deviating by a norm of less than 10^{-5} across all iterations for three univariate polynomial examples and one multivariate example, but none of the test functions. While we have succesfully used local-solvers as oracles, it is important to remember that convergence to equilibria is no longer guaranteed.

The runtime of MO may be improved by adding multiple best responses in each iteration. In the case of the SOS hierarchies, solvers tend to find high-rank interior solutions and computing multiple solutions may be essentially free.

3.3.6. The Runtime of Multiple Oracle

We show the performance that can be expected from MO on various important examples from the literature. We focus on examples which demonstrate interesting behaviors, as well as special classes of games for which efficient oracles and master-problems exist. We will detail the setup individually for each example. In Chapter B, we follow with a more comprehensive evaluation of a general implementation of the algorithm based on nonlinear solvers and finite general-sum games.

The algorithm is initialized by selecting an arbitrary point found heuristically by solvers[2]. The algorithm then continues with standard MO iterations: by computing an equilibrium of the finite subgame and selecting best responses.

We plot the progress of the largest possible gain from a profitable deviation (3.18, ϵ) for each player at each iteration, along with the Wasserstein distance (3.19) between mixed

[2]We experimented with initialization with interior points of players' strategy spaces and with best responses to such points, but did not observe a consistently faster runtime.

strategies from consecutive iterations. Neither sequence is monotonic, and both often attain zero, for example, when only some of players profit from switching strategies.

Example 5 (General Blotto)**.** The utility functions in *General Blotto* games are sums of non-linearly weighted margins of victory over sets of fronts. We chose a game with three isolated fronts defined by the utility $u_1(x_1, x_2) = \sum_{j=1}^{m} f\left(x_{1j} - x_{2j}\right)$, with the weighing function $f(z) = \operatorname{sgn}(z)\,|z|^p$. When $p = 0.5$, the utility function is non-differentiable and the game converges very slowly as shown in Fig. 3.5. Large gains are possible by winning a front by a small margin as the slope of the utility function around zero approaches infinity. After 140 iterations, the algorithm recovers the ϵ-NE with $\epsilon = 0.006$ shown in Fig. 3.6.

Multilinear terms, square roots and sums can be formulated directly via the generic modelling languages *JuMP*, and *AMPL*; however the sign function and absolute value are more challenging. Due to a lack of support and differentiability, all of the solvers: *Ipopt*, *Bonmin*, *Couenne*, *SCIP*, and *Gurobi* were unable to solve a direct formulation of the oracle problem. Instead, we formulated the oracle as the following mixed-integer quadratic program:

$$
\begin{aligned}
\max_{b,z^+,z^-,x,u} \quad & \sum_{j=1}^{m} w_j \sum_{i=1}^{3} (1 - 2b_{ij}) u_{ij} \\
b_{ij} &\in \{0,1\} && \forall i,j \\
u_{ij}, x_{ij}, z^+_{ij}, z^-_{ij} &\in [0,1] && \forall i,j \\
z^+_{ij} &\le b_{ij} \\
z^-_{ij} &\le 1 - b_{ij} \\
x_{ij} - c_{ij} &= z^+_{ij} - z^-_{ij} \\
u^2_{ij} &= z^+_{ij} + z^-_{ij}
\end{aligned}
$$

where c_{ij} is the allocation of the opponent to the ith front in the jth action out of m in the mixed strategy of the opponent weighted by w_j.

The game is symmetric, allowing us to speed up the algorithm by restricting the search to symmetric equilibria. The master-problem was implemented using the bilinear formulation (3.20) with the additional constraint that all strategies μ_i be equal. Only one oracle computation per iteration was necessary due to the symmetry. The runtime to reach 140 iterations was still around 8 minutes.

Example 6 (Tangent Ridge Game A.3.33)**.** Consider a two-player general-sum game on $[0,1]$, with utility functions illustrated in Fig. 3.7:

$$
\begin{aligned}
u_1(x_1, x_2) &= -\sqrt{|x_1 - x_2|} \\
u_2(x_1, x_2) &= -\sqrt{|x_1 - \tan(x_2)|}
\end{aligned}
$$

Let $X_1, X_2 \subset [0,1]$, and $x_1^\star, x_2^\star = \min X_1, \min X_2$. Every finite subgame with strategy sets X_1 and X_2, has the pure NE $(x_1^\star, x_2^\star)$ if $\tan(x_2^\star) \ge x_1^\star \ge x_2^\star$.

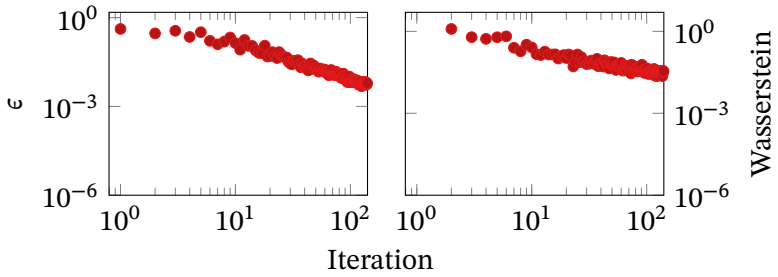


Figure 3.5.: Slow convergence of a continuous but non-differentiable *General Blotto* game valuing 3 isolated fronts with $p = 0.5$. Both players can still profitably deviate by 0.006 utility units after 140 iterations.

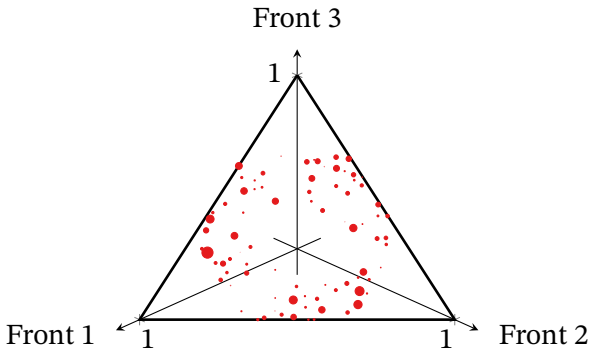


Figure 3.6.: The recovered symmetric strategy after 140 iterations drawn as circles on a simplex, with weight indicated by their size.

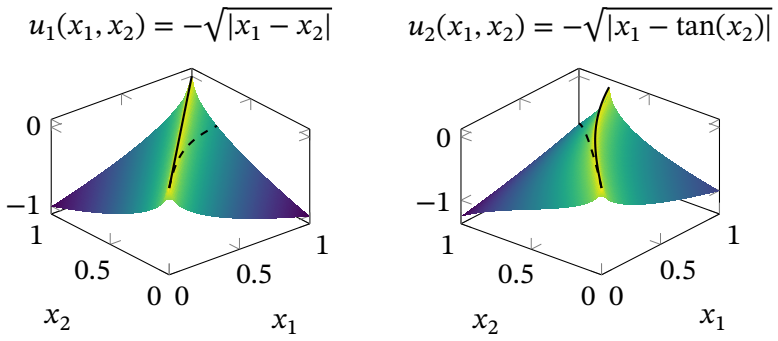


Figure 3.7.: Utility functions of both players in Example 6. The projected best response curves $\beta_1(x_2) = x_2$ and $\beta_2(x_1) = \arctan(x_1)$ show similar behavior near the origin, where they intersect, resulting in a unique pure NE.

Proof. 1. For all $x_1 \in X_1$ we have $x_1 \geq x_1^\star$, hence

$$|x_1 - x_2^\star| = x_1 - x_2^\star \geq x_1^\star - x_2^\star = |x_1^\star - x_2^\star|.$$

2. The function tan is strictly increasing on $[0, 1]$, and $\tan(x_2^\star) \geq x_1^\star$ by assumption. Hence, for all $x_2 \in X_2$ we get $\tan(x_2) \geq \tan(x_2^\star) \geq x_1^\star$ and

$$|x_1^\star - \tan(x_2)| = -x_1^\star + \tan(x_2) \geq -x_1^\star + \tan(x_2^\star) = |x_1^\star - \tan(x_2^\star)|.$$

In the utilities, the distances are mapped through the decreasing function $-\sqrt{\cdot}$ which inverts the relations. Therefore, $u_1(x_1, x_2^\star) \leq u_1(x_1^\star, x_2^\star)$ for all $x_1 \in X_1$; and $u_2(x_1^\star, x_2) \leq u_2(x_1^\star, x_2^\star)$ for all $x_2 \in X_2$. □

It follows that if MO is initialized with a single action for both players, all generated subgames will have pure equilibria. The master-problem can be implemented by placing all weight on the minimal actions, while the oracles are given in closed form by $\beta_1(x_2) = x_2$ and $\beta_2(x_1) = \arctan(x_1)$.

Suppose the algorithm is started with a point outside $x_1 > x_2 > \arctan(x_1)$, e.g. $(1, 1)$. Then the pure equilibria of the generated subgames fall alternately on one of the best response curves. The players take turns expanding their supports with successive iterates of $\arctan^{(i)}(1)$, which converges to the unique pure NE at the origin along a trajectory of slope approximately one. As a result, progress slows significantly in later iterations, mirroring the behavior observed in the *General Blotto* game. MO takes 14424 iterations to reach an $\epsilon = 10^{-3}$ NE, when started from the point $(1, 1)$ as shown in Fig. 3.8.

Example 7 (Game A.3.19)**.** Consider a two-player zero-sum game with strategy sets $[-1, 1]$ and a utility function of the first player

$$u(x, y) = 2xy^2 - x^2 - y.$$

As the generated subgames are all finite two-player zero-sum games, we can use linear programming to find their equilibria. The best response oracles were implemented using Lasserre hierarchies.

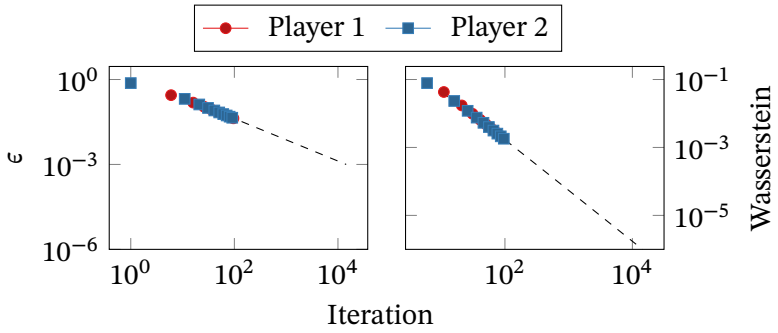


Figure 3.8.: A log-log plot of the sublinear $O(x^{-3/2})$ convergence of Example 6 in both the largest possible deviation by each player, as well as the Wasserstein distance between strategies from successive iterations.

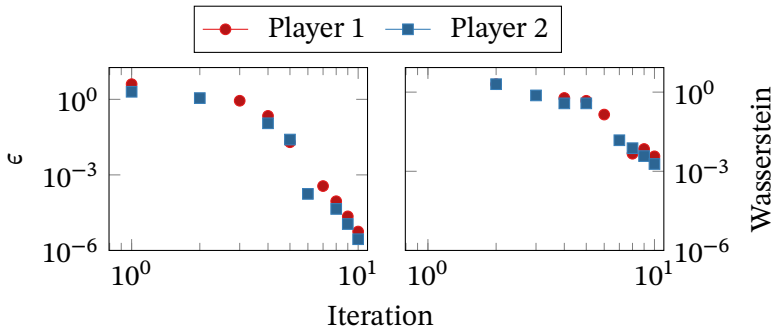


Figure 3.9.: Convergence in Game A.3.19. After 10 iterations, our method finds pure strategies $x \approx 0.4$ and $y \approx 0.63$, resulting in payoffs $(-0.47, 0.47)$.

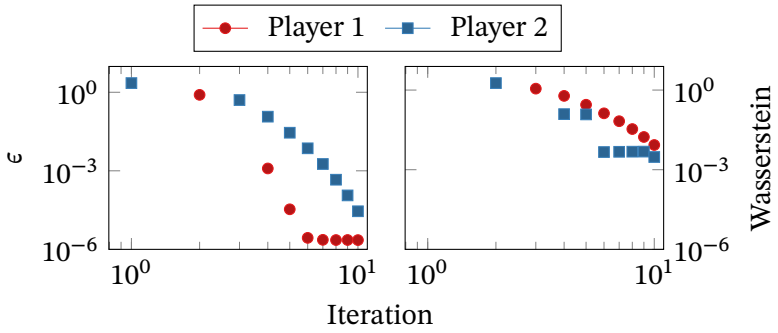


Figure 3.10.: Convergence in Game A.3.22. Our method finds mixed strategies where $x \approx 0.11$ with probability 44.19% and $x \approx -1$ with probability 55.81% and $y \approx 0.72$ resulting in payoffs $(1.13, 1.81)$.

Example 8 (Game A.3.22)**.** The strategy set of each player is $[-1, 1]$ and utility functions are

$$u_1(x, y) = -3x^2y^2 - 2x^3 + 3y^3 + 2xy - x,$$
$$u_2(x, y) = 2x^2y^2 + x^2y - 4y^3 - x^2 + 4y.$$

We computed equilibria in the general-sum matrix subgames using the multilinear program 3.20. The best response oracles were implemented using Lasserre hierarchies.

Example 9 (Random polymatrix games)**.** Separable network games (polymatrix games) with finitely many strategies of each player can be solved in polynomial time [13] by linear programming. We use MO to compute equilibria of five-player polymatrix games defined by 20 by 20 matrices. Specifically, we generated a random matrix for each edge in the network and then transposed and subtracted the matrix of utility functions to make the game globally zero sum. We remark that zero-sum polymatrix games are payoff-equivalent to the general polymatrix games by [13, Theorem 7]. In a test of 100 games, our algorithm found an ϵ-equilibrium with $\epsilon = 0.01$ after 17.69 iterations in a mean time of 0.29 seconds.

Example 10 (Separable Network Games)**.** We can also solve the polynomial network games A.3.28 and A.3.29, which were analyzed with the tools of polynomial optimization earlier in Section 3.2.3. We show their convergence in Figs. 3.11 and 3.12. The master problem was solved using the LP technique by Cai et al. [13, Linear Program 1] with *Mosek*, while the oracles were implemented using spatial branch and bound using *Gurobi*.

The experimental run over 10 iterations of Game A.3.28 is shown in Fig. 3.11. The algorithm converged in 0.15 seconds and 8 iterations to the ϵ-NE:

$$\mu_1^\star \approx \delta_{-0.06},$$
$$\mu_2^\star \approx 0.573\delta_{0.34} + 0.427\delta_{0.37},$$
$$\mu_3^\star \approx 0.722\delta_1 + 0.278\delta_{-1}.$$

We also experimented with oracles based on the SOS hierarchy, which resulted in a slightly slower runtime of 0.29 seconds.

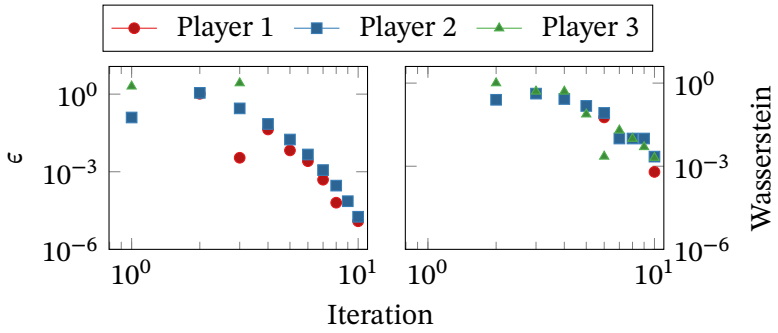


Figure 3.11.: Convergence of Game A.3.28 showing the largest possible deviation by each player, as well as the wasserstein distance between strategies from successive iterations.

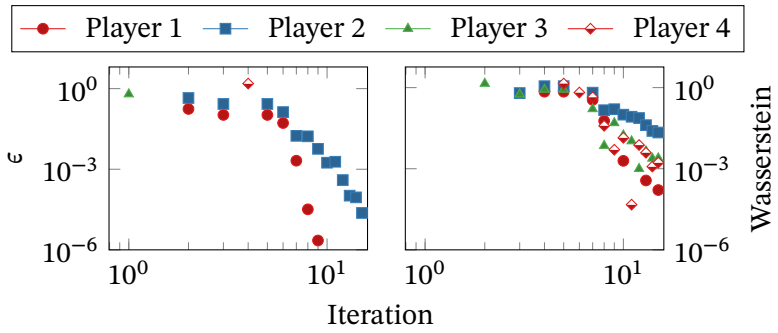


Figure 3.12.: Convergence of Game A.3.29 over 15 iterations.

The experiment on Game A.3.29 is shown in Fig. 3.12. The algorithm converged already after 0.5 s and 12 iterations to:

$$\begin{aligned}
\mu_1^\star &\approx \delta_{(0.29,0,0,0.21)},\\
\mu_2^\star &\approx 0.494\delta_{(0.34,0.22,0,0.24)} + 0.303\delta_{(0.35,0,0.2,0.25)} + 0.204\delta_{(0.43,0.06,0,0.3)},\\
\mu_3^\star &\approx 0.69\delta_{(0,1)} + 0.31\delta_{(1,0)},\\
\mu_4^\star &\approx 0.758\delta_{(1,0)} + 0.242\delta_{(0,1)}.
\end{aligned}$$

Using oracles based on the SOS hierarchy, we found the same equilibrium in 1.2 seconds.

4. Optimal Inverse Kinematics

We show how to compute globally optimal solutions to inverse kinematics by formulating the problem as a non-convex quadratically constrained quadratic program. Our approach makes solving inverse kinematics instances of generic redundant manipulators feasible. We demonstrate the performance on randomly generated designs and real-world robots with up to ten revolute joints. The same technique can be used for manipulator design by introducing kinematic parameters as variables.

4.1. Background

Inverse kinematics (IK) is the problem of finding control parameters that bring a robot into a desired pose within its kinematic constraints. For manipulators of six or fewer joints, this problem is practically solvable and has at most 16 different solutions [54]. Adding additional joints results in an under-constrained problem with infinitely many feasible solutions satisfying the pose constraints. Therefore, it is interesting not simply to look for feasible solutions but optimal ones. In this paper, we focus on minimizing the total movement of joints. Specifically, the task is as follows: Given the Denavit-Hartenberg parameters [83] of a kinematic chain, input joint angles, and a desired pose for its end effector, find joint angles that achieve the desired pose while minimizing the weighted sum of differences from the input (preferred) angles.

The presented problem is very hard in general. The current state-of-the-art technique is based on *sum of squares* (SOS) optimization and Gröbner bases, which are both NP-hard in the context of multivariate polynomial equations. Trutman *et al.* demonstrate [90] that for manipulators with at most seven joints, the optimal solutions admit a quadratic SOS representation, thereby being practically solvable. Such results do not extend to manipulators with more than seven joints.

A common way to avoid the complexity of IK is either to find an approximate solution or to design the robotic arm to be simple to control. Approaches from the first category include an approximation based on the Jacobian by Buss [11], which uses numerical methods to find local optima. Similarly, Dai *et al.* [17] designed a global solution method for the convex approximation of the original IK. On the other hand, the design of the KUKA LBR iiwa manipulator falls into the second category. However, having to design around computational constraints has the downside of limiting the achievable performance of a manipulator.

The problem may be further relaxed for real-time control by dropping the guarantees on optimality or convergence to the desired pose. For example, iKinSlv finds a locally optimal solution which minimizes the distance from the desired pose using interior point optimization; the closed-loop inverse kinematics algorithm suggested in the Pinocchio library

attempts to iteratively reduce the pose error, while modern approaches [79] based on convex *quadratic programming* (QP) rely on asymptotic convergence.

Global solvers recover a globally optimal solution whenever it exists or guarantee that none exists. In the context of mechanical design, it is possible to optimize a design by computing and comparing the actual limits of performance in some desired metric specified by the user. Using the optimality guarantees to test existing solvers' properties is also possible. If a solver fails to provide a solution, it is possible to answer whether the instance was solvable.

The contribution of this paper is based on the premise that while SOS optimization is, in some sense, the correct answer[1] to the *polynomial optimization problem* (POP), incumbent implementations fall significantly short of their promised potential. We will show that highly optimized off-the-shelf global optimizers can converge faster for problems such as IK despite having much higher computational complexity. Instead of reformulating IK into a theoretically easier problem, we reformulate it into a problem for which efficient solvers exist.

Our approach finds globally optimal solutions of generic seven *degree-of-freedom* (DOF) IK instances faster than the state-of-the-art. Our technique is unique in finding globally optimal IK solutions for generic manipulators with eight or more degrees of freedom.

4.2. Quadratic Program Formulation

Starting with the POP for globally solving IK recently proposed by Trutman *et al.* [90], we lift it into a *quadratically constrained quadratic program* (QCQP). Specifically, we find a new basis in lifting variables of the feasible set of the initial POP, such that the new higher-dimensional representation is at most quadratically constrained. For example, for a polynomial constraint $x_1x_2x_3 = 0$ we introduce a variable y and rewrite the constraint as $y = x_1x_2$ and $yx_3 = 0$. Solvers for non-convex QCQPs, such as SCIP [7] or Gurobi [35], can find the global solution to this new problem by the spatial branch-and-bound method.

4.2.1. Denavit–Hartenberg parameters

It is conventional to describe n-joint manipulators in terms of their *Denavit–Hartenberg* (D-H) parameters for each link $i = 1, \dots, n$: the offset d_i, the link length r_i, the link twist α_i, and the joint angle variable θ_i.
The transformation between successive coordinate frames from link i to $i-1$ is defined by a D-H matrix

$$T_i(\theta_i) = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & r_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & r_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The end effector pose w.r.t. the base coordinate system is represented by the matrix P, which induces the *pose constraint* $T_1(\theta_1) \cdots T_n(\theta_n) = P$.

[1] Boaz Barak and David Steurer argue that SOS may be an *optimal algorithm* at sumofsquares.org.

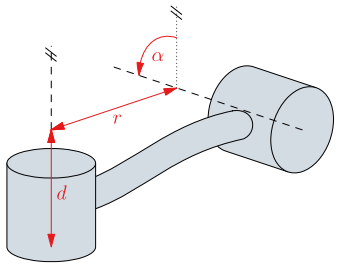


Figure 4.1.: Diagram of Denavit–Hartenberg parameters

4.2.2. Polynomial Formulation

In this section we describe the POP for IK initially formulated by Trutman *et al.* [90]. This formulation avoids trigonometric functions in the D-H matrices. For each link $i = 1, \dots, n$ introduce variables $c_i := \cos\theta_i$ and $s_i := \sin\theta_i$, along with the identity $c_i^2 + s_i^2 = 1$. After the change of variables, every D-H matrix $T_i(\theta_i)$ depends only on c_i and s_i, and we denote is simply by T_i. We assume that each joint angle θ_i is constrained by the maximal value $\theta_i^{max} \in [0, \pi]$ and the minimal value $\theta_i^{min} \in [-\pi, 0]$. The vector of preferred joint angles is $(\hat{\theta}_1, \dots, \hat{\theta}_n)$ and we consider some weights $w_1, \dots, w_n \geq 0$ satisfying $w_1 + \cdots + w_n = 1$. The goal is to minimize the weighted distance

$$\sum_{i=1}^{n} w_i\big((c_i - \cos\hat{\theta}_i)^2 + (s_i - \sin\hat{\theta}_i)^2\big). \tag{4.1}$$

The choice of preferred joint angles $(\hat{\theta}_1, \dots, \hat{\theta}_n)$ makes it possible to minimize the total movement of joints, or to find configurations where some links achieve desirable orientations (e.g. an upright posture in humanoid robots [69]).

Further, we split the kinematic chain with n joints into two subchains that meet in the middle[2]: the first anchored at the origin 1 and ending at ν and the second starting at $\nu+1$ and ending at n. The pose constraint $T_1(\theta_1)\cdots T_n(\theta_n) = P$ then becomes $T_1 \cdots T_\nu = P T_n^{-1} \cdots T_{\nu+1}^{-1}$. This formulation enables us to lower the degree of polynomials appearing in the above constraint and, consequently, to reduce the number of lifting variables (see Section 4.2.3).

The resulting POP (see [90, (9)]) with variables $\mathbf{c} = (c_1, \dots, c_n)$, $\mathbf{s} = (s_1, \dots, s_n)$, the objective

[2]The placement of the split does not significantly affect the runtime as long as each subchain involves at least two joints.

function (4.1), and the constraints formulated above is:

$$\min_{\mathbf{c},\mathbf{s}} \quad \sum_{i=1}^{n} 2w_i(1 - c_i \cos\hat{\theta}_i - s_i \sin\hat{\theta}_i) \tag{4.2}$$

$$\text{s.t.} \quad PT_n^{-1} \cdots T_{\nu+1}^{-1} = T_1 \cdots T_\nu \tag{4.3}$$

$$(c_i + 1) \tan\frac{\theta_i^{min}}{2} \leq s_i \qquad i = 1, \ldots, n \tag{4.4}$$

$$(c_i + 1) \tan\frac{\theta_i^{max}}{2} \geq s_i \qquad i = 1, \ldots, n \tag{4.5}$$

$$c_i^2 + s_i^2 = 1 \qquad i = 1, \ldots, n \tag{4.6}$$

Since (4.1) is equal to the linear function in (4.2), this POP is in fact a minimization problem with a linear objective, and non-convex quadratic 4.6 and multilinear 4.3 constraints. Linear inequalities (4.4)-(4.5) are equivalent to design constraints $\theta_i^{min} \leq \theta_i \leq \theta_i^{max}$ by basic trigonometric identities and the convention $0 \leq \theta_i^{max} \leq \pi$, $-\pi \leq \theta_i^{min} \leq 0$. We use $\theta_i = \text{atan2}(s_i, c_i)$ to recover the optimal joint angles θ_i.

4.2.3. Reformulating Multivariate Polynomials as Quadratics

The SOS-based method by Trutman et al. [90] solves a non-convex POP equivalent to (4.2)–(4.6) using Lasserre's hierarchy presented in Section 2.1; However, quartic systems in 14 variables are challenging already for the method, and obtaining solutions is often predicated on the use of costly algebraic preprocessing using Groebner Basis computation. Reformulating the problem such that it can be solved by Spatial Branch and Bound obviates the need for such preprocessing.

The bounding step in Branch and Bound is predicated on being able to construct convex relaxations of the constraints. We will achieve this by reformulating the multilinear constraints 4.3 as non-convex quadratics, for which the appropriate convex relaxations are McCormick envelopes.

We rewrite each polynomial constraint of the POP into a set of equivalent quadratic constraints in a higher dimensional space. That is, we rewrite the monomials of each polynomial constraint by repeatedly substituting the multiplication of two optimization variables with a new lifting variable equal to their product. For example, the monomial $x_1x_2x_3 \cdots x_n$ could be lifted as $y_{12}x_3 \cdots x_n$, where $y_{12} = x_1x_2$. Each application of such a rewriting reduces the degree of the polynomial and can be applied repeatedly to each monomial of the original expression until only single variables remain. Since the lifting variables are quadratically constrained, we effectively transform the non-convex POP (4.2)–(4.6) into a non-convex QCQP. Such a straight-forward lifting technique was already discussed by Shor [81].

Due to the structure of the pose constraint, we can apply the same rewriting to the expressions involving D-H matrices directly. Since the result of T_iT_j will contain at most quadratic terms, we can introduce a matrix of lifting variables for every matrix product and proceed as described above.

The number of variables and constraints in the resulting program depends on the order of application of such a rewriting rule. For example, the constraint $x_1x_2x_3x_4 + x_2x_3x_5 = 0$ can

be reformulated in at least two different ways:

$$\begin{aligned} 0 &= y_{12}y_{34} + y_{23}x_5 \\ y_{12} &= x_1x_2 \\ y_{34} &= x_3x_4 \\ y_{23} &= x_2x_3 \end{aligned} \tag{A}$$

$$\begin{aligned} 0 &= y_{123}x_4 + y_{23}x_5 \\ y_{23} &= x_2x_3 \\ y_{123} &= x_1y_{23} \end{aligned} \tag{B}$$

Taking advantage of repeated terms, and even the order in which products are lifted, has an observable impact on the speed and behavior of solvers and on the cuts that they apply.

In the next section, we will focus on the performance of the method corresponding to System A and the matrix based method. We will refer to these methods as lift_A and lift_M, respectively. While lift_M often results in the fewest variables after presolve, it is not always the fastest. We found the behavior of lift_A more predictable. The full implementation of this approach is available in our public code repository.

4.2.4. Applicability of QCQP Lifting

The lifting algorithm presented in Section 4.2.3 applies to any IK problem involving polynomials since it needs no additional assumptions about the degree of polynomials or their properties. This enlarges the scope of applicability of our model significantly. For example, instead of the distance from preferred angles (4.1), we can optimize *manipulability*. We can also use the QCQP lifting for the *design of manipulators* by introducing new variables in place of the parameters of the manipulator. This is based on the observation that even if all the parameters in the D-H matrices are treated as optimization variables, the resulting IK problem would still be a POP. Since polynomials are closed under addition and multiplication, the resulting constraints could be lifted as described in Section 4.2.3.

4.3. Results

We compare our QCQP-based procedure to the prior SOS-based method Trutman et al. [90]. Our numerical experiments use an *Intel Xeon Gold 6146* processor limited to 4 threads. Implementing the SOS-based method involves Maple 2022 and Mosek 10, while our method uses Gurobi 10. These solvers are the latest and fastest for each problem class available to us at the time of writing. Results for the SOS-based method are based on the original implementation and use all the techniques described in [90] to improve numerical stability. We show the statistics of solve times estimated from randomly generated feasible and infeasible poses of the KUKA LBR iiwa 7-DOF manipulator, the *iCub* [66] humanoid robot's right arm and torso (7–10 DOF), and on randomly generated designs with seven joints. Unless stated otherwise, the results in following sections pertain to the lifting approach lift_A (see Section 4.2.3).

4.3.1. Factors Influencing the Solution Time

As we solve IK via a non-convex QCQP, it is difficult to predict the solution time of a given instance; however, in our experience, the time increases with the number of non-zeros in

the constraint matrix, the number of variables, and the volume of the space that needs to be searched. In practical terms, the runtime depends on the manipulator's range of motion and the link twists. Specifically, designs with axes offset by 90°, as is the case for the *iCub* and the *KUKA LBR iiwa*, are easier to solve than ones with joints at arbitrary angles.

To demonstrate how the time required to solve IK depends on the aspects of mechanical design, we will consider three sets of randomly generated robotic designs with seven joints:

Orth. (Orthogonal) designs with link twists sampled uniformly from $\{-\pi/2, \pi/2\}$, resulting in orthogonal axes and a 6-radian range of motion $-3 \leq \theta_i \leq 3$,

6-rad designs with arbitrary link twists α_i uniformly sampled from $[-3, 3]$ and 6-radian range of motion,

4-rad designs with arbitrary twists α_i uniformly sampled from $[-3, 3]$, and 4-radian range $-2 \leq \theta_i \leq 2$.

The link offsets and lengths are generated randomly according to a uniform distribution between 10 and 100 centimeters.

As mentioned before in Section 2.2, the size of tree that the Spatial Branch and Bound method has to search depends on the quality of feasible solutions found. While there is nothing preventing a solver from running barrier methods transparently on the background, it may be easier for us to find at least an initial feasible solution. In our case specifically, the lifting approach introduces a large number of auxiliary variables and constraints which increases the problem size (Table 4.2) and could make it more difficult to solve. We found that warm-starting can speed up the global solution search even with low-precision local solutions. We used *IPopt* [92] and the straightforward non-linear program 4.7 to find an initial guess:

$$\begin{aligned} \min_{\theta} \quad & \sum_{i=1}^{n} 2w_i(1 - \cos\theta_i \cos\,\hat{\theta}_i - \sin\theta_i \sin\hat{\theta}_i) \\ \text{s.t.} \quad & T_1 \cdots T_\nu = P T_n^{-1} \cdots T_{\nu+1}^{-1} \\ & \theta_i^{min} \leq \theta_i \leq \theta_i^{max} \qquad\qquad i = 1, \ldots, n \end{aligned} \tag{4.7}$$

This warm-starting may fail to provide a solution, as local solvers may get stuck at points of local infeasibility. We limited the local-solution search to 200 iterations (approximately 200 ms). Unless indicated otherwise, all of the following benchmarks used warm-starting.

Switching to modern QP or iterative real-time techniques with higher precision and lower failure rates will likely improve solution times. Additionally, such techniques could be used during the runtime as branch-and-bound heuristics.

Note that the variable and constraint count alone are not a good predictor of runtime. Despite not allowing for cancellation lift_M results in the fewest variables after presolve; however, while the method is faster on some designs on average, its behavior is less predictable. In extreme cases, this resulted in runtimes almost 100 times higher than the average.

Table 4.1.: QCQP runtime, the positional and rotational error of the final solution. “Cold” is a cold-start without a local solution. The lift_M formulation solved the orthogonal instances in 1.23 s, *4-rad* in 1.93 s, but *6-rad* in 8.96 s on average.

Set	**Time (s)**	**Err. (μm)**	**Err. (μrad)**
Orth.	1.31	2.29	1.28
(cold) Orth.	2.12	2.71	1.28
4-rad	2.36	2.20	1.09
(cold) 4-rad	3.90	2.39	1.26
6-rad	4.91	2.88	1.70
(cold) 6-rad	12.20	2.51	1.15

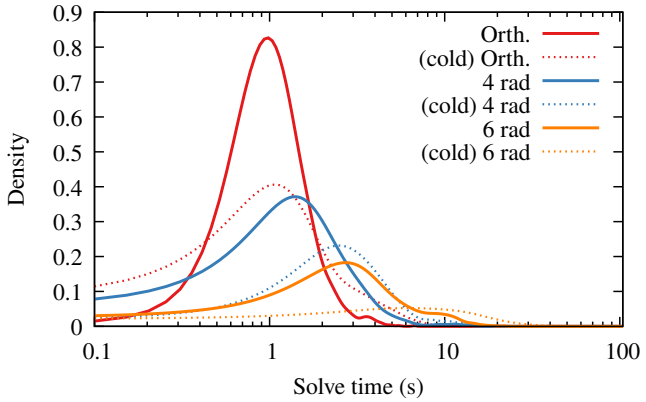


Figure 4.2.: A semi-log plot of solve-time kernel-density-estimates from 1000 samples of randomly generated designs for each parameter set.

Table 4.2.: DOF and the resulting problem size after presolve for each lifting method

		lift_M		lift_A	
Set	**DOF**	**vars.**	**constrs.**	**vars.**	**constrs.**
KUKA	7	71	57	76	62
Orth.	7	78	64	80	66
6-rad	7	118	104	119	105
iCub	7	78	62	78	64
iCub	8	95	79	102	86
iCub	9	136	118	146	128
iCub	10	179	159	190	170

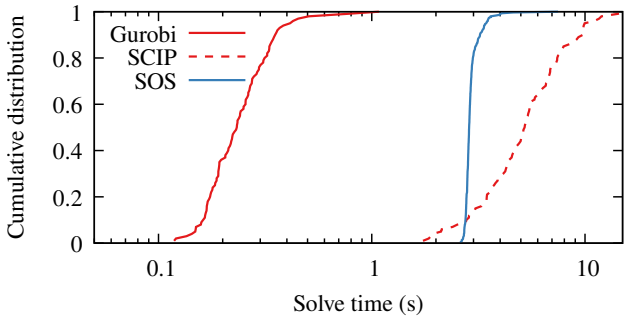


Figure 4.3.: A semi-log plot comparing the normalized cumulative distributions from 100 solve times for the *KUKA LBR iiwa* 7-DOF robot using our QCQP approach (Gurobi, SCIP) and the SOS-based technique.

4.3.2. Comparison Across Solvers and Methods (7 DOF)

Our approach outperforms the SOS-based method [90] when tested on feasible IK instances generated by the *KUKA LBR iiwa* 7-DOF manipulator. The state-of-the-art method was explicitly optimized to take advantage of the manipulator's structure and eliminate one of the joints. Due to Maple's license availability, the following comparison was performed on four threads of *AMD Ryzen 5 4600G*. We did not include the modelling and Matlab overhead in the reported runtimes. The average solve-time (Fig. 4.3) of our technique was 0.26 s, compared to 2.9 s for the optimized SOS approach. We have also tried the free and open-source solver *SCIP* instead of *Gurobi*. However, the average solve time for *SCIP* rises to 5.7 s. While a performance decrease is expected when comparing commercial solvers to free and open-source ones, in this case *SCIP* could also not effectively utilize more than one thread and did not benefit from the local solution provided by warm-starting. The prior method can take advantage of the structure of the manipulator, which significantly improves the runtime compared to generic manipulators.

To compare the methods on randomly generated designs, we rounded the lengths and offsets of the *6-rad* test-set to two significant digits, as the performance of the SOS-based method depends on the number of digits in the problem. Such rounding is unnecessary with our method, but it does not significantly affect the runtime. The average solve-time (Fig. 4.4) of our technique was 12.0 s with *Gurobi* and 931 s with *SCIP*. The modified SOS approach took 4328 s to solve an instance on average. Finally, we note that the prior SOS-based method has roughly a 1% failure rate, a problem we did not encounter with our method.

The symbolic reduction step of the SOS-based method can be a bottleneck for some robots. For example, the 7-DOF IK instances of *iCub*'s right arm (Fig. 4.5) did not consistently finish within 5 hours of runtime even despite rounding to the nearest millimeter (3 significant figures). In this case however, we found that the reduction is not necessary, and *Mosek* could solve the SOS programs directly with only a 1 % failure rate (this was described as the *naïve* approach in [90]). The average solve-time over 1000 samples was 0.17 s for *Gurobi*, 4.5 s for *SCIP*, and 3.3 s for the *naïve* SOS.

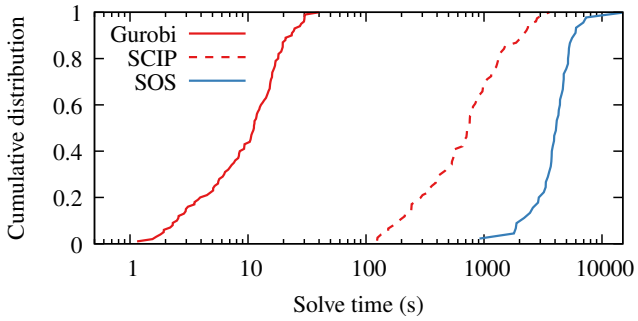


Figure 4.4.: A semi-log plot comparing the normalized cumulative distributions of solve times obtained from 100 samples of randomly generated designs with 6-radian range of motion using our approach and the SOS-based technique (rounding to two significant digits).

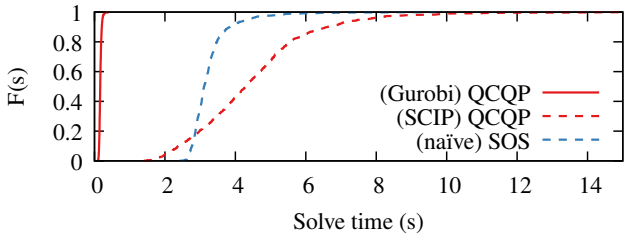


Figure 4.5.: Normalized cumulative distributions $F(s)$ of solve times from 1000 samples of the *iCub's* 7-DOF right arm. The full SOS-based method with symbolic reduction did not finish in time.

Table 4.3.: The QCQP runtime statistics (mean, first and third quartile) and the mean positional and rotational errors of the final solution computed from 1000 IK instances of iCub (7–10 DOF).

Set	Time Avg. (s)	Time Q1 (s)	Time Q3 (s)	Loc. Err. Avg. (μm)	Rot. Err. Avg. (μrad)
7 DOF	0.2	0.1	0.2	0.2	1.0
8 DOF	0.8	0.6	0.9	0.3	1.4
9 DOF	6.5	2.8	9.8	0.5	2.1
10 DOF	77.6	28.0	90.4	0.8	2.9

Finally, we tested the SOS-based method without symbolic reduction on Canadarm2. While the average runtime was 6.3 s, the success rate was only 31% out of 1000 feasible instances. The average runtime for QCQP with *Gurobi* was 1.8 s.

4.3.3. *iCub* Right Arm and Torso (7–10 DOF)

We also demonstrate (Table 4.3) that our technique scales to realistic instances with up to 10 degrees of freedom. Due to the low range of motion of the iCub's arms, it can be beneficial to include the torso in the kinematic chain. We start with the 7-DOF right arm and progressively add torso joints up to the full 10-DOF chain. We are aware of no suitable baseline for this class of problems.

We also checked the scalability by running the 10-DOF instances on 126 threads of *AMD EPYC 7543* processor, which resulted in a mean runtime of 10.1 s over 100 samples.

5. Future Research

We are interested in continuing the research of finding globally optimal solutions over compact spaces in both game theory and in robotics.

Game Theory

The convergence of our algorithm for separable network games was experimentally verified on selected examples. However, to prove that the hierarchy in fact converges is an interesting open problem for further research. While running the experiments, we noticed the computed payoffs were generally of high quality regardless of the solver used; The corresponding measures, on the other hand, were less accurate. For example, the strategies recovered in Example 4 yields a loss of only 0.16 instead of 0.17 for one of the players. This appears to be a problem inherent to our approach rather than something that could be fine-tuned within the solvers.

The idea of MO is to construct a sequence of finite subgames whose equilibria approximate the equilibrium of a given continuous game in the Wasserstein metric. Note that MO makes it possible to approximate the equilibrium of *any* continuous games in the sense of Theorems 6 and 7, with the caveat that an individual sequence of equilibria may fail to converge. Although possible in theory [1], this behavior has never been observed in the sample games.

For MO to be practical, the master-problem must be significantly easier to solve than the original game. We assume that the number of iterations will be small, so that only a small number of games with small action spaces need to be solved. While we show in Appendix B that MO converges quickly on many examples from the literature, establishing a general convergence rate for any class of continuous games remains an open problem. Notably, even for the considerably simpler case of the Double Oracle algorithm applied to finite games, convergence analysis remains an active area of research [25].

Inverse Kinematics

Our approach to reformulating the polynomial formulation in Section 4.2 was very direct. While we have experimented briefly with solvers capable of polynomial optimization by different techniques, such as *RAPOSa*, *Baron* and *Couenne*, this has not resulted in runtime improvements. Still, the structure of the problem should enable us to create constraints which could significantly speed up the solution process. For example, the feasible set of the multilinear kinematic constraints should be a polytope. While its full representation may be too costly to construct, finding a few of its facets near the optimum could be feasible. Similarly, the results of Trutman et al. [90] indicate that the constraints have a lower-degree

representation even without auxiliary variables. Whether this extends to higher degrees of freedom is still an open question.

Our method currently seems most appropriate for the use-cases of manipulator design, where determining the actual performance of manipulators in some metric specified by the user is likely to be preferable to a heuristic value. Moreover, the runtime requirements are not so strict in such contexts. Although our method does not apply to real-time control, additional pre-processing and appropriate heuristics may make such a use case possible. There already exist appropriate methods in the literature. For example, we could use modern real-time IK solvers based on convex QP as runtime heuristics or generate initial feasible solutions to warm-start the search. While running the experiments, we observed that a significant fraction of the solution time for *Gurobi* was spent proving that the solution was optimal. Using the outer approximation technique described by Dai, Izatt, and Tedrake [17] could make this stage faster. Finally, it could be possible to use the current configuration of a manipulator to construct a feasible solution using homotopy methods. To make use of such homotopies would likely involve reformulating the kinematic constraint using quaternions.

In practice, we are also able to significantly accelerate the proof-of-concept technique of Trutman et al. [90]. Their Grobner basis computation is implemented directly in *Maple*. Including a cheap pre-processing step to find the required number of polynomials (and moving to *Msolve*) is a simple way to optimize the slowest step of the algorithm.

A. List of Infinite Games

While there exist datasets, such as GAMUT, useful for testing algorithms designed for finite games, we found no established projects for continuous games. To address this, we present a compilation of parametrized games drawn from both recent and classical literature, which should provide a convincing and comprehensive evaluation of any game-solving algorithm.

Note that continuous games are inherently complex, as even apparently simple games may only have Nash equilibria supported on infinitely many points. A case in point are Games A.3.2 and A.3.5 played on the $[0, 1]$ interval.

We chose to consistently present the utility function of player i denoted as u_i in the variables x_{ij}, where the first digit i denotes the player, while j is the variable index. None of the games that follow have more than ten players nor variables for any of the players.

A.1. General Blotto

The following examples are various settings for the classical two-player zero-sum game known as *General Blotto*. The utility functions are sums of non-linearly weighted margins of victory over fronts. Golman and Page [32] completely characterized the existence of pure equilibria for the following family of weighing functions:

$$f(z) = \operatorname{sgn}(z)\,|z|^p$$

parametrized by p, which is strictly positive for continuous games. The margins of victory are differences of products of the allocations to each front.

Game A.1.1. A two-player zero-sum game in $m \times m$ variables on the simplex

$$x_i \in \Delta^{m-1} = \left\{ z \in \mathbb{R}^m \;\middle|\; z_j \geq 0,\ \sum_{j=1}^{m} z_j = 1 \right\} \quad \forall i \in \{1, 2\}.$$

The utility function is defined by

$$u_1(x_1, x_2) = \sum_{j=1}^{m} f\left(x_{1j} - x_{2j}\right)$$

where $f(x) = \operatorname{sign}(x)\,|x|^p$ and $p > 0$. This is a continuous version of *General Blotto*, where all isolated fronts have equal value. When $p = 1$, every strategy is optimal; when $p > 1$, players may allocate all resources to any one front; no pure equilibrium exists for p less than one [32].

Game A.1.2. A two-player zero-sum game in $m \times m$ variables on the simplex

$$x_i \in \Delta^{m-1} \quad \forall i \in \{1, 2\}.$$

The utility function is defined by

$$u_1(x_1, x_2) = \sum_{i=1}^{m} \sum_{j=i+1}^{m} f\left(x_{1i}x_{1j} - x_{2i}x_{2j}\right) + \sum_{i=1}^{m} f\left(x_{1i} - x_{2i}\right)$$

where $f(x) = \text{sign}(x)\,|x|^p$ and $p > 0$. This is a continuous version of *General Blotto*, where all pairs of fronts have equal value. When $p = 1$, players should allocate $\frac{1}{m}$ to all fronts; otherwise, no pure equilibrium exists [32].

Game A.1.3. A two-player zero-sum game in $m \times m$ variables on the simplex

$$x_i \in \Delta^{m-1} \quad \forall i \in \{1, 2\}.$$

The utility function is defined by

$$u_1(x_1, x_2) = \sum_{\substack{I \subseteq \{1,\dots,m\} \\ I \neq \emptyset}} f\left(\prod_{i \in I} x_{1i} - \prod_{i \in I} x_{2i}\right)$$

where $f(x) = \text{sign}(x)\,|x|^p$ and $p > 0$. This is a continuous version of *General Blotto*, where all sets of fronts have equal value. When $p = 1$ players should allocate $\frac{1}{m}$ to all fronts. No pure equilibrium exists for other values of p [32].

A.2. Polynomial Saddle Point Games

The following examples are the saddle point problems of polynomials studied by Nie, Yang, and Zhou [62], which we have reinterpreted as two-player zero-sum polynomial games.

Game A.2.1. A zero-sum 2-player polynomial game in 3×3 variables on the simplex

$$x_i \in \Delta^2 \quad \forall i \in \{1, 2\}.$$

The nonconcave-nonconvex utility function [62, Example 6.1 (i)] is

$$u_1(x_1, x_2) = -x_{11}x_{12} - x_{11}x_{23} - x_{12}x_{13} - x_{13}x_{21} - x_{21}x_{22} - x_{22}x_{23}$$

This game has the pure NE

$$x_1^\star \approx (0, 1, 0),\ x_2^\star \approx (0.25, 0.5, 0.25).$$

Game A.2.2. A zero-sum 2-player polynomial game in 3×3 variables on the simplex

$$x_i \in \Delta^2 \quad \forall i \in \{1, 2\}.$$

The nonconcave-nonconvex utility function [62, Example 6.1 (ii)] is

$$\begin{aligned} u_1(x_1, x_2) = &- x_{11}^3 - x_{12}^3 + x_{13}^3 + x_{21}^3 + x_{22}^3 - x_{23}^3 \\ &- x_{13}x_{21}x_{22}(x_{21} + x_{22}) - x_{12}x_{21}x_{23}(x_{21} + x_{23}) - x_{11}x_{22}x_{23}(x_{22} + x_{23}) \end{aligned}$$

This game has the pure NE

$$x_1^\star \approx (0, 0, 1),\ x_2^\star \approx (0, 0, 1).$$

Game A.2.3. A zero-sum 2-player polynomial game in 4×4 variables on the simplex

$$x_i \in \Delta^3 \quad \forall i \in \{1, 2\}.$$

The nonconcave-nonconvex utility function [62, Example 6.1 (iii)] is

$$u_1(x_1, x_2) = \sum_{i=1}^{4} \sum_{\substack{j=1 \\ i \neq j}}^{4} (x_{1i}x_{1j} + x_{2i}x_{2j}) - \sum_{i=1}^{4} x_{1i}^2 x_{2i}^2$$

This game has 4 pure NE involving each one-hot vector e_i

$$\text{NE}_i : \ x_1^\star \approx (0.25, 0.25, 0.25, 0.25),\ x_2^\star = e_i \quad \forall i \in \{1, 2, 3, 4\}.$$

Game A.2.4. A zero-sum 2-player polynomial game in 3×3 variables on the simplex

$$x_i \in \Delta^2 \quad \forall i \in \{1, 2\}.$$

the nonconcave-nonconvex utility function [62, Example 6.1 (iv)] is

$$u_1(x_1, x_2) = x_{23}^2 x_{11}^2 - x_{11}x_{12}x_{21}x_{22} - x_{11}x_{13}x_{21}x_{23} + x_{21}^2 x_{12}^2 - x_{12}x_{13}x_{22}x_{23} + x_{22}^2 x_{13}^2$$

The utility function has no saddle point.

Game A.2.5. A zero-sum 2-player game in 2×2 variables on the unit hypercube

$$x_i \in [0, 1]^2 \quad \forall i \in \{1, 2\}$$

with the concave-nonconvex utility function

$$u_1(x_1, x_2) = -(1 + x_{11} + x_{12} + x_{21} + x_{22})^2 + 4(x_{11} + x_{22} + x_{11}x_{12} + x_{12}x_{21} + x_{21}x_{22})$$

Nie, Yang, and Zhou [62, Example 6.2 (i)] report the pure NE

$$x_1^\star \approx (0.3249, 0.3249),\ x_2^\star = (1, 0).$$

Similarly, we recover a pure ϵ-NE with any feasible $x_{11} = x_{12}$ and $x_2 = (0, 1)$.

Game A.2.6. A zero-sum 2-player game in 3×3 variables on the unit hypercube

$$x_i \in [0,1]^3 \quad \forall i \in \{1,2\}$$

with the nonconcave-nonconvex utility function [62, Example 6.2 (ii)]

$$\begin{aligned} u_1(x_1,x_2) = &- x_{11} - x_{12} - x_{13} - x_{21} - x_{22} - x_{23} - x_{22}^2 x_{11}^2 \\ &- x_{23}^2 x_{11}^2 + x_{21}^2 x_{12}^2 - x_{23}^2 x_{12}^2 + x_{21}^2 x_{13}^2 + x_{22}^2 x_{13}^2 \end{aligned}$$

The utility function has no saddle point.

Game A.2.7. A zero-sum 2-player game in 3×3 variables on the hypercube

$$x_i \in [-1,1]^3 \quad \forall i \in \{1,2\}$$

with the nonconcave-nonconvex utility function [62, Example 6.3 (i)]

$$u_1(x_1,x_2) = -\sum_{i=1}^{3}(x_{1i} + x_{2i}) + \prod_{i=1}^{3}(x_{1i} - x_{2i})$$

This game has three pure Nash equilibria:

$$\begin{aligned} \text{NE}_1 &: x_1^\star = (-1,-1,1), x_2^\star = (1,1,1), \\ \text{NE}_2 &: x_1^\star = (-1,1,-1), x_2^\star = (1,1,1), \\ \text{NE}_3 &: x_1^\star = (1,-1,-1), x_2^\star = (1,1,1). \end{aligned}$$

Game A.2.8. A zero-sum 2-player game in 3×3 variables on the hypercube

$$x_i \in [-1,1]^3 \quad \forall i \in \{1,2\}$$

with the nonconcave-nonconvex utility function [62, Example 6.3 (ii)]

$$u_1(x_1,x_2) = -\sum_{i=1}^{3} x_{2i}^2 + \sum_{i=1}^{3} x_{1i}^2 - \sum_{i=1}^{3}\sum_{\substack{j=1 \\ i<j}}^{3}(x_{1i}x_{2j} - x_{1j}x_{2i})$$

This game has the following pure Nash equilibrium:

$$x_1^\star = (-1,1,-1), x_2^\star = (-1,1,-1).$$

Game A.2.9. A zero-sum 2-player game in 3×3 variables on the sphere

$$x_i \in \left\{z \in \mathbb{R}^3 \,\middle|\, \|z\| = 1\right\} \quad \forall i \in \{1,2\}$$

with the utility function [62, Example 6.4 (i)]

$$\begin{aligned} u_1(x_1,x_2) = &- x_{11}^3 - x_{12}^3 - x_{13}^3 - x_{21}^3 - x_{22}^3 - x_{23}^3 \\ &- 2(x_{11}x_{12}x_{21}x_{22} + x_{11}x_{13}x_{21}x_{23} + x_{12}x_{13}x_{22}x_{23}) \end{aligned}$$

This game has the following 9 pure NE:

$$\text{NE}_{ij} : \; x_1^\star = e_i, \; x_2^\star = e_j \quad \forall i,j \in \{1,2,3\}.$$

Game A.2.10. A zero-sum 2-player game in 3×3 variables on the sphere

$$x_i \in \left\{z \in \mathbb{R}^3 \,|\, \|z\| = 1\right\} \quad \forall i \in \{1,2\}$$

with the utility function [62, Example 6.4 (ii)]

$$\begin{aligned} u_1(x_1,x_2) = &- x_{11}^2 x_{21}^2 - x_{12}^2 x_{22}^2 - x_{13}^2 x_{23}^2 - x_{11}^2 x_{22} x_{23} - x_{12}^2 x_{21} x_{23} \\ &- x_{13}^2 x_{21} x_{22} - x_{21}^2 x_{12} x_{13} - x_{22}^2 x_{11} x_{13} - x_{23}^2 x_{11} x_{12} \end{aligned}$$

The utility function has no saddle points.

Game A.2.11. A zero-sum 2-player game in 3×3 variables on the ball

$$x_i \in \left\{z \in \mathbb{R}^3 \,|\, \|z\| \leq 1\right\} \quad \forall i \in \{1,2\}$$

with the nonconcave-convex utility function [62, Example 6.5]

$$u_1(x_1,x_2) = -x_{11}^2 x_{21} - 2x_{12}^2 x_{22} - 3x_{13}^2 x_{23} + x_{11} + x_{12} + x_{13}$$

This game has the following pure Nash equilibrium:

$$\mathrm{NE}_3 : x_1^\star \approx (0.7264, 0.4576, 0.3492), x_2^\star \approx (0.6883, 0.5463, 0.4772).$$

Game A.2.12. A zero-sum 2-player game in 3×3 variables on the intersection of the nonnegative orthant and the sphere

$$x_i \in \left\{z \in \mathbb{R}^3_+ \,|\, \|z\| = 1\right\} \quad \forall i \in \{1,2\}$$

with the utility function [62, Example 6.6]

$$u_1(x_1,x_2) = -x_{11}^2 x_{22} x_{23} - x_{21}^2 x_{12} x_{13} - x_{12}^2 x_{21} x_{23} - x_{22}^2 x_{11} x_{13} - x_{13}^2 x_{21} x_{22} - x_{23}^2 x_{11} x_{12}$$

The utility function has no saddle points.

Game A.2.13. A zero-sum 2-player game in 5×5 variables on the simplex

$$x_i \in \Delta^4 \quad \forall i \in \{1,2\}.$$

The utility function is defined by [62, Example 6.10]

$$u_1(x_1,x_2) = x_1^\mathsf{T} A_1 x_1 + x_2^\mathsf{T} A_2 x_2 + x_1^\mathsf{T} B x_2$$

where the matrix parameters A_1, A_2, and B are

$$A_1 = \begin{bmatrix} -4 & 4 & 0 & 3 & -4 \\ 3 & 4 & 3 & -4 & -5 \\ -3 & 0 & -2 & 0 & 4 \\ -4 & -4 & -1 & 3 & -5 \\ 4 & 1 & -3 & 0 & -5 \end{bmatrix}, A_2 = \begin{bmatrix} -4 & 4 & 1 & 0 & 1 \\ -2 & -4 & 2 & -3 & 1 \\ -3 & 1 & 1 & 4 & 4 \\ 3 & -4 & 0 & 1 & -2 \\ -1 & -3 & -1 & 3 & -2 \end{bmatrix}$$

$$B = \begin{bmatrix} -2 & -4 & -2 & -5 & 3 \\ 0 & 0 & 2 & 4 & 2 \\ 0 & -4 & -1 & -5 & 3 \\ 1 & -3 & -4 & 0 & -3 \\ 3 & -1 & -5 & 4 & -4 \end{bmatrix}$$

The game has the following two NE:

$$\begin{aligned} \text{NE}_1 &: x_1^\star = (0,1,0,0,0), x_2^\star = (1,0,0,0,0), \\ \text{NE}_2 &: x_1^\star = (0,1,0,0,0), x_2^\star = (0,1,0,0,0). \end{aligned}$$

A.3. Miscellaneous Games

The following is a list of example games appearing in papers and books by various authors, which we loosely grouped by the research topic.

A.3.1. Games from Classical Literature

The following games appear in books from the 1950s and 60s, where they were used as demonstrations that the solutions of continuous games may be supported on anything from singleton sets to Cantor sets and whole intervals.

Game A.3.1. A two-player zero-sum game in 1×1 variables on the interval

$$x_{i1} \in [0,1] \quad \forall i \in \{1,2\},$$

with the utility function [23, pg. 110]

$$u_1(x_1, x_2) = (x_{11} - x_{21})^2$$

The Nash equilibrium is $\mu_1^\star = \mathcal{U}\{0,1\}$; $\mu_2^\star = \delta_{0.5}$ and the value is $\frac{1}{4}$.

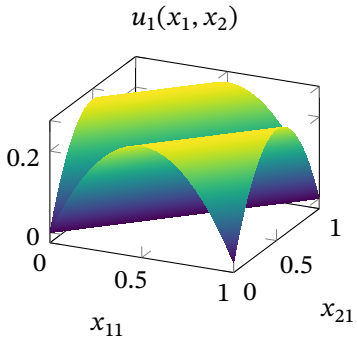
Game A.3.2. A two-player zero-sum game in 1×1 variables on the interval

$$x_{i1} \in [0,1] \quad \forall i \in \{1,2\},$$

with the utility function

$$u_1(x_1, x_2) = (1 - |x_{21} - x_{11}|)\,|x_{21} - x_{11}|$$

Dresher [23, pg. 111] proved that the only equilibrium of this game has an infinite support: $\mu_i^\star = \mathcal{U}[0,1]$, and its value is $\frac{1}{6}$.



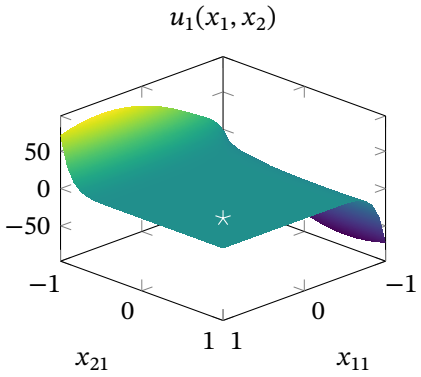


Figure A.1.: 12-degree polynomial game on a hypercube with a prescribed solution at 0.5, 0.5 constructed using the method described by Gale and Gross [28] with parameters $\alpha_i = (1, 0)$, $\sigma_i = (0.5)$, and $\mu_i = (1) \quad \forall i \in \{1, 2\}$.

Game A.3.3. A two-player zero-sum game in $m_1 \times m_2$ variables on compact subsets of real numbers

$$x_i \in X_i \subset \mathbb{R}^{m_i} \quad \forall i \in \{1, 2\}.$$

The polynomial utility function is defined by

$$\begin{aligned} u(x_1, x_2) =& f(x_1)\phi(x_1)\left(g(x_2)\phi(x_1) + \sum_{i=1}^{c_2}(g_i(x_2) - \mu_{2i})\phi_i(x_1)\right) \\ &- g(x_2)\psi(x_2)\left(f(x_1)\psi(x_2) + \sum_{i=1}^{c_1}(f_i(x_1) - \mu_{1i})\psi_i(x_2)\right) \\ &- (f(x_1)\phi(x_1))^2 + (g(x_2)\psi(x_2))^2 \end{aligned}$$

where

$$f(x_1) = \prod_{s\in\sigma_1} \|x_1 - s\|^2, \qquad \phi(x_1) = \prod_{a\in\alpha_1} \|x_1 - a\|^2,$$

$$f_i(x_1) = \prod_{\substack{s\in\sigma_1 \\ s\neq\sigma_{1i}}} \frac{\|z - s\|^2}{\|\sigma_{1i} - s\|^2}, \qquad \phi_i(x_1) = \prod_{\substack{a\in\alpha_1 \\ a\neq\alpha_{1i}}} \|x_1 - a\|^2;$$

with g and ψ being defined analogously for x_2, σ_2, μ_2, and α_2.

This game has the property that the parameters σ and μ define a unique mixed Nash equilibrium supported on $\sigma_i \in X_i^{c_i}$ weighted by $\mu_i \in \Delta^{c_i-1}$, for any set of cluster points $\alpha_i \in X_i^{c_{-i}+1}$ that do not overlap with the supports.

This construction of polynomial games with a unique prescribed solution was discovered by Gale and Gross [28]. The generated games may be challenging to solve numerically given their large degrees and relative flatness near the equilibrium as shown in figure A.1.

Game A.3.4. A two-player zero-sum game in 1×1 variables on the interval

$$x_{i1} \in [0,1] \quad \forall i \in \{1,2\}.$$

The utility function is defined by

$$u_1(x_1, x_2) = \frac{1}{1 + \lambda (x_{11} - x_{21})^2}$$

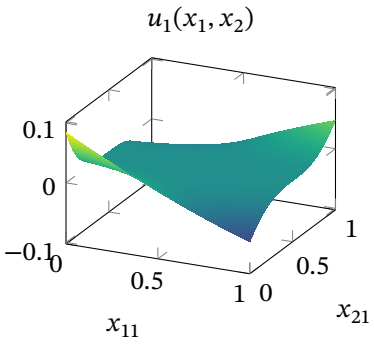
where $\lambda > 0$. Karlin [41, Example 1] proved that the game has finitely supported optimal strategies.

Game A.3.5. A two-player zero-sum game in 1×1 variables on the interval

$$x_{i1} \in [0,1] \quad \forall i \in \{1,2\}.$$

The utility is defined by the rational function

$$u_1(x_1, x_2) = (x_{21} - 0.5)\left(\frac{1 + (x_{21} - 0.5)^2 (x_{11} - 0.5)}{1 + (x_{21} - 0.5)^4 (x_{11} - 0.5)^2}\right) - \frac{x_{21} - 0.5}{1 + (x_{21} - 0.5)^4 \left(\frac{1}{3} x_{11} - 0.5\right)}$$



Karlin [41, Example 2] proved that the unique optimal strategy is the Cantor distribution.

Game A.3.6. A two-player zero-sum game in 1×1 variables on the interval

$$x_{i1} \in [0,1] \quad \forall i \in \{1,2\}.$$

The utility is defined by [41, Problem 1]:

$$u_1(x_1, x_2) = \frac{2(x_{11} + x_{21})}{(1 + 2x_{11})(1 + 2x_{21})}$$

The game has a Nash equilibrium where both players play $\mu^\star(z) = \frac{1+2z}{2}$.

Game A.3.7. A two-player zero-sum game in 1×1 variables on the interval

$$x_{i1} \in [0,1] \quad \forall i \in \{1,2\}.$$

The utility is defined by

$$u_1(x_1, x_2) = \frac{(1 + x_{11})(1 + x_{21})}{(1 + x_{11} x_{21})^2}$$

Karlin [41, Problem 2] showed that the value of this game is $\frac{1}{\ln 2}$ when both players play $\mu^\star(z) = \frac{\ln(1+z)}{\ln 2}$.

A.3.2. Experiments on Games with Interesting Weaker Solution Concepts

The following examples are from papers that study algorithms recovering weaker solution concepts, such as min-max critical points, due to the potential non-existence and the complexity of computing Nash Equilibria. Examples ... were used to demonstrate the failure of convergence of gradient-based algorithms.

Game A.3.8. A two-player zero-sum game in 1×1 variables on the interval

$$x_{i1} \in [0,1] \quad \forall i \in \{1,2\},$$

with the utility function

$$u_1(x_1, x_2) = (-0.5 + x_{11})(-0.5 + x_{21}).$$

This game was studied by Daskalakis et al. [20, Appendix D]. The game has a value of 0 and a pure Nash equilibrium at $(x_{11}, x_{21}) = (0.5, 0.5)$.

Game A.3.9. A two-player zero-sum game in 1×1 variables on the interval

$$x_{i1} \in [0,1] \quad \forall i \in \{1,2\},$$

with the utility function [20, Appendix E]

$$u_1(x_1, x_2) = \left(-4x_{11}^2 + \frac{x_{21}^4}{10} + \left(-3x_{11} + x_{21} + \frac{x_{11}^3}{20}\right)^2\right) e^{\frac{1}{100}\left(-x_{11}^2 - x_{21}^2\right)}$$

The only local min-max equilibrium is $(x_{11}, x_{21}) = (0,0)$. This game was also studied by Chinchilla, Yang, and Hespanha [16], Wang, Zhang, and Ba [93], and Zheng et al. [98].

Game A.3.10. A two-player zero-sum game in 1×1 variables on the interval

$$x_{i1} \in [0,1] \quad \forall i \in \{1,2\}.$$

The utility function is defined by

$$u_1(x_1, x_2) = -\left(x_{11} - \frac{1}{2}\right)\left(x_{21} - \frac{1}{2}\right) - \frac{1}{3} e^{-\left(x_{11} - \frac{1}{4}\right)^2 - \left(x_{21} - \frac{3}{4}\right)^2}$$

Mertikopoulos et al. [58, Fig. 1] demonstrated divergent behavior of *Mirror Descent* on this non-monotone saddle-point problem. The game has an ϵ-NE:

$$\mu_1^\star \approx \delta_{0.38},$$
$$\mu_2^\star \approx 0.433\delta_0 + 0.567\delta_1$$

achieving a value of 0.247.

Game A.3.11. A two-player zero-sum game in 1×1 variables on the interval

$$x_{i1} \in [-1, 1] \quad \forall i \in \{1, 2\}.$$

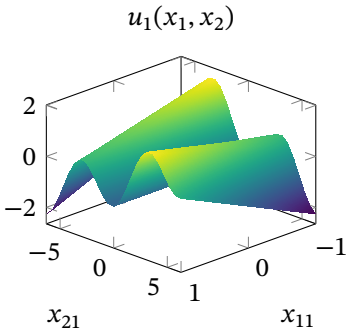
The utility function is

$$u_1(x_1, x_2) = \left(1 - x_{21}^2 + x_{21}^4 x_{11}^2\right)\left(-1 - x_{11}^2 - x_{21}^2 x_{11}^4\right)$$

This non-monotone saddle-point problem was studied by Mertikopoulos et al. [58, (2.2)]. The unique saddle point is $x^\star = (0, 0)$.

Game A.3.12. A two-player zero-sum game in 1×1 variables on the intervals

$$x_1 \in [-1, 1]$$
$$x_2 \in [-2\pi, 2\pi]$$



with the utility function [74, Example 1]

$$u_1(x_1, x_2) = -\cos(x_{21}) + 0.2 x_{11} x_{21}$$

The game has the ϵ-NE:

$$\mu_1^\star = 0.5\delta_{-1} + 0.5\delta_1,$$
$$\mu_2^\star = 0.5\delta_{-2\pi} + 0.5\delta_{2\pi}.$$

Game A.3.13. A two-player zero-sum game in 1×1 variables on the interval

$$x_i \in [-1, 1] \quad \forall i \in \{1, 2\}$$

with the utility function [74, Example 3]

$$u_1(x_1, x_2) = 2x_{11}x_{21} - x_{21}^2 + x_{11}^3$$

The game has the mixed ϵ-NE:

$$\mu_1^\star \approx \frac{1}{3}\delta_1 + \frac{2}{3}\delta_{-0.5},$$
$$\mu_2^\star \approx \frac{5}{16}\delta_1 + \frac{11}{16}\delta_{-1}.$$

Game A.3.14. A two-player zero-sum game in 1×1 variables on the interval

$$x_i \in [-1, 1] \quad \forall i \in \{1, 2\}$$

with utility function given by

$$u_1(x_1, x_2) = -2x_{11}^2 - 4x_{11}x_{21} + x_{21}^2 - \frac{4}{3}x_{21}^3 + \frac{1}{4}x_{21}^4$$

This example was studied by Zheng et al. [98, Convex-Nonconcave Example] and also Chinchilla, Yang, and Hespanha [16] and Adolphs et al. [3].

Game A.3.15. A two-player zero-sum game in 1×1 variables on the interval

$$x_i \in [-1, 1] \quad \forall i \in \{1, 2\}$$

with utility function [98, KŁ-Nonconcave Example] given by

$$u_1(x_1, x_2) = -x_{11}^2 + 4x_{21}^2 + 10\sin^2(x_{21}) - 3\sin^2(x_{21})\sin^2(x_{11})$$

This game has the pure NE $x^\star = (0, 0)$.

Game A.3.16. A two-player zero-sum game in 1×1 variables on the interval

$$x_i \in [-4, 4] \quad \forall i \in \{1, 2\}$$

with utility function [98, Bilinearly-coupled Minimax] given by

$$u_1(x_1, x_2) = -f(x_{11}) - Ax_{11}x_{21} + f(x_{21})$$

where $f(z) = (z + 1)(z - 1)(z + 3)(z - 3)$. This example was also studied by Grimmer et al. [33].

Game A.3.17. A two-player zero-sum game in 1×1 variables on the interval

$$x_i \in [-1.5, 1.5] \quad \forall i \in \{1, 2\}$$

with utility function [98, "Forsaken" example] given by

$$u1(x, y) = -x_{11}(x_{21} - 0.45) - \phi(x_{11}) + \phi(x_{21})$$

where $\phi(z) = \frac{1}{4}z^2 - \frac{1}{2}z^4 + \frac{1}{6}z^6$. This problem was also studied by Hsieh, Mertikopoulos, and Cevher [39, Example 5.2]. The game has the mixed ϵ-NE:

$$\mu_1^\star = 0.5\delta_{-1.31} + 0.5\delta_{1.31},$$
$$\mu_2^\star = 0.328\delta_{-1.31} + 0.672\delta_{1.31}.$$

A.3.3. Experiments on Games by Parillo, Stein and Ozdaglar

The following examples are games where the strategy spaces for each player are intervals. The games were originally solved using Lasserre hierarchies, discretization techniques, or were used to demonstrate methods for obtaining bounds on the size of support of equilibria.

Game A.3.18. A two-player zero-sum game in 1×1 variables on the interval

$$x_i \in [-1, 1] \quad \forall i \in \{1, 2\}.$$

The utility function is [68, Example 2.1]

$$u_1(x_1, x_2) = (x_{11} - x_{21})^2$$

The Nash equilibrium is $\mu_1^\star = \mathcal{U}\{-1, 1\}$; $\mu_2^\star = \delta_0$.

Game A.3.19. A two-player zero-sum game in 1×1 variables on the interval

$$x_i \in [-1, 1] \quad \forall i \in \{1, 2\}.$$

The utility function is [68, Example 3.1]

$$u_1(x_1, x_2) = -x_{21} - x_{11}^2 + 2x_{21}^2 x_{11}$$

The game has the following pure NE and a value of $-\frac{3\sqrt[3]{2}}{8}$:

$$x_1^\star = 4^{-\frac{2}{4}},$$
$$x_2^\star = 4^{-\frac{1}{4}}.$$

Game A.3.20. A two-player zero-sum game in 1×1 variables on the interval

$$x_i \in [-1, 1] \quad \forall i \in \{1, 2\}.$$

The utility function is [68, Example 3.2]

$$u_1(x_1, x_2) = -x_{21} - 2x_{11}^2 + 5x_{11}x_{21} - 2x_{21}^2 x_{11}$$

The game has the following mixed NE with a value of 0.48:

$$\mu_1^\star \approx \delta_{0.2},$$
$$\mu_2^\star \approx 0.78\delta_1 + 0.22\delta_{-1}.$$

Game A.3.21. A general-sum 2-player game in 1×1 variables on the interval

$$x_i \in [-1, 1] \quad \forall i \in \{1, 2\}$$

with utility functions [86, Example 3.1] given by

$$u_1(x_1, x_2) = 0.554 + 1.36x_{11} - 1.2x_{21} + 0.596x_{11}^2 + 2.072x_{11}x_{21} - 0.394x_{21}^2$$
$$u_2(x_1, x_2) = 1.918x_{11}x_{21} - 1.886 - 1.232x_{11} + 0.842x_{21} - 0.108x_{11}^2 - 1.044x_{21}^2.$$

The game has an equilibrium $x^\star = (1, 1)$.

Game A.3.22. A general-sum 2-player game in 1×1 variables on the interval

$$x_i \in [-1, 1] \quad \forall i \in \{1, 2\}$$

with utility functions [85, Example 2.3] given by

$$u_1(x_1, x_2) = -x_{11} + 2x_{11}x_{21} - 2x_{11}^3 + 3x_{21}^3 - 3x_{21}^2 x_{11}^2$$
$$u_2(x_1, x_2) = 4x_{21} - x_{11}^2 + x_{11}^2 x_{21} - 4x_{21}^3 + 2x_{21}^2 x_{11}^2$$

The game has the mixed NE:

$$\mu_1^\star \approx 0.5532\delta_{-1} + 0.4468\delta_{0.1149},$$
$$\mu_2^\star \approx \delta_{0.7166}.$$

Game A.3.23. A general-sum 2-player game in 1×1 variables on the interval

$$x_i \in [-\pi, \pi] \quad \forall i \in \{1, 2\}$$

with utility functions [85, Example 2.4] given by

$$\begin{aligned} u_1(x_1, x_2) &= \cos(x_{11} - x_{21}) \\ u_2(x_1, x_2) &= \cos(-0.5 + x_{11} - x_{21}) \end{aligned}$$

The game has mixed equilibria where each player plays θ_i and $\theta_i + \pi$ with equal probability, where θ_i is arbitrary.

Game A.3.24. A 3-player game in $1 \times 1 \times 1$ variables on the interval

$$x_i \in [-1, 1] \quad \forall i \in \{1, 2, 3\}$$

with utility functions [85, Example 3.10] given by

$$\begin{aligned} u_1(x_1, x_2, x_3) &= 1 + 2x_{11} + 3x_{11}^2 + 2x_{21}x_{31} + 4x_{11}x_{21}x_{31} + 6x_{11}^2x_{21}x_{31} \\ &\quad + 3x_{31}^2x_{21}^2 + 6x_{31}^2x_{21}^2x_{11} + 9x_{31}^2x_{21}^2x_{11}^2 \\ u_2(x_1, x_2, x_3) &= 7 + 2x_{11} + 2x_{21} + 3x_{11}^2 + 4x_{11}x_{21} + 3x_{31}^2 + 6x_{11}^2x_{21} \\ &\quad + 6x_{31}^2x_{11} + 9x_{31}^2x_{11}^2 \\ u_3(x_1, x_2, x_3) &= -x_{31} - 2x_{11}x_{31} - 2x_{21}x_{31} - 3x_{11}^2x_{31} - 4x_{11}x_{21}x_{31} \\ &\quad - 3x_{31}^2x_{21} - 6x_{11}^2x_{21}x_{31} - 6x_{31}^2x_{11}x_{21} - 9x_{31}^2x_{11}^2x_{21} \end{aligned}$$

The game has the pure NE $x^\star = (1, 1, -0.5)$.

A.3.4. Experiments on Applied Games and Others

The following games are a mix of game theory applied to communication networks and security games, as well as others that did not fit elsewhere.

Game A.3.25. A general-sum 2-player game in 1×1 variables constrained to the torus

$$x_{i1} \in [-\pi, \pi] \quad \forall i \in \{1, 2\}$$

with utilities parameterized by $\phi = \left(0, \frac{\pi}{8}\right), \alpha = (1, 1.5)$:

$$\begin{aligned} u_1(x_1, x_2) &= \alpha_1 \cos(x_{11} - \phi_1) - \cos(x_{11} - x_{21}) \\ u_2(x_1, x_2) &= \alpha_2 \cos(x_{21} - \phi_2) - \cos(x_{21} - x_{11}) \end{aligned}$$

This game appeared in a paper by Chasnov et al. [15, p. 5.2], who reported pure equilibria at $(-1.063, 1.014)$ and $(1.408, -0.325)$.

Game A.3.26. A two-player zero-sum game in 1×1 variables constrained to the interval

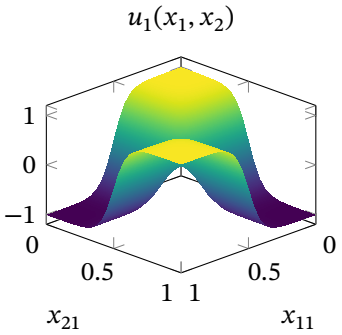
$$x_{i1} \in [0, 1] \quad \forall i \in \{1, 2\}.$$

The utility function is

$$u_1(x_1, x_2) = \sigma(x_2)^\mathsf{T} A \sigma(x_1)$$

where $\sigma(x)$ is the *softmax* vector

$$\sigma(x) = \left[\frac{e^{10x}}{e^{10x} + e^{10(1-x)}}, \frac{e^{10(1-x)}}{e^{10x} + e^{10(1-x)}} \right]$$



This continuous version of the classical matching-pennies game was studied by Chasnov et al. [15, B.1], who reported a pure equilibrium at $(x_{11}, x_{21}) = (0.5, 0.5)$.

Game A.3.27. A general-sum 2-player game in 1×1 variables on the interval

$$x_i \in [-\pi, \pi] \quad \forall i \in \{1, 2\}.$$

The utility functions are given by

$$\begin{aligned} u_1(x_1, x_2) &= \cos(x_{11}) - \alpha_1 \cos(x_{11} - x_{21}) \\ u_2(x_1, x_2) &= \cos(x_{21}) - \alpha_2 \cos(-x_{11} + x_{21}). \end{aligned}$$

Ratliff, Burden, and Sastry [73, Location Game] studied this game with $\alpha = [1, 1.05]$, in which case the game has two NE: at (x_1, x_2) equal to $(1, -1.1)$, or $(-1, 1.1)$.

Game A.3.28. A 3-player polynomial network game in $1 \times 1 \times 1$ variables on the interval

$$x_{i1} \in [0, 1] \quad \forall i \in \{1, 2, 3\}.$$

The utilities [44, Example 5] are

$$\begin{aligned} u_1(x_1, x_2, x_3) &= -x_{21} - 2x_{31} - 2x_{11}^2 + 5x_{11}x_{21} - 4x_{11}x_{31} - 2x_{21}^2 x_{11} \\ u_2(x_1, x_2, x_3) &= x_{21} - 2x_{11}^2 - 5x_{11}x_{21} - 2x_{21}^2 + 5x_{21}x_{31} + 2x_{21}^2 x_{11} - 2x_{31}^2 x_{21} \\ u_3(x_1, x_2, x_3) &= 2x_{31} + 4x_{11}^2 + 4x_{11}x_{31} + 2x_{21}^2 - 5x_{21}x_{31} + 2x_{31}^2 x_{21} \end{aligned}$$

The game has the following ϵ-NE:

$$\begin{aligned} \mu_1^\star &= 0.291\delta_{-0.056} + 0.191\delta_{-0.066} + 0.518\delta_{-0.06} \\ \mu_2^\star &= 0.232\delta_{0.347} + 0.147\delta_{0.359} + 0.622\delta_{0.352} \\ \mu_3^\star &= 0.281\delta_{-1.0} + 0.719\delta_{1.0}. \end{aligned}$$

Game A.3.29. A 4-player game in $4 \times 4 \times 2 \times 2$ variables constrained to the scaled simplexes:

$$x_i \in \left\{ z \in \mathbb{R}^4 \;\middle|\; z_i \geq 0,\ \sum_{i=1}^{m} z_i = a_i \right\} \quad \forall i \in \{1, 2\},$$
$$x_i \in \left\{ z \in \mathbb{R}^2 \;\middle|\; z_i \geq 0,\ \sum_{i=1}^{m} z_i = 1 \right\} \quad \forall i \in \{3, 4\}.$$

The utilities are defined by

$$\begin{aligned}
u_1(x_1, x_2, x_3, x_4) &= a_1 - f_1(x_{11}, x_{31}) - f_2(x_{12}, x_{32}) - f_3(x_{13}, x_{41}) - f_4(x_{14}, x_{42}) - o \\
u_2(x_1, x_2, x_3, x_4) &= a_2 - f_1(x_{21}, x_{31}) - f_2(x_{22}, x_{32}) - f_3(x_{23}, x_{41}) - f_4(x_{24}, x_{42}) - o \\
u_3(x_1, x_2, x_3, x_4) &= f_1(x_{11}, x_{31}) + f_2(x_{12}, x_{32}) + f_1(x_{21}, x_{31}) + f_2(x_{22}, x_{32}) - o \\
u_4(x_1, x_2, x_3, x_4) &= f_3(x_{13}, x_{41}) + f_4(x_{14}, x_{42}) + f_3(x_{23}, x_{41}) + f_4(x_{24}, x_{42}) - o
\end{aligned}$$

where $a = [0.5, 0.8]$, $o = 0.325$, and f are the *efficiency* functions:

$$\begin{aligned}
f_1(u, v) &= (1 - v^2)0.7u^2 \\
f_2(u, v) &= (1 - v)^2(1.4u - 0.7u^2) \\
f_3(u, v) &= (1 - v^2)(1.8u - 0.9u^2) \\
f_4(u, v) &= (1 - v)^2 0.9u^2
\end{aligned}$$

This polynomial security game was studied by Kroupa, Vannucci, and Votroubek [44, Example 6].

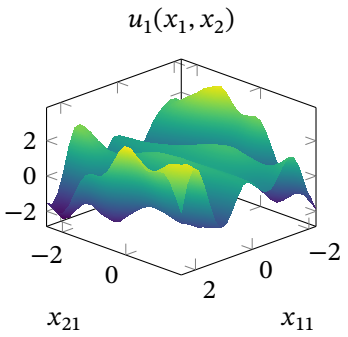
Game A.3.30. A two-player zero-sum game in 1×1 variables constrained to the intervals

$$x_{11} \in [-2.25, 2.5]$$
$$x_{21} \in [-2.5, 1.75]$$

with the utility given by the *Townsend* function

$$u_1(x_1, x_2) = x_{11} \sin(3x_{11} + x_{21}) + \cos^2((x_{11} - 0.1)\, x_{21})$$

This game was studied by Adam et al. [2, Townsend].



Game A.3.31. A general-sum 2-player game in $(m + 1) \times 1$ variables on the unit hypercubes

$$x_1 \in [0, 1]^{m+1}$$
$$x_2 \in [0, 1].$$

The utility functions are given by

$$u_1(x_1, x_2) = \sum_{\substack{i=0,\dots,m \\ j=i+1}} x_{1j}\binom{m}{i}x_{21}^i(1-x_{21})^{m-i} - \frac{\gamma}{2^m}\sum_{\substack{i=0,\dots,m \\ j=i+1}} x_{1j}\binom{m}{i} - 1$$

$$u_2(x_1, x_2) = -(x_{21} - 0.8)^2 + \sum_{\substack{i=0,\dots,m \\ j=i+1}} \binom{m}{i}(1-x_{1j})x_{21}^i(1-x_{21})^{m-i}.$$

This example is based on the *hypothesis testing game* by Yasodharan and Loiseau [97].

Game A.3.32. A 6-player game in $1\times1\times1\times1\times1\times1$ variables on the interval

$$x_{i1} \in [x_{\min}, x_{\max}] \quad \forall i \in \{1, \dots, 6\}.$$

The utility functions are defined by:

$$u_i(x) = -\eta_i x_{i1} + \beta_i\left(\ln\left(1 + a_i\frac{\gamma_i(x)}{1-\Phi_{ii}\gamma_i(x)}\right) - x_{i1}\right) \quad \forall i \in \{1, \dots, 6\},$$

where η_i, β_i, and a_i are channel-specific parameters. The parameter $\gamma_i(x)$ represents the OSNR [64], defined as

$$\gamma_i(x) = \frac{x_{i1}}{n_0 + \sum_j \Phi_{ij}x_{j1}},$$

for a given system matrix Φ and input noise n_0. The numerical setting is adopted from an article by Nguyen et al. [60]. Specifically, $n_0 = 0.43\times10^{-6}$, $\eta_1 = \dots = \eta_6 = 1$,

$$\beta = [0.5, 0.51, 0.52, 0.3, 0.31, 0.32],$$
$$a = [0.261, 0.494, 0.107, 0.366, 0.208, 0.305],$$

$x_{\min} = 0.2$, $x_{\max} = 2$, and the matrix

$$\Phi = \begin{bmatrix} 7.463 & 7.378 & 7.293 & 7.210 & 7.127 & 6.965 \\ 7.451 & 7.365 & 7.281 & 7.198 & 7.115 & 6.953 \\ 7.438 & 7.353 & 7.269 & 7.186 & 7.103 & 6.942 \\ 7.427 & 7.342 & 7.258 & 7.175 & 7.093 & 6.931 \\ 7.409 & 7.324 & 7.240 & 7.157 & 7.075 & 6.914 \\ 7.387 & 7.303 & 7.219 & 7.136 & 7.055 & 6.894 \end{bmatrix} \times 10^{-5}.$$

The game has a Nash equilibrium $x^\star \approx [0.3329, 0.3375, 0.3412, 0.2305, 0.2361, 0.2421]$.

Game A.3.33 (Tangent Ridge Game)**.** A two-player game in 1×1 variables on the interval

$$x_{i1} \in [0, 1] \quad \forall i \in \{1, 2\}.$$

The utility functions are defined by:

$$u_1(x_1, x_2) = -\sqrt{|x_1 - x_2|}$$
$$u_2(x_1, x_2) = -\sqrt{|x_1 - \tan(x_2)|}$$

The game has an equilibrium at $x^\star = [0, 0]$.

A.3.5. Rational Saddle Point Games

The following examples are the saddle point problems of rational polynomials studied by Zhou, Wang, and Zhao [99], which we interpret here as two-player zero-sum games where each player controls multiple variables.

Game A.3.34. A two-player zero-sum game in 2×2 variables on the unit hypercube

$$x_i \in [0,1]^2 \quad \forall i \in \{1,2\}.$$

The rational utility function [99, Example 5.3 (i)] is given by

$$u_1(x_1, x_2) = \frac{-x_{11}^2 - x_{12}^2 + x_{21}^2 + x_{22}^2}{x_{11} + x_{22} + 1}$$

This game has a pure Nash equilibrium at

$$x_1^\star = (0,0),\ x_2^\star = (0,0).$$

Game A.3.35. A two-player zero-sum game in 3×3 variables on the unit hypercube

$$x_i \in [0,1]^3 \quad \forall i \in \{1,2\}.$$

The rational utility function [99, Example 5.3 (ii)] is given by

$$u_1(x_1, x_2) = \frac{-\sum_{i=1}^{3}(x_{1i} + x_{2i}) - \sum_{\substack{i=1 \\ i<j}}^{3}\sum_{j=1}^{3}(x_{1i}^2 x_{2j}^2 - x_{2i}^2 x_{1j}^2)}{x_{11}^2 + x_{22}^2 + x_{13}x_{23} + 1}$$

This game has no pure Nash equilibria.

A.4. Optimization Test Functions

Following the idea of Adam et al. [2], who used a constrained-optimization example as the utility function of a two-player zero-sum game in Game A.3.30, we constructed games based on the dataset of optimization test functions by Surjanovic and Bingham [88]. Specifically, we selected one example from each shape category: many local minima, bowl-shaped, plate-shaped, valley-shaped, steep ridges, and other.

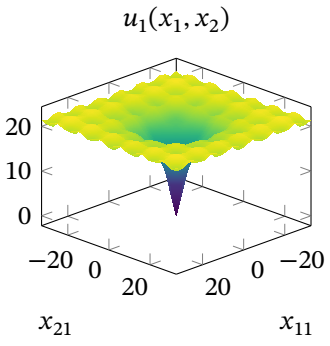
Game A.4.1. A two-player zero-sum game in 1×1 variables on the interval

$$x_i \in [-32.768, 32.768] \quad \forall i \in \{1,2\}$$

with the utility function [88, Ackley]

$$u_1(x_1, x_2) = -ae^{-b\sqrt{\frac{1}{2}\sum_{i=1}^{2} x_i^2}} - e^{\frac{1}{2}\sum_{i=1}^{2}\cos(cx_i)} + a + e$$

where $a = 20$, $b = 0.2$, and $c = 2\pi$.

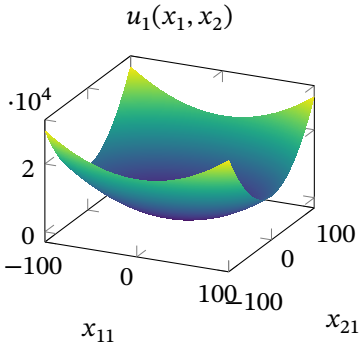


Game A.4.2. A two-player zero-sum game in 1×1 variables on the interval

$$x_i \in [-100, 100] \quad \forall i \in \{1, 2\}$$

with the utility function [88, Bohachevsky 1]

$$u_1(x_1, x_2) = -0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + x_1^2 + 2x_2^2 + 0.7$$



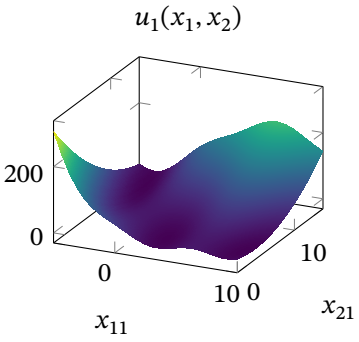
Game A.4.3. A two-player zero-sum game in 1×1 variables on the intervals

$$x_1 \in [-5, 10]$$
$$x_2 \in [0, 15]$$

with the utility function [88, Branin]

$$u_1(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1 - t)\cos(x_1) + s$$

where $a = 1$, $b = \frac{5.1}{4\pi^2}$, $c = \frac{5}{\pi}$, $r = 6$, $s = 10$, $t = \frac{1}{8\pi}$.

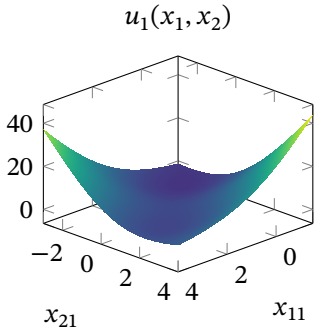


Game A.4.4. A two-player zero-sum game in 1×1 variables on the intervals

$$x_1 \in [-1.5, 4]$$
$$x_2 \in [-3, 4]$$

with the utility function [88, McCormick]

$$u_1(x_1, x_2) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - \frac{3x_1 + 5x_2 + 2}{2}$$



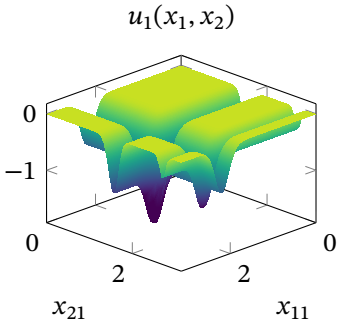
Game A.4.5. A two-player zero-sum game in 1×1 variables on the interval

$$x_i \in [0, \pi] \quad \forall i \in \{1, 2\}$$

with the utility function [88, Michalewicz]

$$u_1(x_1, x_2) = -\sum_{i=1}^{2} \sin(x_i) \sin^{2m}\left(\frac{i x_i^2}{\pi}\right)$$

where $m = 20$.

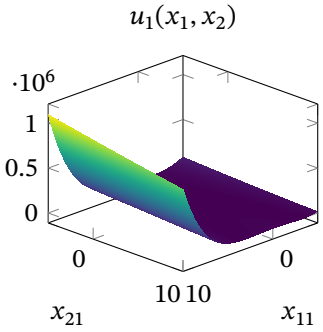


Game A.4.6. A two-player zero-sum game in 1×1 variables on the interval

$$x_i \in [-5, 10] \quad \forall i \in \{1, 2\}$$

with the utility function [88, Rosenbrock]

$$u_1(x_1, x_2) = 100(y - x^2)^2 + (x - 1)^2$$



A.5. Non-continuous Games with Known Equilibria

For completeness, we finish with a small number of games that either contain discontinuities or have non-compact strategy sets. Neither equilibria nor best responses necessarily exist in such games; for this reason, we include only examples with known solutions.

Game A.5.1. A two-player zero-sum game in $m \times m$ variables on the simplex

$$x_i \in \Delta^{m-1} = \left\{ z \in \mathbb{R}^m \;\middle|\; z_j \geq 0,\ \sum_{j=1}^{m} z_j = 1 \right\} \quad \forall i \in \{1, 2\}$$

The utility is defined by the discontinuous function

$$u_1(x_1, x_2) = \sum_{i=1}^{m} \operatorname{sign}\left(x_{1j} - x_{2j}\right)$$

This is an infinite version of the classical *Colonel Blotto* game [47, 45, 34, 32], in which all isolated fronts have equal value. Pure strategies exist only for $m \in \{1, 2\}$, in which case every strategy is a tie; or for $m = 3$, when players should allocate $\frac{1}{3}$ to every front. Otherwise, Roberson [76] proved that players must play a strategy with marginals $\mu_{ij}^{\star} = \mathcal{U}\left[0, \frac{2}{m}\right]$ for every front j.

A.5.1. Unbounded Polynomial Saddle Point Games

The following polynomial games are played over unbounded strategy sets.

Game A.5.2. A zero-sum 2-player game in 4×4 variables on the nonnegative orthant

$$x_i \in \mathbb{R}^4_+ \quad \forall i \in \{1, 2\}$$

with the utility function [62, Example 6.7]

$$\begin{aligned} u_1(x_1, x_2) = &- x_{21}(x_{12} + x_{13} + x_{14} - 1)^2 - x_{22}(x_{11} + x_{13} + x_{14} - 2)^2 \\ &- x_{23}(x_{11} + x_{12} + x_{14} - 3)^2 + x_{24}(x_{11} + x_{12} + x_{13} - 4)^2 \\ &+ x_{11}(x_{22} + x_{23} + x_{24} - 1)^2 + x_{12}(x_{21} + x_{23} + x_{24} - 2)^2 \\ &- x_{13}(x_{21} + x_{22} + x_{24} - 3)^2 + x_{14}(x_{21} + x_{22} + x_{23} - 4)^2 \end{aligned}$$

The game has a pure Nash equilibrium:

$$\begin{aligned} x_1^\star &\approx (1.5075, 0.5337, 0.0000, 0.5018), \\ x_2^\star &\approx (2.4143, 1.1463, 0.0000, 0.0000). \end{aligned}$$

Game A.5.3. A zero-sum 2-player game in 3×3 variables with no constraints

$$x_i \in \mathbb{R}^3 \quad \forall i \in \{1, 2\}.$$

The utility function [62, Example 6.8] is

$$u_1(x_1, x_2) = -\sum_{i=1}^{3}(x_{1i}^4 - x_{2i}^4 + x_{1i} + x_{2i}) - \sum_{i=1}^{3}\sum_{\substack{j=1 \\ i \neq j}}^{3} x_{1i}^3 x_{2j}^3$$

The game has a pure Nash equilibrium:

$$\begin{aligned} x_1^\star &\approx -(0.6981, 0.6981, 0.6981), \\ x_2^\star &\approx (0.4979, 0.4979, 0.4979). \end{aligned}$$

Game A.5.4. A zero-sum 2-player game in 3×3 variables subject to the constraints

$$x_i \in \left\{ z \in \mathbb{R}^3 \,\middle|\, z_1 \geq 0, z_1 z_2 \geq 1, z_2 z_3 \geq 1 \right\} \quad \forall i \in \{1, 2\}$$

which define a non-compact set. The utility function [62, Example 6.9] is

$$u_1(x_1, x_2) = -x_{11}^3 x_{21} - x_{12}^3 x_{22} - x_{13}^3 x_{23} + 3x_{11}x_{12}x_{13} + x_{21}^2 + 2x_{22}^2 + 3x_{23}^2$$

The game has a pure Nash equilibrium:

$$\begin{aligned} x_1^\star &\approx (1.2599, 1.2181, 1.3032), \\ x_2^\star &\approx (1.0000, 1.1067, 0.9036). \end{aligned}$$

A.5.2. Discontinuous Rational Saddle Point Games

The following games were created from the saddle point problems of rational polynomials studied by Zhou, Wang, and Zhao [99]. The utility functions are not continuous on the whole strategy set as the divisor often vanishes on the boundary.

Game A.5.5. A two-player zero-sum game in 3×3 variables on the simplex

$$x_i \in \Delta^2 \quad \forall i \in \{1, 2\}.$$

The rational utility function [99, Example 5.1] is given by

$$u_1(x_1, x_2) = \frac{-\sum_{i=1}^{3}\sum_{j=1}^{3} x_{1i}^2 x_{2j}^2 + \sum_{i=1}^{3}\sum_{\substack{j=1 \\ i\neq j}}^{3}(x_{1i}x_{1j} + x_{2i}x_{2j})}{x_{11} + x_{12} + x_{22} + x_{23}}$$

This game has two pure Nash equilibria:

$$\begin{aligned} \mathrm{NE}_1 &: x_1^\star \approx (0.3165, 0.3165, 0.3670),\ x_2^\star = (0, 1, 0), \\ \mathrm{NE}_2 &: x_1^\star \approx (0.3165, 0.3165, 0.3670),\ x_2^\star = (0, 0, 1). \end{aligned}$$

Game A.5.6. A two-player zero-sum game in 3×3 variables on the hypercube

$$x_i \in [-1, 1]^3 \quad \forall i \in \{1, 2\}.$$

The rational utility function [99, Example 5.2 (ii)] is given by

$$u_1(x_1, x_2) = \frac{-\sum_{i=1}^{3} x_{1i}^2 + \sum_{i=1}^{3} x_{2i}^2}{\sum_{i=1}^{3} x_{2i}^2 - \sum_{i=1}^{3} x_{1i}^2 + 3}$$

This game has a pure Nash equilibrium at

$$x_1^\star = (0, 0, 0),\ x_2^\star = (0, 0, 0).$$

Game A.5.7. A zero-sum 2-player game in 3×3 variables on the nonnegative orthant

$$x_i \in \mathbb{R}_+^3 \quad \forall i \in \{1, 2\}$$

with the utility function [99, Example 5.4 (i)]

$$u_1(x_1, x_2) = \frac{-x_{11}^4 - x_{12}^4 - x_{13}^4 + x_{21}^4 + x_{22}^4 + x_{23}^4}{x_{11} + x_{21} + 1}$$

The game has a pure Nash equilibrium:

$$\begin{aligned} x_1^\star &\approx (0.0015, 0.0015, 0.0015), \\ x_2^\star &\approx (0.0015, 0.0015, 0.0015). \end{aligned}$$

Game A.5.8. A two-player zero-sum game in 2×2 variables on the ball

$$x_i \in \left\{ z \in \mathbb{R}^2 \,\middle|\, \|z\| \leq 1 \right\} \quad \forall i \in \{1, 2\}$$

The rational utility function [99, Example 5.5 (i)] is

$$u_1(x_1, x_2) = \frac{-x_{11}^2 - x_{12}^2 + 3x_{11}x_{12} + x_{21}^2 + x_{22}^2 - x_{21}x_{22}}{x_{11}x_{21} + 1}$$

This game has no pure Nash equilibria.

Game A.5.9. A two-player zero-sum game in 3×3 variables on the ball

$$x_i \in \left\{ z \in \mathbb{R}^3 \,\middle|\, \|z\| \leq 1 \right\} \quad \forall i \in \{1, 2\}$$

The rational utility function [99, Example 5.5 (ii)] is

$$u_1(x_1, x_2) = \frac{-x_{11}^2 x_{21} - 2x_{12}^2 x_{22} - 3x_{13}^2 x_{23} + x_{11} + x_{12} + x_{13}}{x_{11}x_{21} + 1}$$

This game has a pure Nash equilibrium at

$$x_1^\star \approx (0.4730, 0.4476, 0.3416),\ x_2^\star \approx (0.6708, 0.5585, 0.4879).$$

Game A.5.10. A two-player zero-sum game in 3×3 variables on the sphere

$$x_i \in \left\{ z \in \mathbb{R}^3 \,\middle|\, \|z\| = 1 \right\} \quad \forall i \in \{1, 2\}$$

The rational utility function [99, Example 5.6 (i)] is

$$u_1(x_1, x_2) = \frac{-\sum_{i=1}^{3}(x_{1i}^3 + x_{2i}^3) - 2(x_{11}x_{12}x_{21}x_{22} + x_{11}x_{13}x_{21}x_{23} + x_{12}x_{13}x_{22}x_{23})}{x_{11} - x_{21} + 1}$$

The game has no pure NE.

Game A.5.11. A two-player zero-sum game in 3×3 variables on the sphere

$$x_i \in \left\{ z \in \mathbb{R}^3 \,\middle|\, \|z\| = 1 \right\} \quad \forall i \in \{1, 2\}$$

The rational utility function [99, Example 5.6 (ii)] is

$$u_1(x_1, x_2) = \frac{-\sum_{i=1}^{3}(x_{1i}^2 + x_{2i}^2 - x_{1i} - x_{2i})}{x_{13} + x_{22} + 2}$$

This game has a pure Nash equilibrium at

$$x_1^\star \approx (0.4082, 0.4082, 0.8165),\ x_2^\star \approx (-0.4082, -0.8165, -0.4082).$$

Game A.5.12. A two-player zero-sum game in 3×2 variables on the intersection of the nonnegative orthant and the n-sphere

$$x_1 \in \left\{z \in \mathbb{R}^3_+ \,|\, \|z\| = 1\right\}$$
$$x_2 \in \left\{z \in \mathbb{R}^2_+ \,|\, \|z\| = 1\right\}$$

The rational utility function is

$$u_1(x_1, x_2) = \frac{-\sum_{i=1}^{3}(x_{1i}^2 - x_{1i}) - \sum_{i=1}^{2}(x_{2i} - x_{2i}^2)}{x_{11}^2 - x_{21}^2 + 1}$$

The utility has a discontinuity at the boundary of x_1 when $x_{21} = 1$. Zhou, Wang, and Zhao [99, Example 5.7 (i)] report that this game has no saddle point, although we found the following pure ϵ-NE (with $\epsilon < 10^{-8}$):

$$x_1^\star \approx (0.4351314094, 0.6366555805, 0.6366555805),\ x_2^\star \approx (0.5255286571, 0.8507758992).$$

We also show this game reprojected in polar coordinates in Game A.5.13, where we recover the same solution.

Game A.5.13. A two-player zero-sum game in 3×2 variables on the intersection of the nonnegative orthant and the n-sphere in polar coordinates

$$x_1 \in \left[0, \frac{\pi}{2}\right]^2$$
$$x_2 \in \left[0, \frac{\pi}{2}\right]$$

The utility function is

$$u_1(x_1, x_2) = \upsilon\left(\left[\sin(x_{11})\cos(x_{12}), \sin(x_{11})\sin(x_{12}), \cos(x_{11})\right], \left[\cos(x_{21}), \sin(x_{21})\right]\right)$$

where

$$\upsilon(z_1, z_2) = \frac{-\sum_{i=1}^{3}(z_{1i}^2 - z_{1i}) - \sum_{i=1}^{2}(z_{2i} - z_{2i}^2)}{z_{11}^2 - z_{21}^2 + 1}$$

This example is the polar coordinate reformulation of the example by Zhou, Wang, and Zhao [99, Example 5.7 (i)]. Despite containing a discontinuity at the boundary of x_1 when $x_{21} = 0$ the game has the pure ϵ-NE shown in Fig. A.2:

$$x_1^\star \approx (0.8813, 0.9716),$$
$$x_2^\star \approx 1.0175.$$

Game A.5.14. A two-player zero-sum game in 3×3 variables on the intersection of the nonnegative orthant and the sphere

$$x_i \in \left\{z \in \mathbb{R}^3_+ \,|\, \|z\| = 1\right\} \quad \forall i \in \{1, 2\}$$

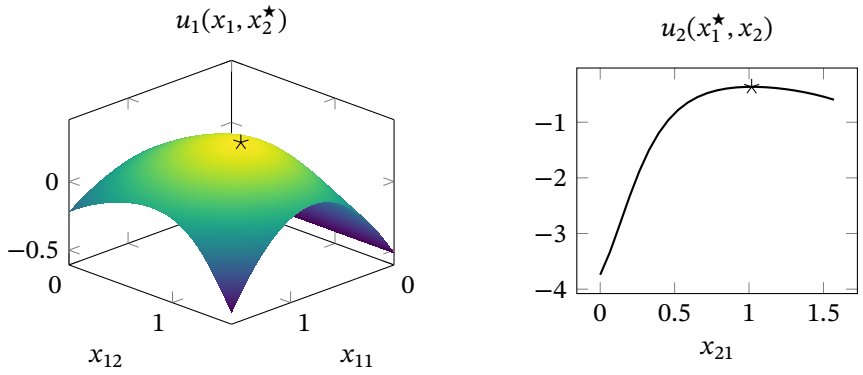


Figure A.2.: The best-response curves of both players at the ϵ-NE of Game A.5.13.

with the utility function [99, Example 5.7 (ii)]

$$u_1(x_1, x_2) = \frac{-\sum_{i=1}^{3}(x_{1i}-1)^2 - \sum_{i=1}^{3}(x_{2i}-1)^2}{x_{11}^2 - x_{21}^2 + 1}$$

The game has the pure NE

$$x_1^\star \approx (0.9519, 0.2167, 0.2167),\ x_2^\star \approx (1, 0, 0).$$

Game A.5.15. A general-sum 2-player game in 3×3 variables with no constraints

$$x_i \in \mathbb{R}^3 \quad \forall i \in \{1, 2\}.$$

with the utility function [99, Example 5.8 (i)]

$$u_1(x_1, x_2) = \frac{-x_{11}^2 - x_{12}^2 - x_{13}^2 - x_{21}^2 - x_{22}^2 + x_{11} + x_{12}}{x_{11}^2 + x_{23}^2 + 1}$$

The game has a pure Nash equilibrium:

$$x_1^\star \approx (0.3508, 0.5000, 0.0000),$$
$$x_2^\star \approx (0.0000, 0.0000, 0.0000).$$

Game A.5.16. A zero-sum 2-player game in 3×3 variables subject to the constraints

$$x_i \in \left\{z \in \mathbb{R}^3 \,\middle|\, z_1 \geq 0, z_1 z_2 \geq 1, z_2 z_3 \geq 1\right\} \quad \forall i \in \{1, 2\}$$

which define a non-compact set. The utility function [99, Example 5.9 (i)] is

$$u_1(x_1, x_2) = \frac{-\sum_{i=1}^{3}(x_{1i}^2 - x_{1i} + x_{2i} - x_{2i}^2)}{x_{11}^2 + x_{21}^2 + 1}$$

The game has a pure Nash equilibrium:

$$x_1^\star \approx (0.8914, 1.1219, 0.8914),$$
$$x_2^\star \approx (0.8914, 1.1219, 0.8914).$$

Game A.5.17. A general-sum 2-player game in 3×3 variables subject to the constraints

$$x_i \in \left\{ z \in \mathbb{R}^3 \,\middle|\, z_1 \geq 0, z_1 z_2 \geq 1, z_2 z_3 \geq 1 \right\} \quad \forall i \in \{1, 2\}$$

which define a non-compact set. The utility function [99, Example 5.9 (ii)] is

$$u_1(x_1, x_2) = \frac{-x_{11}^4 - x_{12}^4 + x_{21}^4 + x_{22}^4 - x_{11}x_{13} - x_{21}x_{23}}{x_{13}^2 + x_{23}^2 + 1}$$

The game has a pure Nash equilibrium:

$$x_1^\star \approx (0.9230, 1.0834, 2.8239),$$
$$x_2^\star \approx (1.0459, 0.9561, 1.3804).$$

B. Experimental Results of Multiple Oracle

In the main chapters, we focused on cases where the oracles and the master-problem would be tailored to each game. While the ability to leverage the problem structure by providing custom oracles based on closed-form best responses or reductions to classic combinatorial problems may be powerful, it naturally puts into question the applicability of our method.

It is certainly not possible to develop an oracle generic enough to handle all games, as this reduces to global optimization over arbitrary functions. Nevertheless, we will show that an off-the-shelf global optimization solver is sufficient to solve a surprisingly wide variety of continuous games automatically. To this end, we use the suite of continuous games compiled from the game-theoretic and optimisation literature presented in Appendix A. The collection spans multi-player polynomial, rational-polynomial, and more generic games, with action spaces that include n-spheres and simplexes.

A consistent implementation also enables us to report meaningful and comparable runtimes across games. Our experiments show that a generic algorithm can solve many continuous games efficiently. Notably, our implementation locates the pure-strategy Nash equilibrium of Game A.5.5 within a distance of less than 3×10^{-4} in 0.086 seconds compared to the 184.5 seconds achieved with a custom algorithm by Zhou, Wang, and Zhao [99].

MO may converge to mixed solutions with many atoms even in games where pure solutions exist. We found that tightening the ϵ is not a reliable method to produce simpler equilibria.

B.1. Experimental Evaluation

We show the performance of MO implemented in *Julia 1.11* using *Gurobi 12* as the oracle, and *gambit-logit* as the master-problem solver. Experiments were performed on a desktop PC with *AMD Ryzen 5 4600G* running *Linux*. The reported solve times include the complete run of the algorithm after a warm-start to force compilation. The stopping criterion was uniformly set to $\epsilon = 0.001$, i.e., no player should be able to deviate by more than ϵ utility units.

B.1.1. Experiments on General Blotto

The utility functions in *General Blotto* games are sums of non-linearly weighted margins of victory over sets of fronts. The family of weighing functions described by Golman and Page [32] is $f(z) = \operatorname{sgn}(z)\,|z|^p$, parametrized by p which is strictly positive for continuous games. The margins of victory are differences of products of the allocations to each front.

While products, exponents, sums and differences should be available through most modeling interfaces for non-linear problems; neither sgn, nor the absolute value is available via the

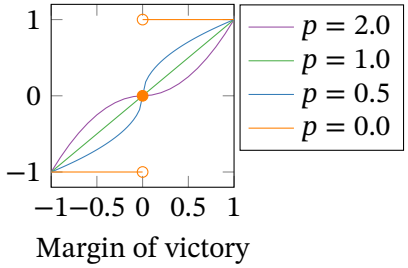


Figure B.1.: Behavior of weighing functions for *General Blotto* for $p \in \{0, 0.5, 1, 2\}$.

non-linear input for *Gurobi*. Instead, we modelled the absolute value as $\sqrt{z^2}$.

For each Game A.1.1-A.1.3, we computed equilibria in a game with 5 fronts for each of the values $p \in \{0.5, 1, 2\}$ as representative cases of each behavior as shown in Fig. B.1. The weighing functions for simplify to: $z\sqrt{z^2}$, for $p = 2$; and z for $p = 1$.

In the smooth case of $p = 2$, MO was able to recover a pure equilibrium for the game with isolated fronts, and mixed equilibria in the other two games where pure equilibria do not exist. The finding an initial feasible solution approximately 2 milliseconds for each example. Initializing with a best response solution did not consistently improve the runtimes.

ϵ-NE of A.1.1 ($p = 2$) In the case of 5 isolated fronts with $p = 2$, we computed the pure equilibrium $x_1^\star = (1.0, 0.0, 0.0, 0.0, 0.0)$, $x_2^\star = (1.0, 0.0, 0.0, 0.0, 0.0)$ in a single iteration, with a total runtime of 0.4 s.

ϵ-NE of A.1.2 ($p = 2$) For the case of all pairs of front we computed the mixed ϵ-NE:

$$\mu_1^\star \approx \frac{1}{3}\delta_{(1,0,0,0,0)} + \frac{1}{3}\delta_{(0,0,0,1,0)} + \frac{1}{3}\delta_{(0,1,0,0,0)},$$
$$\mu_2^\star \approx \frac{1}{3}\delta_{(1,0,0,0,0)} + \frac{1}{3}\delta_{(0,1,0,0,0)} + \frac{1}{3}\delta_{(0,0,0,1,0)}$$

in 5 iterations and a time of 0.91 s. Equilibrium:

ϵ-NE of A.1.3 ($p = 2$) With all sets equally valued, we got the following mixed ϵ-NE:

$$\mu_1^\star \approx \frac{1}{3}\delta_{(0,0,0,0,1)} + \frac{1}{3}\delta_{(1,0,0,0,0)} + \frac{1}{3}\delta_{(0,1,0,0,0)},$$
$$\mu_2^\star \approx \frac{1}{3}\delta_{(1,0,0,0,0)} + \frac{1}{3}\delta_{(0,0,0,0,1)} + \frac{1}{3}\delta_{(0,1,0,0,0)};$$

in 5 iterations and a time of 2.2 s.

MO was also able to successfully converge to pure equilibria in all the *General Blotto* games with $p = 1$:

ϵ-NE of A.1.1 ($p = 1$) For isolated fronts, MO computed ϵ-NE: $x_1^\star \approx (1, 0, 0, 0, 0)$, $x_2^\star \approx (1, 0, 0, 0, 0)$ in a single iteration and a time of 0.004 s.

ϵ-NE of A.1.2 ($p = 1$) For all pairs, MO computed the following ϵ-NE in two iterations and a time of 0.018 s: $x_1^\star \approx \left(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right)$, $x_2^\star \approx \left(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right)$

ϵ-NE of A.1.3 ($p = 1$) For all sets, MO found $x_1^\star \approx \left(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right)$, $x_2^\star \approx \left(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right)$ in two iterations and 1.5 s.

The utility functions are continuous but not differentiable for $0 < p < 1$; Since *Gurobi* does not accept neither the sign function nor absolute value through its non-linear interface. Attempting to model the utility as $z/\sqrt{z^2}$ introduces a discontinuity which *Gurobi* is able to avoid; however, the oracle did not find a best response within two minutes. The local solver *Ipopt* failed to compute a solution as well, since the utility is not differentiable. The global solver *Couenne* via *AMPL* depends on *Ipopt* and was similarly unsuccessful.

While our generic algorithm failed on all non-differentiable instances, specifying a custom oracle makes it possible to send more information about the utility to solvers. We investigated this approach in section 3.3.6.

To illustrate the complexity of the utility functions in Game A.1.3 where all sets of five fronts are equally important, we write it out partially below:

$$\begin{aligned} u_1(x_1, x_2) = {} & \operatorname{sgn}(x_{11}x_{13}x_{14} - x_{21}x_{23}x_{24})|x_{11}x_{13}x_{14} - x_{21}x_{23}x_{24}|^p \\ & + \operatorname{sgn}(x_{11}x_{12}x_{15} - x_{21}x_{22}x_{25})|x_{11}x_{12}x_{15} - x_{21}x_{22}x_{25}|^p \\ & + \operatorname{sgn}(x_{13}x_{14}x_{15} - x_{23}x_{24}x_{25})|x_{13}x_{14}x_{15} - x_{23}x_{24}x_{25}|^p \\ & + \operatorname{sgn}(x_{11}x_{12}x_{13}x_{15} - x_{21}x_{22}x_{23}x_{25})|x_{11}x_{12}x_{13}x_{15} - x_{21}x_{22}x_{23}x_{25}|^p \\ & + \operatorname{sgn}(x_{11}x_{13}x_{14}x_{15} - x_{21}x_{23}x_{24}x_{25})|x_{11}x_{13}x_{14}x_{15} - x_{21}x_{23}x_{24}x_{25}|^p \\ & + \ldots \text{ 26 more terms.} \end{aligned}$$

MO still successfully converges for a game with $p = 2$ and 40 isolated fronts in 11 seconds; with 20 fronts when all pairs are important in 12.7 seconds; and for a game with 7 fronts, of which all sets are important in 23.6 seconds, at which point the utility function contains the weighing function 127 times.

B.1.2. Experiments on Two-Player Zero-Sum Polynomial Games

The following examples are two-player zero-sum polynomial games, that originally served as experiments for the algorithm developed by Nie, Yang, and Zhou [62]. Their algorithm can prove the existence of (global) saddle points, i.e. pure Nash Equilibria.

MO usually converged to mixed strategies, even in the examples where pure equilibria exist. Tightening the ϵ did not result in simpler soultions.

ϵ-NE of A.2.1 found after 0.077 s and 7 iterations:

$$\begin{aligned} \mu_1^\star &\approx 0.001\delta_{(0,0,1)} + 0.999\delta_{(0,1,0)}, \\ \mu_2^\star &\approx 0.002\delta_{(0.54,0.46,0)} + 0.998\delta_{(0,0.49,0.51)}. \end{aligned}$$

ε-NE of A.2.2 found after 0.019 s and 2 iterations:

$$x_1^\star \approx (0,0,1),\ x_2^\star \approx (0,0,1).$$

ε-NE of A.2.3 found after 1.1 s and 22 iterations:

$$\mu_1^\star \approx 0.003\delta_{(0.23,0.24,0.27,0.26)} + 0.03\delta_{(0.24,0.24,0.25,0.27)} + 0.966\delta_{(0.25,0.25,0.25,0.25)},$$
$$\mu_2^\star \approx 0.236\delta_{(0,0,1,0)} + 0.236\delta_{(0,0,0,1)} + 0.242\delta_{(1,0,0,0)} + 0.286\delta_{(0,1,0,0)}.$$

ε-NE of A.2.4 found after 0.044 s and 7 iterations:

$$\mu_1^\star \approx 0.333\delta_{(1,0,0)} + 0.334\delta_{(0,1,0)} + 0.334\delta_{(0,0,1)},$$
$$\mu_2^\star \approx \delta_{(0.33,0.33,0.33)}.$$

ε-NE of A.2.5 found after 0.009 s and 2 iterations:

$$x_1^\star \approx (0,0),\ x_2^\star \approx (1,0).$$

ε-NE of A.2.6 found after 0.047 s and 8 iterations:

$$\mu_1^\star \approx 0.289\delta_{(0,0,0)} + 0.711\delta_{(0,0,1)},$$
$$\mu_2^\star \approx 0.376\delta_{(0.69,0.69,1)} + 0.624\delta_{(0.72,0.72,1)}.$$

ε-NE of A.2.7 found after 0.03 s and 3 iterations:

$$x_1^\star \approx (-1,1,-1),\ x_2^\star \approx (1,1,1).$$

ε-NE of A.2.8 found after 0.029 s and 5 iterations:

$$\mu_1^\star \approx 0.12\delta_{(1,1,-1)} + 0.16\delta_{(1,-1,-1)} + 0.16\delta_{(1,1,1)} + 0.28\delta_{(-1,-1,-1)} + 0.28\delta_{(-1,1,1)},$$
$$\mu_2^\star \approx 0.12\delta_{(1,1,-1)} + 0.16\delta_{(1,-1,-1)} + 0.16\delta_{(1,1,1)} + 0.28\delta_{(-1,-1,-1)} + 0.28\delta_{(-1,1,1)}.$$

ε-NE of A.2.9 found after 0.061 s and 2 iterations:

$$\mu_1^\star \approx \delta_{(-1,0,0)},$$
$$\mu_2^\star \approx 0.5\delta_{(0,0,1)} + 0.5\delta_{(1,0,0)}.$$

ε-NE of A.2.10 found after 1.5 s and 13 iterations:

$$\begin{aligned}\mu_1^\star \approx\ & 0.078\delta_{(0.71,-0.71,0.00)} + 0.098\delta_{(0.53,-0.8,0.27)} + 0.103\delta_{(-0.81,0.46,0.35)} \\ & + 0.104\delta_{(-0.77,0.13,0.63)} + 0.107\delta_{(-0.58,0.79,-0.21)} + 0.113\delta_{(-0.32,0.81,-0.49)} \\ & + 0.121\delta_{(-0.07,-0.67,0.74)} + 0.136\delta_{(0.69,0.03,-0.72)} + 0.14\delta_{(-0.45,-0.36,0.82)},\end{aligned}$$
$$\mu_2^\star \approx 0.008\delta_{(0.58,0.61,0.54)} + 0.209\delta_{(0.61,0.57,0.56)} + 0.783\delta_{(-0.57,-0.58,-0.58)}.$$

ϵ-NE of A.2.11 found after 0.094 s and 10 iterations:

$$\mu_1^\star \approx 0.204\delta_{(0.74,0.44,0.37)} + 0.796\delta_{(0.72,0.47,0.34)},$$
$$\mu_2^\star \approx 0.5\delta_{(0.7,0.57,0.44)} + 0.5\delta_{(0.69,0.53,0.49)}.$$

ϵ-NE of A.2.12 found after 0.13 s and 7 iterations:

$$\mu_1^\star \approx 0.333\delta_{(0,0,1)} + 0.333\delta_{(0,1,0)} + 0.333\delta_{(1,0,0)},$$
$$\mu_2^\star \approx \delta_{(0.58,0.58,0.58)}.$$

ϵ-NE of A.2.13 found after 0.017 s and 2 iterations:

$$\mu_1^\star \approx \delta_{(0,1,0,0,0)},$$
$$\mu_2^\star \approx 0.167\delta_{(0,1,0,0,0)} + 0.833\delta_{(1,0,0,0,0)}.$$

B.1.3. Games from Classical Literature

ϵ-NE of A.3.1 found after 0.018 s and 4 iterations:

$$\mu_1^\star \approx 0.5\delta_0 + 0.5\delta_1,$$
$$\mu_2^\star \approx \delta_{0.5}.$$

ϵ-NE of A.3.2 found after 0.14 s and 8 iterations:

$$\mu_1^\star \approx 0.523\delta_1 + 0.477\delta_0,$$
$$\mu_2^\star \approx 0.919\delta_{0.5} + 0.081\delta_{0.53}.$$

ϵ-NE of A.3.3 found after 0.042 s and 2 iterations:

$$x_1^\star \approx 0.5,\ x_2^\star \approx 0.5.$$

ϵ-NE of A.3.4 found after 0.074 s and 6 iterations:

$$\mu_1^\star \approx \delta_{0.5},$$
$$\mu_2^\star \approx 0.499\delta_1 + 0.501\delta_0.$$

ϵ-NE of A.3.5 found after 0.1 s and 4 iterations:

$$\mu_1^\star \approx 0.379\delta_0 + 0.621\delta_1,$$
$$\mu_2^\star \approx 0.313\delta_1 + 0.687\delta_{0.11}.$$

ϵ-NE of A.3.6 found after 0.043 s and 3 iterations:

$$\mu_1^\star \approx 0.25\delta_0 + 0.75\delta_1,$$
$$\mu_2^\star \approx 0.25\delta_0 + 0.75\delta_1.$$

ϵ-NE of A.3.7 found after 0.12 s and 6 iterations:

$$\mu_1^\star \approx 0.054\delta_{0.72} + 0.087\delta_0 + 0.087\delta_0 + 0.144\delta_1 + 0.628\delta_{0.41},$$
$$\mu_2^\star \approx 0.08\delta_0 + 0.231\delta_{0.41} + 0.333\delta_{0.19} + 0.356\delta_{0.8}.$$

B.1.4. Experiments on Games with Interesting Weaker Solution Concepts

ε-NE of A.3.8 found after 0.015 s and 1 iteration:

$$x_1^\star \approx 0.5,\ x_2^\star \approx 0.5.$$

ε-NE of A.3.9 found after 0.063 s and 4 iterations:

$$\mu_1^\star \approx 0.5\delta_1 + 0.5\delta_{-1},$$
$$\mu_2^\star \approx \delta_0.$$

ε-NE of A.3.10 found after 0.087 s and 6 iterations:

$$\mu_1^\star \approx 0.435\delta_{0.44} + 0.565\delta_{0.33},$$
$$\mu_2^\star \approx 0.192\delta_{2.5\times10^{-6}} + 0.238\delta_0 + 0.569\delta_1.$$

ε-NE of A.3.11 found after 0.027 s and 3 iterations:

$$x_1^\star \approx 0,\ x_2^\star \approx 0.$$

ε-NE of A.3.12 found after 0.016 s and 3 iterations:

$$\mu_1^\star \approx 0.5\delta_{-1} + 0.5\delta_1,$$
$$\mu_2^\star \approx 0.5\delta_{-6.28} + 0.5\delta_{6.28}.$$

ε-NE of A.3.13 found after 0.06 s and 5 iterations:

$$\mu_1^\star \approx 0.333\delta_{1.0} + 0.667\delta_{-0.5},$$
$$\mu_2^\star \approx 0.312\delta_{1.0} + 0.688\delta_{-1.0}.$$

ε-NE of A.3.14 found after 0.17 s and 9 iterations:

$$\mu_1^\star \approx 0.064\delta_{-0.04} + 0.936\delta_{-0.02},$$
$$\mu_2^\star \approx 0.03\delta_{-0.05} + 0.064\delta_1 + 0.302\delta_{-0.03804} + 0.302\delta_{-0.03802} + 0.302\delta_{-0.03801}.$$

ε-NE of A.3.15 found after 0.014 s and 2 iterations:

$$x_1^\star \approx 0,\ x_2^\star \approx 0.$$

ε-NE of A.3.16 found after 0.18 s and 8 iterations:

$$\mu_1^\star \approx 0.001\delta_{2.26} + 0.024\delta_{-2.25} + 0.476\delta_{-2.24} + 0.499\delta_{2.24},$$
$$\mu_2^\star \approx 0.25\delta_{2.24} + 0.25\delta_{2.24} + 0.5\delta_{-2.24}.$$

ε-NE of A.3.17 found after 0.13 s and 5 iterations:

$$\mu_1^\star \approx 0.498\delta_{1.32} + 0.502\delta_{-1.31},$$
$$\mu_2^\star \approx 0.33\delta_{-1.31} + 0.67\delta_{1.31}.$$

B.1.5. Experiments on Games by Parillo, Stein and Ozdaglar

We note that Game A.3.23 has much smaller known equilibria.

ϵ-NE of A.3.18 found after 0.016 s and 4 iterations:

$$\mu_1^\star \approx 0.5\delta_{-1.0} + 0.5\delta_{1.0},$$
$$\mu_2^\star \approx 1.0\delta_{0.0}.$$

ϵ-NE of A.3.19 found after 0.036 s and 6 iterations:

$$x_1^\star \approx 0.39,\ x_2^\star \approx 0.63.$$

ϵ-NE of A.3.20 found after 0.073 s and 9 iterations:

$$\mu_1^\star \approx 0.118\delta_{0.18} + 0.882\delta_{0.2},$$
$$\mu_2^\star \approx 0.224\delta_{-1.0} + 0.776\delta_{1.0}.$$

ϵ-NE of A.3.21 found after 0.018 s and 4 iterations:

$$x_1^\star \approx 1,\ x_2^\star \approx 1.$$

ϵ-NE of A.3.22 found after 0.13 s and 8 iterations:

$$\mu_1^\star \approx 0.056\delta_{0.111} + 0.379\delta_{0.115} + 0.565\delta_{-1.0},$$
$$\mu_2^\star \approx 0.309\delta_{0.73} + 0.691\delta_{0.71}.$$

ϵ-NE of A.3.23 found after 0.48 s and 22 iterations:

$$\begin{aligned}\mu_1^\star \approx\ & 0.066\delta_{-2.5} + 0.069\delta_{-3.0} + 0.069\delta_{-2.0} + 0.078\delta_{2.78} + 0.078\delta_{-1.5} \\ & + 0.091\delta_{2.28} + 0.091\delta_{-1.0} + 0.107\delta_{1.78} + 0.107\delta_{-0.5} + 0.122\delta_{1.28} + 0.122\delta_{0.0},\end{aligned}$$
$$\begin{aligned}\mu_2^\star \approx\ & 0.076\delta_{-3.0} + 0.076\delta_{-2.5} + 0.078\delta_{2.78} + 0.078\delta_{-2.0} + 0.081\delta_{2.28} + 0.081\delta_{-1.5} \\ & + 0.085\delta_{1.78} + 0.085\delta_{-1.0} + 0.089\delta_{1.28} + 0.089\delta_{-0.5} + 0.092\delta_{0.78} + 0.092\delta_{0.0}.\end{aligned}$$

ϵ-NE of A.3.24 found after 0.017 s and 3 iterations:

$$x_1^\star \approx 1.0,\ x_2^\star \approx 1.0,\ x_3^\star \approx -0.5.$$

B.1.6. Experiments on Applied Games and Others

ϵ-NE of A.3.25 found after 0.037 s and 3 iterations:

$$\mu_1^\star \approx \delta_{-1.08},$$
$$\mu_2^\star \approx 0.5\delta_{0.98} + 0.5\delta_{0.98}.$$

ε-NE of A.3.26 found after 0.034 s and 3 iterations:

$$\mu_1^\star \approx 0.5\delta_{0.0} + 0.5\delta_{1.0},$$
$$\mu_2^\star \approx 0.5\delta_{0.0} + 0.5\delta_{1.0}.$$

ε-NE of A.3.27 found after 0.02 s and 2 iterations:

$$x_1^\star \approx 0,\ x_2^\star \approx -3.14.$$

ε-NE of A.3.28 found after 0.32 s and 9 iterations:

$$\mu_1^\star \approx 0.218\delta_{-0.08} + 0.782\delta_{-0.06},$$
$$\mu_2^\star \approx 0.161\delta_{0.37} + 0.839\delta_{0.35},$$
$$\mu_3^\star \approx 0.278\delta_{-1} + 0.722\delta_{1}.$$

ε-NE of A.3.29 found after 1.6 s and 11 iterations:

$$\mu_1^\star \approx 0.001\delta_{(0.3,0,0,0.2)} + 0.218\delta_{(0.29,0,0,0.21)} + 0.245\delta_{(0.29,0,0,0.21)} + 0.268\delta_{(0.29,0,0,0.21)} + 0.268\delta_{(0.29,0,0,0.21)},$$
$$\mu_2^\star \approx 0.278\delta_{(0.34,0,0.22,0.24)} + 0.279\delta_{(0.32,0.25,0,0.23)} + 0.444\delta_{(0.4,0.12,0,0.28)},$$
$$\mu_3^\star \approx 0.157\delta_{(1,0)} + 0.158\delta_{(0.97,0.03)} + 0.171\delta_{(0.06,0.94)} + 0.171\delta_{(0.05,0.95)} + 0.171\delta_{(0.02,0.98)} + 0.171\delta_{(0,1)},$$
$$\mu_4^\star \approx 0.239\delta_{(0,1)} + 0.761\delta_{(1,0)}.$$

ε-NE of A.3.30 found after 0.62 s and 11 iterations:

$$\mu_1^\star \approx 0.008\delta_{-2.09} + 0.013\delta_{1.33} + 0.979\delta_{0.03},$$
$$\mu_2^\star \approx 0.064\delta_{-2.17} + 0.137\delta_{-0.9} + 0.253\delta_{0.93} + 0.546\delta_{-2.11}.$$

ε-NE of A.3.31 found after 0.015 s and 3 iterations:

$$x_1^\star \approx (1,1),\ x_2^\star \approx 0.8.$$

ε-NE of A.3.32 found after 0.37 s and 2 iterations:

$$\mathbf{x}^\star \approx (0.33, 0.34, 0.34, 0.23, 0.23, 0.24).$$

B.1.7. Experiments on Two-Player Zero-Sum Rational Games

ε-NE of A.3.34 found after 0.006 s and 1 iterations:

$$x_1^\star \approx (0,0),\ x_2^\star \approx (0,0).$$

ε-NE of A.3.35 found after 0.16 s and 6 iterations:

$$\mu_1^\star \approx 0.149\delta_{(0,0,0)} + 0.425\delta_{(0,0,1)} + 0.425\delta_{(0,0,1)},$$
$$\mu_2^\star \approx 0.46\delta_{(0.7,0.24,1)} + 0.54\delta_{(0.58,0.24,0)}.$$

B.1.8. Experiments on Optimization Test Functions

As we searched for an ϵ-NE with $\epsilon = 0.001$ consistently across all games, two of the following examples had to be rescaled. Game A.4.2 was scaled by 10^{-2}, and Game A.4.6 by 10^{-3} as its utility reaches over 10^6 making the stopping criterion impractically strict.

ϵ-NE of A.4.1 found after 0.023 s and 2 iterations:

$$x_{1\star} \approx -32.5,\ x_{2\star} \approx 0.$$

ϵ-NE of A.4.2 found after 0.012 s and 2 iterations:

$$x_1^\star \approx -100.0,\ x_2^\star \approx 0.$$

ϵ-NE of A.4.3 found after 0.18 s and 11 iterations:

$$\mu_1^\star \approx 0.151\delta_{6.23} + 0.362\delta_{6.25} + 0.487\delta_{-5.0},$$
$$\mu_2^\star \approx 0.023\delta_{9.01} + 0.464\delta_{8.91} + 0.514\delta_{8.95}.$$

ϵ-NE of A.4.4 found after 0.15 s and 10 iterations:

$$\mu_1^\star \approx 0.403\delta_{-1.5} + 0.597\delta_4,$$
$$\mu_2^\star \approx 0.128\delta_{0.51} + 0.872\delta_{0.49}.$$

ϵ-NE of A.4.5 found after 0.015 s and 2 iterations:

$$x_1^\star \approx 0,\ x_2^\star \approx 1.57.$$

ϵ-NE of A.4.6 found after 0.013 s and 2 iterations:

$$x_1^\star \approx 10,\ x_2^\star \approx 10.$$

B.1.9. Results on Discontinuous Games

MO is not guaranteed to converge on discontinuous games; However, as long as the discontinuities can be avoided, the algorithm often recovers a solution. The following games by Zhou, Wang, and Zhao [99] contain discontinuities, but converge without any reformulation or modification to the algorithm. Time required for warm-starting is indicated in parentheses.

ϵ-NE of A.5.5 found after 0.1 (+0.002) s and 4 iterations:

$$x_1^\star \approx (0.32, 0.32, 0.37),\ x_2^\star \approx (0, 0, 1).$$

ϵ-NE of A.5.8 found after 0.22 (+0.002) s and 6 iterations:

$$\mu_1^\star \approx 0.424\delta_{(0.71,0.71)} + 0.307\delta_{(-0.71,-0.71)} + 0.269\delta_{(-0.71,-0.71)},$$
$$\mu_2^\star \approx 1.0\delta_{(0,0)}.$$

ε-NE of A.5.9 found after 0.38 (+0.002) s and 10 iterations:

$$\mu_1^\star \approx 0.548\delta_{(0.46,0.47,0.34)} + 0.452\delta_{(0.49,0.43,0.35)},$$
$$\mu_2^\star \approx 0.641\delta_{(0.67,0.54,0.5)} + 0.359\delta_{(0.64,0.59,0.49)}.$$

ε-NE of A.5.11 found after 0.062 (+0.006) s and 3 iterations:

$$x_1^\star \approx (0.41, 0.41, 0.81),\ x_2^\star \approx (-0.39, -0.84, -0.39).$$

ε-NE of A.5.12 found after 0.035 (+0.004) s and 2 iterations:

$$x_1^\star \approx (0.42, 0.64, 0.64),\ x_2^\star \approx (0.54, 0.84).$$

Our implementation of MO initially failed to solve Examples A.5.6 and A.5.13 as it attempted to initialize at the discontinuity. Both examples were solved by initializing in the interior of the sets.

ε-NE of A.5.6 found after 0.0086 (+0.002) s and 1 iterations:

$$x_1^\star \approx (0, 0, 0),\ x_2^\star \approx (0, 0, 0).$$

ε-NE of A.5.13 found after 0.056 (+0.002) s and 2 iterations:

$$x_1^\star \approx (0.87, 1),\ x_2^\star \approx 1.$$

MO cannot directly be applied to search for solutions in Games with unbounded strategy spaces. We know of no global solver powerful enough to automatically infer bounds on the location of best responses in general games.

Bibliography

[1] Lukáš Adam et al. “Double Oracle Algorithm For Computing Equilibria In Continuous Games”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021, pp. 5070–5077.

[2] Lukáš Adam et al. “Double Oracle Algorithm for Computing Equilibria in Continuous Games”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.6 (May 2021), pp. 5070–5077. DOI: 10.1609/aaai.v35i6.16641. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16641.

[3] Leonard Adolphs et al. “Local Saddle Point Optimization: A Curvature Exploitation Approach”. In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, Apr. 2019, pp. 486–495. URL: https://proceedings.mlr.press/v89/adolphs19a.html.

[4] J.-P. Bailey and G. Piliouras. “Multi-Agent Learning in Network Zero-Sum Games is a Hamiltonian System”. In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. Montreal, QC, Canada: International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 233–241.

[5] T. Başar and G. Olsder. *Dynamic Noncooperative Game Theory, 2nd Edition*. Society for Industrial and Applied Mathematics, 1999.

[6] Dimitri Bertsekas. *Nonlinear Programming*. Athena Scientific, 2016.

[7] Ksenia Bestuzheva et al. *The SCIP Optimization Suite 8.0*. Technical Report. Optimization Online, Dec. 2021. URL: http://www.optimization-online.org/DB_HTML/2021/12/8728.html.

[8] Branislav Bošanský et al. “An Exact Double-Oracle Algorithm For Zero-Sum Extensive-Form Games With Imperfect Information”. In: *Journal of Artificial Intelligence Research* 51 (2014), pp. 829–866.

[9] L M Bregman and I N Fokin. “On separable non-cooperative zero-sum games”. In: *Optimization* 44 (1998).

[10] Benjamin Brooks and Philip J. Reny. “A canonical game—75 years in the making—showing the equivalence of matrix games and linear programming”. In: *Economic Theory Bulletin* 11.2 (Oct. 2023), pp. 171–180. ISSN: 2196-1093. DOI: 10.1007/s40505-023-00252-8. URL: https://doi.org/10.1007/s40505-023-00252-8.

[11] Samuel Buss. “Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods”. In: *IEEE Transactions in Robotics and Automation* 17 (May 2004).

[12] Y. Cai and C. Daskalakis. “On Minmax Theorems for Multiplayer Games”. In: *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*. San Francisco, California: Society for Industrial and Applied Mathematics, 2011, pp. 217–234. DOI: 10.1137/1.9781611973082.20.

[13] Yang Cai et al. “Zero-Sum Polymatrix Games: A Generalization Of Minmax”. In: *Mathematics of Operations Research* 41.2 (2016), pp. 648–655.

[14] Benjamin Chasnov et al. “Convergence Analysis Of Gradient-Based Learning In Continuous Games”. In: *Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 935–944.

[15] Benjamin Chasnov et al. “Convergence analysis of gradient-based learning in continuous games”. In: *Uncertainty in artificial intelligence*. PMLR. 2020, pp. 935–944.

[16] Raphael Chinchilla, Guosong Yang, and João P. Hespanha. “Newton and interior-point methods for (constrained) nonconvex–nonconcave minmax optimization with stability and instability guarantees”. In: *Mathematics of Control, Signals, and Systems* 36.2 (June 2024), pp. 381–421. ISSN: 1435-568X. DOI: 10.1007/s00498-023-00371-4. URL: https://doi.org/10.1007/s00498-023-00371-4.

[17] Hongkai Dai, Gregory Izatt, and Russ Tedrake. “Global inverse kinematics via mixed-integer convex optimization”. In: *Int. J. Rob. Res.* 38.12-13 (Oct. 2019), pp. 1420–1441.

[18] Constantinos Daskalakis. “Non-concave games: A challenge for game theory’s next 100 years”. In: *Nobel symposium” One Hundred Years of Game Theory: Future Applications and Challenges*. 2021.

[19] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. “The Complexity Of Computing A Nash Equilibrium”. In: *SIAM Journal on Computing* 39.1 (2009), pp. 195–259.

[20] Constantinos Daskalakis et al. *STay-ON-the-Ridge: Guaranteed Convergence to Local Minimax Equilibrium in Nonconvex-Nonconcave Games*. 2022. arXiv: 2210.09769 [cs.LG]. URL: https://arxiv.org/abs/2210.09769.

[21] Constantinos Daskalakis et al. “Stay-on-the-ridge: Guaranteed convergence to local minimax equilibrium in nonconvex-nonconcave games”. In: *The Thirty Sixth Annual Conference on Learning Theory*. PMLR. 2023, pp. 5146–5198.

[22] Santanu S Dey, Yatharth Dubey, and Marco Molinaro. “Branch-and-bound solves random binary IPs in polytime”. In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2021, pp. 579–591.

[23] Melvin Dresher. *Games of strategy*. Santa Monica, CA: RAND, Sept. 2007.

[24] I Dunning, J Huchette, and M Lubin. “JuMP: a modeling language for mathematical optimization”. In: *SIAM Rev.* 59 (2017).

[25] Adam Dziwoki and Rostislav Horcik. *Perturbing Best Responses in Zero-Sum Games*. 2025. arXiv: 2511.12523 [cs.GT]. URL: https://arxiv.org/abs/2511.12523.

[26] C Ewerhart and K Valkanova. “Fictitious play in networks”. In: *Games Econom. Behav.* 123 (2020).

[27] Miriam Fischer and Akshay Gupte. “Multilinear Formulations for Computing a Nash Equilibrium of Multi-Player Games”. en. In: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. DOI: 10.4230/LIPICS.SEA.2023.12. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.SEA.2023.12.

[28] David Gale and Oliver Gross. “A note on polynomial and separable games.” In: (1958).

[29] Sam Ganzfried. “Algorithm for Computing Approximate Nash Equilibrium in Continuous Games with Application to Continuous Blotto”. In: *Games* 12.2 (2021), p. 47.

[30] I. L. Glicksberg. “A Further Generalization of the Kakutani Fixed Point Theorem, With Application to Nash Equilibrium Points”. In: *Proceedings of the American Mathematical Society* 3 (1952), pp. 170–174.

[31] Russell Golman and Scott E Page. “General Blotto: Games Of Allocative Strategic Mismatch”. In: *Public Choice* 138.3-4 (2009), pp. 279–299.

[32] Russell Golman and Scott E. Page. “General Blotto: games of allocative strategic mismatch”. In: *Public Choice* 138.3 (Mar. 2009), pp. 279–299. ISSN: 1573-7101. DOI: 10.1007/s11127-008-9359-x. URL: https://doi.org/10.1007/s11127-008-9359-x.

[33] Benjamin Grimmer et al. “The landscape of the proximal point method for nonconvex–nonconcave minimax optimization”. In: *Mathematical Programming* 201.1 (Sept. 2023), pp. 373–407. ISSN: 1436-4646. DOI: 10.1007/s10107-022-01910-8. URL: https://doi.org/10.1007/s10107-022-01910-8.

[34] Oliver Alfred Gross and R. A. Wagner. *A Continuous Colonel Blotto Game*. Santa Monica, CA: RAND Corporation, 1950.

[35] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2022. URL: https://www.gurobi.com.

[36] Didier Henrion and Jean-Bernard Lasserre. “Detecting Global Optimality and Extracting Solutions in GloptiPoly”. In: *Positive Polynomials in Control*. Ed. by Didier Henrion and Andrea Garulli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 293–310. ISBN: 978-3-540-31594-0. DOI: 10.1007/10997703_15. URL: https://doi.org/10.1007/10997703_15.

[37] Michael Hinze et al. *Optimization with PDE constraints*. Vol. 23. Springer Science & Business Media, 2008.

[38] Josef Hofbauer and Sylvain Sorin. “Best Response Dynamics for Continuous Zero-sum Games”. In: *Discrete and Continuous Dynamical Systems–Series B* 6.1 (2006), p. 215.

[39] Ya-Ping Hsieh, Panayotis Mertikopoulos, and Volkan Cevher. “The Limits of Min-Max Optimization Algorithms: Convergence to Spurious Non-Critical Sets”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 4337–4348. URL: https://proceedings.mlr.press/v139/hsieh21a.html.

[40] Nitin Kamra et al. “Policy Learning for Continuous Space Security Games Using Neural Networks”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018, pp. 1103–1112.

[41] Samuel Karlin. “CHAPTER 7 - MISCELLANEOUS GAMES**This entire chapter is of more special nature and may be omitted at first reading without loss of continuity.” In: *The Theory of Infinite Games*. Ed. by Samuel Karlin. Pergamon, 1959, pp. 175–205. ISBN: 978-1-4831-9897-2. DOI: https://doi.org/10.1016/B978-1-4831-9897-2.50012-3. URL: https://www.sciencedirect.com/science/article/pii/B9781483198972500123.

[42] Samuel Karlin. *Mathematical Methods and Theory in Games, Programming and Economics. Vol. 2: The Theory of Infinite Games*. Addison-Wesley Publishing Company, 1959.

[43] Igor Klep, Janez Povh, and Jurij Volcic. “Minimizer extraction in polynomial optimization is robust”. In: *SIAM Journal on Optimization* 28.4 (2018), pp. 3177–3207.

[44] Tomáš Kroupa, Sara Vannucci, and Tomáš Votroubek. “Separable Network Games with Compact Strategy Sets”. In: *Decision and Game Theory for Security*. Ed. by Branislav Bošanský et al. Cham: Springer International Publishing, 2021, pp. 37–56. ISBN: 978-3-030-90370-1.

[45] Dmitriy Kvasov. “Contests with limited resources”. In: *Journal of Economic Theory* 136.1 (2007), pp. 738–748. ISSN: 0022-0531. DOI: https://doi.org/10.1016/j.jet.2006.06.007. URL: https://www.sciencedirect.com/science/article/pii/S0022053106001244.

[46] Rida Laraki and Jean B Lasserre. “Semidefinite Programming for Min–max Problems and Games”. In: *Mathematical programming* 131.1-2 (2012), pp. 305–332.

[47] Jean-François Laslier and Nathalie Picard. “Distributive Politics and Electoral Competition”. In: *Journal of Economic Theory* 103.1 (2002), pp. 106–130. ISSN: 0022-0531. DOI: https://doi.org/10.1006/jeth.2000.2775. URL: https://www.sciencedirect.com/science/article/pii/S0022053100927753.

[48] Jean Bernard Lasserre. *An Introduction To Polynomial And Semi-Algebraic Optimization*. Vol. 52. Cambridge University Press, 2015.

[49] Jean Bernard Lasserre. *An Introduction to Polynomial and Semi-Algebraic Optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2015.

[50] Jean Bernard Lasserre, Edouard Pauwels, and Mihai Putinar. *The Christoffel–Darboux Kernel for Data Analysis*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2022.

[51] Monique Laurent. “Sums of Squares, Moment Matrices and Optimization Over Polynomials”. In: *Emerging Applications of Algebraic Geometry*. Ed. by Mihai Putinar and Seth Sullivant. New York, NY: Springer New York, 2009, pp. 157–270. ISBN: 978-0-387-09686-5. DOI: 10.1007/978-0-387-09686-5_7. URL: https://doi.org/10.1007/978-0-387-09686-5_7.

[52] Zun Li and Michael P Wellman. "Evolution Strategies for Approximate Solution of Bayesian Games". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 6. 2021, pp. 5531–5540.

[53] R. Lougee-Heimer. "The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community". In: *IBM Journal of Research and Development* 47.1 (2003), pp. 57–66. DOI: 10.1147/rd.471.0057.

[54] D. Manocha and J.F. Canny. "Efficient inverse kinematics for general 6R manipulators". In: *IEEE Transactions on Robotics and Automation* 10.5 (1994), pp. 648–657. DOI: 10.1109/70.326569.

[55] H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. "Planning in the Presence of Cost Functions Controlled by an Adversary". In: *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 2003, pp. 536–543.

[56] Panayotis Mertikopoulos and Zhengyuan Zhou. "Learning In Games With Continuous Action Sets And Unknown Payoff Functions". In: *Mathematical Programming* 173.1 (2019), pp. 465–507.

[57] Panayotis Mertikopoulos et al. "Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile". In: (July 2018). arXiv: 1807.02629 [cs.LG].

[58] Panayotis Mertikopoulos et al. *Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile*. 2018. arXiv: 1807.02629 [cs.LG]. URL: https://arxiv.org/abs/1807.02629.

[59] John Nash. "Non-Cooperative Games". In: *The Annals of Mathematics* 54.2 (Sept. 1951), p. 286. ISSN: 0003-486X. DOI: 10.2307/1969529. URL: http://dx.doi.org/10.2307/1969529.

[60] Duong Thuy Anh Nguyen et al. "Nash Equilibrium Seeking Over Digraphs With Row-Stochastic Matrices and Network-Independent Step-Sizes". In: *IEEE Control Systems Letters* 7 (2023), pp. 3543–3548. DOI: 10.1109/LCSYS.2023.3337206.

[61] Jiawang Nie. "Optimality conditions and finite convergence of Lasserre's hierarchy". In: *Mathematical Programming* 146.1 (Aug. 2014), pp. 97–121. ISSN: 1436-4646. DOI: 10.1007/s10107-013-0680-x. URL: https://doi.org/10.1007/s10107-013-0680-x.

[62] Jiawang Nie, Zi Yang, and Guangming Zhou. "The Saddle Point Problem of Polynomials". In: *Foundations of Computational Mathematics* 22.4 (Aug. 2022), pp. 1133–1169. ISSN: 1615-3383. DOI: 10.1007/s10208-021-09526-8. URL: https://doi.org/10.1007/s10208-021-09526-8.

[63] Luyao Niu et al. "A Game-Theoretic Framework for Controlled Islanding in the Presence of Adversaries". In: *International Conference on Decision and Game Theory for Security*. Springer. 2021, pp. 231–250.

[64] Y. Pan and L. Pavel. "Global Convergence of An Iterative Gradient Algorithm for The Nash Equilibrium in An Extended OSNR Game". In: *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*. 2007, pp. 206–212. DOI: 10.1109/INFCOM.2007.32.

[65] Victor M Panaretos and Yoav Zemel. *An Invitation To Statistics In Wasserstein Space*. Springer Nature, 2020.

[66] Alberto Parmiggiani et al. "The Design of the iCub Humanoid Robot". In: *International Journal of Humanoid Robotics* 9 (Dec. 2012). DOI: 10.1142/S0219843612500272.

[67] P.A. Parrilo. "Polynomial Games and Sum of Squares Optimization". In: *Decision and Control, 2006 45th IEEE Conference on*. 2006, pp. 2855–2860.

[68] Pablo A. Parrilo. "Polynomial games and sum of squares optimization". In: *Proceedings of the 45th IEEE Conference on Decision and Control*. 2006, pp. 2855–2860. DOI: 10.1109/CDC.2006.377261.

[69] Ugo Pattacini et al. "An experimental evaluation of a novel minimum-jerk Cartesian controller for humanoid robots". In: Oct. 2010, pp. 1668–1674. DOI: 10.1109/IROS.2010.5650851.

[70] Gabriel Peyré and Marco Cuturi. "Computational Optimal Transport: With Applications To Data Science". In: *Foundations and Trends® in Machine Learning* 11.5-6 (2019), pp. 355–607.

[71] M Putinar. "Positive polynomials on compact semi-algebraic sets". In: *Indiana Univ. Math. J.* 42 (1993).

[72] Mihai Putinar. "Positive Polynomials on Compact Semi-algebraic Sets". In: *Indiana University Mathematics Journal* 42.3 (Feb. 1993). Full publication date: Fall, 1993, pp. 969–984. URL: http://www.jstor.org/stable/24897130.

[73] Lillian J. Ratliff, Samuel A. Burden, and S. Shankar Sastry. "Characterization and computation of local Nash equilibria in continuous games". In: *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 2013, pp. 917–924. DOI: 10.1109/Allerton.2013.6736623.

[74] Meisam Razaviyayn et al. "Nonconvex Min-Max Optimization: Applications, Challenges, and Recent Theoretical Advances". In: *IEEE Signal Processing Magazine* 37.5 (2020), pp. 55–66. DOI: 10.1109/MSP.2020.3003851.

[75] John Rehbeck. "Note on Unique Nash Equilibrium in Continuous Games". In: *Games and Economic Behavior* 110 (2018), pp. 216–225.

[76] Brian Roberson. "The Colonel Blotto game". In: *Economic Theory* 29.1 (Sept. 1, 2006), pp. 1–24. ISSN: 1432-0479. DOI: 10.1007/s00199-005-0071-5. URL: https://doi.org/10.1007/s00199-005-0071-5.

[77] Benjamin Roussillon and Patrick Loiseau. "Scalable Optimal Classifiers for Adversarial Settings under Uncertainty". In: *International Conference on Decision and Game Theory for Security*. Springer. 2021, pp. 80–97.

[78] A. Rubinstein. "Inapproximability of Nash Equilibrium". In: *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*. Portland, Oregon, USA: Association for Computing Machinery, 2015, pp. 409–418. DOI: 10.1145/2746539.2746578.

[79] Seyed Sina Mirrazavi Salehian, Nadia Figueroa, and Aude Billard. "A unified framework for coordinated multi-arm motion planning". In: *The International Journal of Robotics Research* 37.10 (2018), pp. 1205–1232. DOI: 10.1177/0278364918765952. URL: https://doi.org/10.1177/0278364918765952.

[80] Konrad Schmüdgen. "TheK-moment problem for compact semi-algebraic sets". In: *Mathematische Annalen* 289.1 (Mar. 1991), pp. 203–206. ISSN: 1432-1807. DOI: 10.1007/BF01446568. URL: https://doi.org/10.1007/BF01446568.

[81] N. Z. Shor. "An approach to obtaining global extremums in polynomial mathematical programming problems". In: *Cybernetics* 23.5 (Sept. 1987), pp. 695–700. ISSN: 1573-8337. DOI: 10.1007/BF01074929. URL: https://doi.org/10.1007/BF01074929.

[82] E.M.B. Smith and C.C. Pantelides. "A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs". In: *Computers & Chemical Engineering* 23.4 (1999), pp. 457–478. ISSN: 0098-1354. DOI: https://doi.org/10.1016/S0098-1354(98)00286-5. URL: https://www.sciencedirect.com/science/article/pii/S0098135498002865.

[83] Mark W Spong, Seth Hutchinson, and M Vidyasagar. *Robot Modeling and Control*. 2nd ed. Standards Information Network, Feb. 2020.

[84] N. D. Stein, A. Ozdaglar, and P. A. Parrilo. "Separable and Low-rank Continuous Games". In: *International Journal of Game Theory* 37.4 (2008), pp. 475–504.

[85] Noah D. Stein, Asuman Ozdaglar, and Pablo A. Parrilo. "Separable and low-rank continuous games". In: *International Journal of Game Theory* 37.4 (Dec. 2008), pp. 475–504. ISSN: 1432-1270. DOI: 10.1007/s00182-008-0129-2. URL: https://doi.org/10.1007/s00182-008-0129-2.

[86] Noah D. Stein, Pablo A. Parrilo, and Asuman Ozdaglar. "Characterization and computation of correlated equilibria in infinite games". In: *2007 46th IEEE Conference on Decision and Control*. 2007, pp. 759–764. DOI: 10.1109/CDC.2007.4434890.

[87] B. Sturmfels. *Solving Systems of Polynomial Equations*. Conference Board of the Mathematical Sciences Regional Confe. Conference Board of the Mathematical Sciences, 2002. ISBN: 9780821832516. URL: https://books.google.cz/books?id=ULtVBQAAQBAJ.

[88] S. Surjanovic and D. Bingham. *Virtual Library of Simulation Experiments: Test Functions and Datasets*. Retrieved August 18, 2025, from http://www.sfu.ca/~ssurjano.

[89] G. Szegő. *Orthogonal Polynomials*. American Math. Soc: Colloquium publ. American Mathematical Society, 1975. ISBN: 9780821810231. URL: https://books.google.cz/books?id=ZOhmnsXlcY0C.

[90] Pavel Trutman et al. "Globally optimal solution to inverse kinematics of 7DOF serial manipulator". In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 6012–6019.

[91] S. Karlin W. Dresher and L. S. Shapley. “Polynomial Games”. In: *Contributions to the Theory of Games*. Ed. by H. W. Kuhn and A. W. Tucker. Vol. I. Annals of Mathematics Studies 24. Princeton University Press, 1950, pp. 161–180.

[92] Andreas Wächter and Lorenz T. Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical Programming* 106 (2006), pp. 25–57.

[93] Yuanhao Wang, Guodong Zhang, and Jimmy Ba. *On Solving Minimax Optimization Locally: A Follow-the-Ridge Approach*. 2019. arXiv: 1910.07512 [cs.LG]. URL: https://arxiv.org/abs/1910.07512.

[94] T. Weisser et al. *Polynomial and Moment Optimization in Julia and JuMP*. Presented at JuliaCon. https://pretalx.com/juliacon2019/talk/QZBKAU. 2019.

[95] Lily Xu et al. “Robust reinforcement learning under minimax regret for green security”. In: *Uncertainty in Artificial Intelligence*. PMLR. 2021, pp. 257–267.

[96] Sarath Yasodharan and Patrick Loiseau. “Nonzero-sum Adversarial Hypothesis Testing Games”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 7310–7320.

[97] Sarath Yasodharan and Patrick Loiseau. “Nonzero-sum adversarial hypothesis testing games”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[98] Taoli Zheng et al. “Universal gradient descent ascent method for nonconvex-nonconcave minimax optimization”. In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. NIPS ’23. New Orleans, LA, USA: Curran Associates Inc., 2023.

[99] Guangming Zhou, Qin Wang, and Wenjie Zhao. “Saddle points of rational functions”. In: *Computational Optimization and Applications* 75.3 (Apr. 2020), pp. 817–832. ISSN: 1573-2894. DOI: 10.1007/s10589-019-00141-6. URL: https://doi.org/10.1007/s10589-019-00141-6.

Notation

$\mathbb{R}[\mathbf{x}]$	Polynomial ring in multiple variables over the real numbers
$\mathbb{R}[\mathbf{x}]'$	Dual of $\mathbb{R}[\mathbf{x}]$
$\mathbb{N}^m_d$	Set of multi-indices truncated to degree d in m dimensions
$\mathcal{S}(\mathbf{g})$	Basic semialgebraic set defined by polynomials in $\mathbf{g}$
$\mathcal{M}(K)$	Set of regular Borel probability measures on K
$\mathcal{M}_d(K)$	Set of moments up to degree d with a representing measure on K
$\mathcal{P}(K)$	Set of polynomials nonnegative on K.
$\mathcal{P}_d(K)$	Set of polynomials up to degree d which are nonnegative on K
$\mathcal{Q}(\mathbf{g})$	Quadratic module generated by polynomials in $\mathbf{g}$
$\mathcal{Q}_d(\mathbf{g})$	Quadratic module generated by polynomials in $\mathbf{g}$ truncated to degree d
$M_d(\mathbf{y})$	Moment matrix of order d
$M_d(g, \mathbf{y})$	Localizing matrix of order d
$\mu_i \in \mathcal{M}(X_i)$	Mixed strategy in the strategy set of Player i
$\boldsymbol{\mu} \in \mathbf{M}$	Profile of mixed strategies
$\boldsymbol{\mu}_{-i} \in \mathbf{M}_{-i}$	Mixed strategies of all players except i
$u_i(\mathbf{x})$	Utility of Player i given a pure strategy profile $\mathbf{x}$
$U_i(\boldsymbol{\mu})$	Expected utility of Player i given a mixed strategy profile $\boldsymbol{\mu}$
$\boldsymbol{\mu}^\star$	Epsilon Nash-Equilibrium
$T(\theta)$	Denavit–Hartenberg Matrix at angle θ