Games, Fair and Square: Constructive Approaches to Equilibrium Stability, Fairness and Computation

Dhivya Anand Kumar

A thesis submitted for the degree of
Doctor of Philosophy

Adam Smith Business School
University of Glasgow

May 2026

Abstract

Game theory has evolved from analysing simple strategic interactions to addressing the computational and normative challenges of complex multi-agent systems. This thesis advances three interconnected dimensions of modern game theoretic analysis: the stability of equilibria under perturbations, the algorithmic discovery of equilibrium structures in large scale games, and the design of fair mechanisms for resource allocation. Together, these contributions bridge theoretical foundations with computational practice, offering constructive solutions to fundamental problems in strategic decision making.

The first contribution revisits hyperstability, a refinement concept for equilibria robust against structural perturbations. Building on Hauk and Hurkens (2002), we propose a systematic alternative to adding complex mixed strategies, which they do to eliminate, following perturbations, certain equilibrium components; this alternative utilises the addition of pure strategies, along with binary, enforcing players. By introducing pure strategy duplicates paired with auxiliary disciplinary players using a construction inspired by *Levy* games (Levy, 2016), we enforce we enforce an imitation of mixed strategies through strategic interactions. This framework avoids manual mixture calculations, generalises to N-player games, and provides a transparent method for identifying hyperstable equilibria.

The second contribution addresses equilibrium discovery in Polymatrix Games. We establish that determining the existence of a pure strategy Nash equilibrium in these games is NP-complete. To navigate this complexity, we show that a specific parametrisation of Lemke (1965)'s algorithm implements the Linear Tracing Procedure (van den Elzen and Talman, 1999), enabling principled computation from arbitrary priors. To overcome the bias of forward tracing toward dominant basins, we introduce the *Tracing Backwards* algorithm. By using discovered equilibria as "seeds" for breadth-first exploration, we uncover a novel property termed fertility: the capacity of an equilibrium to lead to new discoveries. Our analysis reveals that while pure equilibria

are easy to find, they exhibit low fertility compared to completely mixed equilibria, which act as critical hubs in the strategic landscape.

The third contribution evaluates fair division problems involving indivisible goods and monetary transfers. We conduct an empirical comparison of three mechanisms: Bundled Auction (BA), *Bid & Sell* (*B*&*S*), and *Divide & Choose* (*D*&*C*). Our simulations demonstrate that mechanism responsiveness, i.e., the ability to adapt to heterogeneous agent preferences, is the primary driver of social welfare. While *D*&*C* performs well in additive environments with independent valuations, *B*&*S* emerges as the superior mechanism in complex settings with strong complementarities or substitution effects. Its continuous pricing structure effectively captures "unbundling surplus" that discrete mechanisms fail to exploit, proving that the computational overhead of sophisticated mechanisms yields substantial welfare gains, validating their practical value in real world allocation problems.

Collectively, this thesis offers a constructive journey through equilibrium analysis: from establishing when equilibria are strategically robust, to developing algorithms that systematically map equilibrium manifolds in large games, to designing mechanisms that efficiently and fairly allocate scarce resources. These contributions address theoretical, computational, and applied challenges in modern game theory, providing tools and insights for analysing strategic interactions in increasingly complex multi agent systems.

Contents

List of Figures

List of Tables

Acknowledgements

I feel incredibly fortunate to have had supervisors who cared as much for my wellbeing as for my research.

I owe my biggest thanks to Dr John Levy for taking me under his wing, starting from the earliest proofreading days and continuing through to today. Being his student has been one of the most enjoyable and meaningful experiences of my academic journey. I am grateful for our lighthearted conversations, his patience during moments of struggle and constant encouragement. He has been more than a supervisor to me; he has been a mentor, a guardian and a friend. Thank you John!

I will always be inspired by Professor Hervé Moulin's curiosity, and I hope to carry forward his wonderful zest for work and life. During a difficult time, he once kindly told me, "*If you were my daughter, I'd say go to the Highlands and stay until you feel better.*" His compassion meant a lot more than he may know.

I thank Professor Bernhard von Stengel for his enthusiasm and friendship. From whiteboard discussions to shared lunches, it was a genuine pleasure to work on our project together.

My sincere gratitude to the University of Glasgow and the Adam Smith Business School for fostering a supportive environment for my research. Gentle thanks to Irene, for dragging me to writing retreats. Zhou Zhou started this journey with me - thank you for always wishing me well. I'm so glad we made it together.

With much fondness to Rémy, for the silly fun and all the joy we shared, les moments de tendresse, and for being present when I needed it most.

My love to Jeanne, Amhed, and Timi - having them meant I had a family in Glasgow, the unconditional presence and love and respect, the feeling of being deeply rooted and of coming home.

Amma, Appa, Subi, and Aditya - truly incredible people, my best friends, my strength. I know you are always with me, no matter where you are. Thank you for believing in me and being proud of me, no matter what I do.

Declaration

I declare that, except where explicit reference is made to the contribution of others, that this dissertation is the result of my own work. Chapter 3 is joint work with Bernhard von Stengel.

This dissertation has not been submitted for any other degree at the University of Glasgow or any other institution.

Dhivya Anand Kumar

Chapter 1

Introduction

The foundations of game theory rest on the concept of equilibrium configurations of strategies where no agent can unilaterally improve their position. Since Nash's seminal contributions, equilibrium analysis has evolved from a primarily descriptive tool into a computational and prescriptive framework for understanding strategic interaction in economic, political, and digital systems. Yet three fundamental challenges persist at the frontier of game theoretic research: ensuring that predicted equilibria are strategically stable under perturbations, developing algorithms capable of systematically discovering equilibria in complex games, and designing mechanisms that translate equilibrium predictions into fair and efficient outcomes.

This thesis addresses these challenges through three interconnected contributions that span theoretical refinement, algorithmic innovation, and empirical mechanism design. The unifying theme is *constructivism*: we develop practical frameworks for eliminating non-robust equilibria, systematically mapping equilibrium structures, and implementing fair allocation rules. Each chapter tackles a distinct problem while contributing to a coherent vision of game theory as both rigorous mathematical theory and actionable computational practice.

Equilibrium Stability through Hyperstability Refinement Not all Nash equilibria exhibit equal strategic robustness. We may demand that an equilibrium, or equilibrium component, satisfies the additional requirement that profitable deviations are not possible for any player even following small changes in game structure, such as introducing new outside options or expanding the strategy space. Hyperstability, introduced by Kohlberg and Mertens (1986) and discussed in Hauk and Hurkens (2002), provides a refinement concept that identifies equilibria robust to such perturbations.

Their key insight was that certain non-hyperstable equilibria can be eliminated by adding carefully chosen mixed strategies as new pure actions *and then perturbing*, forcing players to reconsider their strategic commitments.

However, the original construction requires identifying the *correct* mixture to add. This is a process that is often complex and game-specific, offering limited guidance for systematic application. Chapter 2 develops an alternative approach that transforms the mixture-addition procedure into a more transparent construction. Instead of adding mixed strategies directly, we introduce duplicate pure strategies paired with *additional* auxiliary players whose payoffs are structured using Levy (2016) game principles. These auxiliary players act as disciplinary mechanisms, coordinating their play to simulate the desired mixing ratios through pure strategy interactions. This reformulation achieves the same equilibrium elimination effect as the original mixed strategy addition, but in a form that is more readily verified and generalises naturally to N-player settings. The result is a constructive framework for hyperstability analysis that replaces opaque mixture calculations with explicit game-theoretic structures.

Algorithmic Discovery of Equilibrium Manifolds As game theoretic models grow in complexity, particularly when representing network interactions among many agents, the computational challenge shifts from existence proofs to practical discovery. Polymatrix games provide an elegant framework for such problems: each player's utility is the sum of payoffs from separate bilateral games with their neighbours, allowing compact representation of otherwise intractable normal form games. Despite this structure, Chapter 3 establishes that even determining whether a pure strategy Nash equilibrium exists is *NP-complete*, confirming the inherent computational difficulty of these models. For mixed equilibria, we turn to homotopy methods, specifically implementing the *Linear Tracing Procedure* (LTP) through a parametrised version of Lemke's algorithm. LTP constructs a path from an arbitrary *prior* belief to an equilibrium, providing a principled alternative to naive fixed-point iteration. However, our experiments reveal a critical observation: forward tracing from random starting points exhibits strong convergence bias, repeatedly finding the same equilibria (typically pure or nearly-pure strategies with large basins of attraction) while systematically needing many random samples to reach more complex, completely mixed solutions.

To address this discovery gap, we develop the *Tracing Backwards* algorithm. Rather than always starting from random priors, we use previously discovered equilibria as new starting points, effectively reversing the tracing direction to explore adjacent regions of the equilibrium manifold. This breadth-first exploration strategy

systematically maps the global structure of equilibria, treating the equilibrium set as a connected network rather than isolated points. The backward tracing approach reveals a striking property we term fertility: the number of distinct equilibria discovered from a given starting equilibrium. Our empirical results show that pure equilibria, despite being easy to find from random starts, exhibit remarkably low fertility. In contrast, completely mixed equilibria, though harder to discover initially, often serve as high-fertility hubs that unlock access to multiple distinct strategic regions. This finding reframes equilibrium computation as a graph exploration problem, where the goal is not merely to find any equilibrium but to systematically characterise the manifold's topology.

Fair Division with Indivisible Goods While the first two contributions focus on identifying and computing equilibria, Chapter 4 examines how game theoretic mechanisms translate into practical allocation rules. The fair division of indivisible goods, for example, dividing an inheritance, assigning rooms in a shared apartment or allocating schedules presents a fundamental challenge: without the possibility of fractional splits, achieving envy-freeness requires monetary transfers to compensate for disparities in assigned bundles.

We compare three mechanisms that guarantee envy-free outcomes through different pricing and partition structures: *Bundled Auction* (BA), *Bid & Sell* (*B*&*S*), and *Divide & Choose* (*D*&*C*). The central question is mechanism responsiveness: how effectively does each rule adapt to heterogeneity in agent preferences? An ideal mechanism should extract surplus from *greedy* agents with strong complementarities while rewarding *frugal* agents who are easy to satisfy. Through extensive randomised simulations across diverse utility landscapes, we demonstrate that BA's one-size-fits-all approach fails to capture significant social surplus, particularly when agents exhibit complex substitution or complementarity patterns. *D*&*C* performs well in simple additive environments where goods are valued independently, benefiting from its coarse partition structure. However, *B*&*S* emerges as the superior mechanism in heterogeneous settings. Its continuous pricing structure allows fine-grained price discrimination, enabling the mechanism to capture *unbundling surplus*: value that arises when breaking apart bundles exposes hidden substitution or complementarity effects. Our results demonstrate that the additional computational cost of implementing more sophisticated mechanisms yields substantial welfare gains, with *B*&*S* consistently outperforming simpler alternatives in complex environments. This finding validates the practical importance of mechanism design theory: investing in

sophisticated allocation procedures is not merely an academic exercise but a concrete path to improving real world outcomes.

Thesis Organisation and Contributions The three chapters of this thesis form a progression from theoretical foundations to computational methods to practical implementation. Chapter 2 establishes a constructive framework for equilibrium refinement, replacing ad-hoc calculations with systematic game-theoretic constructions. Chapter 3 develops algorithms for comprehensive equilibrium discovery, transforming the problem from finding any solution to mapping the entire solution manifold. Chapter 4 evaluates mechanism performance in realistic settings, demonstrating how theoretical principles translate into measurable improvements in social welfare.

Collectively, these contributions advance a constructive vision of game theory: strategic analysis should not only characterise equilibria abstractly but provide concrete tools for identifying robust solutions, computing them systematically, and implementing them fairly. As multi-agent systems become increasingly central to economic, social, and computational domains, this constructive approach ensuring our models are stable, our algorithms are thorough, and our mechanisms are just becomes ever more essential.

Chapter 2

On Hyperstability using Duplicate Pure Actions and Additional Players

2.1 Introduction

While the classic Nash equilibrium serves as the bedrock of non-cooperative game theory, it frequently suffers from a severe operational limitation: it can generate a multiplicity of equilibria, many of which rely on strategically unconvincing behaviour, such as incredible threats or irrational out-of-equilibrium beliefs. To isolate economically meaningful predictions, the literature utilises *equilibrium refinements* Hillas and Kohlberg (2002). One class of refinements, of particular relevance to this chapter, are filters designed to assess the structural robustness of an equilibrium when the ideal conditions of the game are subjected to micro-shocks.

Conceptually, these models capture real-world economic interactions characterised by minor operational friction, asymmetric information, or strategic uncertainty. For instance, in commercial wage bargaining, market entry deterrent scenarios, or multi-stage network auctions, agents cannot realistically assume that their opponents will execute decisions with absolute, error-free perfection. Instead, they must evaluate whether a chosen strategy profile remains self-enforcing if an opponent occasionally slips, miscalculates, or is faced with an unexpected structural detour, such as a baseline outside option van Damme (1989); Hauk and Hurkens (2002). By requiring equilibria to withstand small perturbations in choices or payoffs, refinements ensure that the selected strategic paths rely solely on credible, forward-looking rationality, providing policy-makers and market architects with highly stable, predictive benchmarks.

Within the context of non-cooperative games, there has been a great deal of work on the stability of Nash equilibria since the seminal work by Kohlberg and Mertens (1986). For our purposes, in order to compare and contrast the stability notions relevant to this chapter, we describe them heuristically to guide the reader through the chapter.

An equilibrium set is *strategically stable* if it persists under small perturbations to the game's structure. This notion encompasses the intuition that players may occasionally commit minor errors in implementing their intended strategies (formalised, for example, by the trembling hand perfect equilibria); if all players independently face a small probability of such execution errors, the perturbed game should always contain an equilibrium close to the original unperturbed set. It is well known that not all Nash equilibria are strategically stable.

A component is *hyperstable* if it exhibits an even stronger form of robustness: it remains self-enforcing when the underlying payoff matrix itself is perturbed. Rather than restricting attention to execution errors, hyperstability requires more than shifts that alter the true payoff values; it also changes strategy representation. Thus, hyperstability represents a stringent form of resilience to fundamental perturbations in the game's parameters such as perturbations to payoffs as well as perturbations to players' strategies, i.e., by introducing additional pure strategies that are convex combinations of their original pure strategies.

In an analysis evaluating how strategic stability concepts interact with forward induction reasoning (that is, if a player can determine through reasoning about the extensive form structure what outcome will result, the equilibrium must be consistent with this inference), Hauk and Hurkens (2002) use a normal form game preceded by an outside option. In this game, there are two distinct sets of equilibria: the forward induction equilibrium and the outside option component, a split that arises as a result of the most viable equilibrium e^* of the normal form game paying the players more than the outside option. The forward induction concept as defined in van Damme (1989) dictates that the only plausible solution is for the player to choose to play the game and realise e^*. In this specific example, when one agent's actions are duplicated by adding a copy of a specific mixed strategy as pure to the agent's action set, the sensitivity of equilibria to perturbation changes as compared to their sensitivity of pertubations of payoffs of the original game. The strategically essential set of Nash equilibria (i.e., those equilibria robust to small payoff perturbations) of the unperturbed game includes those with the outside option outcome. If the strategies are perturbed by adding an additional pure option (which is a mix of two

pure actions from the player's set of actions) followed by a perturbation of payoffs for one of the players, all outside option outcomes are eliminated and the forward induction equilibrium is chosen uniquely. Hauk and Hurkens also present another game with an outside option as a counterexample to show that not all hyperstable equilibria are forward induction equilibria.

The equivalent game in the earlier example, constructed by incorporating a mixed strategy as an additional pure strategy, serves as the starting point for this work. While this technique has been used to verify the hyperstability of equilibria in certain non-cooperative games, identifying the appropriate mixed strategy is not always straightforward. This chapter proposes an alternative, generalisable approach: instead of adding specific mixed strategies, we introduce duplicates of existing pure strategies and additional players whose role is to adjust the original player's strategic choices in the resulting game. This reformulation aims to facilitate the analysis of hyperstability across a broader class of games. The complexity of how the game is perturbed is simplified if the same outcomes can be achieved with duplicate pure strategy additions. Our results in this chapter demonstrate that it is indeed possible to use pure duplicates in place of a mixed duplicate strategy in many cases. We further add additional layers to the construction of an equivalent game, employing the help of the additional players and games to simulate desired weights on these duplicate pure strategies to achieve the same result as in Hauk and Hurkens (2002). We start with the normal form of a game in which some equilibria can be eliminated by adding redundant strategies. To begin with the simplest case, we assume a singleton equilibrium of such a game with two players is eliminated by adding to the action set of a player a single mixture of precisely two pure strategies.

The structure of the subsequent sections is designed to lay out the theoretical foundations and then progressively develop the analysis. Section 2.2 surveys the existing literature relevant to our study, situating our contribution within the broader field. Section 2.3 establishes the preliminaries, beginning with key definitions and formal notation, and proceeding through a game from Hauk and Hurkens (2002) and a game from Levy (2016), which serve as central examples in our analysis. Section 2.4 focuses on the demonstration of hyperstability in two-player games, considering cases involving adding simple mixed strategies i.e. a new strategy that is a mixture of just two actions, which is then replaced by pure duplicates, results in the elimination of non-hyperstable equilibrium components, and the extension to an equilibrium component, as well as to a setting with more players. Sections 2.5 and 2.6 extend these insights to the N-player setting, demonstrating that non-hyperstable equilibrium components can be eliminated through the iterative introduction of pure duplicates,

thereby obviating the need for complex mixtures previously required to achieve the same outcome.

2.2 Literature

The theoretical context for this work is rooted in the extensive study of non-cooperative games, particularly concerning the stability of Nash equilibria, a topic reignited by the seminal work of Kohlberg and Mertens (1986). While standard strategic stability allows for payoff perturbations, a more stringent concept, *hyperstability*, is required here. Hyperstability extends the class of perturbations to include the introduction of additional pure strategies that are convex combinations of original strategies, resulting in a smaller, more robust set of equilibria. This concept is closely related to the strategically essential sets studied by Govindan and Wilson (2005), which offer sufficient conditions for stable equilibria.

A central difficulty in working with the foundational literature arises from significant terminology inconsistencies. As noted in Govindan and Wilson (2005) and Hillas (1990), the same terms often denote distinct mathematical structures across different frameworks. For instance, *essential* sometimes refers to payoff robustness in the sense of Hauk and Hurkens (2002), while in other contexts, particularly in Govindan and Wilson (2005), it denotes a projection mapping with non-zero index: a topologically distinct property. While non-zero index is equivalent to uniform hyperstability (Govindan and Wilson, 2005), a slight strengthening of hyperstability, it is an open question as to whether hyperstability is equivalent to being a component of non-zero index.

Similarly, the term *hyperstable set* carries different meanings across the literature. In Kohlberg and Mertens (1986) and Hillas (1990), a hyperstable set is required to be minimal with respect to its robustness properties. That is, no proper subset of the set retains the same degree of stability. In contrast, our treatment of hyperstability as given in 2.3.2, following Govindan and Wilson (2005), evaluates the robustness property of a component without imposing a minimality constraint. We adopt this latter definition to maintain consistency with our constructive approach, but the reader should be aware of this divergence from classical treatments.

To contextualise our hyperstability framework, it is necessary to contrast it with classic error-based refinements like *trembling-hand* perfection[1] (Selten, 1975) and ϵ-properness (Myerson, 1978) analyses how equilibria behave when players commit small implementation mistakes. A perfect equilibrium is one that remains robust when every pure action receives positive probability weight, however small, in the perturbed game. As Kohlberg and Mertens (1986) note, these concepts satisfy several desirable axiomatic properties: existence, a connectedness condition, consistency with backward induction, invariance, and iterated elimination of dominated strategies.

Despite these strengths, hyperstability operates on a fundamentally different principle. Rather than model behavioural error (mistakes players make in executing a fixed game) hyperstability tests whether an equilibrium survives when the game itself is perturbed. That is, we ask whether an equilibrium component remains robust to small changes in not just payoffs, but strategy structure. As demonstrated by Kohlberg and Mertens, this payoff-perturbation approach is strictly stronger than error-based refinements: any hyperstable equilibrium is automatically perfect, proper, and sequential.

This distinction yields two important consequences. First, error-based refinements mandate full-support strategies in approximations (every action played with positive probability), whereas hyperstability does not. By perturbing payoffs rather than constraining action choices, a hyperstable equilibrium can remain robust even when entire blocks of pure strategies are unplayed in equilibrium. Second, the two approaches address fundamentally different stability questions: error-based refinements ask whether an equilibrium survives small behavioural perturbations in a fixed environment, while hyperstability asks whether it survives small structural changes to the game itself. Our framework emphasises the latter: that an equilibrium should be robust to the discovery that payoff values are slightly different from what was believed.

The starting point for this chapter is the work of Hauk and Hurkens (2002), which shows that not all hyperstable equilibria are forward induction equilibria. The authors demonstrated, using a normal form game preceded by an outside option, how an equilibrium component corresponding to the outside option could be eliminated by adding a specific mixed strategy as a pure action for one player, uniquely selecting the forward induction equilibrium. Our methodology also leverages the general techniques from Levy (2016), specifically on the realisation of functions as projections of the equilibrium correspondence of auxiliary games. These *Levy* games provide the

[1] Crucially, as noted by Kohlberg and Mertens (1986) in Footnote 5, they drop the prefix 'trembling-hand' and refer to this concept simply as a *perfect* equilibrium.

necessary disciplinary mechanism, allowing us to simulate desired mixed strategies over duplicate pure actions through the introduction of additional players, thereby providing a generalisable alternative to the complex mixed-strategy additions used previously. The overall analysis is positioned within the general stability literature, which also relates to concepts from van Damme (1989) regarding stable equilibria and forward induction, as well as the topological and index theory foundations underlying these stability concepts.

2.3 Preliminaries

2.3.1 Games and Spaces of Strategies

Throughout this chapter, we work in the space $\Sigma = \prod_{i \in I} \Delta(A_i)$ of mixed strategy profiles, where $\Delta(A_i)$ denotes the simplex of probability distributions over player i's pure strategies. This space is equipped with the Euclidean topology of $\mathbb{R}^k$, where $k = \sum_{i \in I} |A_i|$. An *open neighbourhood* V of a set $S \subseteq \Sigma$ is an open set in this relative topology containing S. Formally, V is an open neighbourhood of S if V is open in Σ and $S \subseteq V$. Equivalently, for each $s \in S$, there exists an $\epsilon > 0$ such that the open ball $B_\epsilon(s) = \{s' \in \Sigma : \|s' - s\| < \epsilon\}$ is entirely contained within V. Before introducing equilibrium refinements, we formally establish the baseline notation for a standard finite game in strategic form.

Strategic Game: A game is a tuple $G = (N, (A_i)_{i \in N}, (U_i)_{i \in N})$, where $N = \{1, \ldots, n\}$ is the set of players, A_i is the finite set of pure strategies for player i, and $U_i : \prod_{j \in N} A_j \to \mathbb{R}$ is the payoff function of player i.

Mixed Strategies and Profiles: A mixed strategy $x_i \in \Delta(A_i)$ is a probability distribution over A_i. The space of all mixed strategy profiles is denoted by $\Sigma = \prod_{i \in I} \Delta(A_i)$, where the expected payoff $U_i(x)$ for a profile $x \in \Sigma$ is the multilinear extension of the pure payoff function.

Nash Equilibrium: A mixed strategy profile $x^* = (x_i^*, x_{-i}^*) \in \Sigma$ is a *Nash equilibrium* if, for every player $i \in N$ and all $x_i \in \Delta(A_i)$:

$$U_i(x_i^*, x_{-i}^*) \geq U_i(x_i, x_{-i}^*)$$

The full set of Nash equilibria for the game G is denoted by $E(G) \subseteq \Sigma$.

2.3.2 Definitions

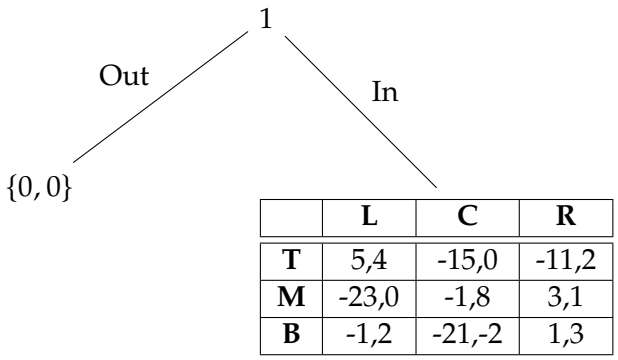
1. *Equivalent games:* Two (pure or mixed) strategies of one player are equivalent if they yield every player the same expected payoff for each profile of others' strategies. z_1 and z_2 are equivalent if for each Player i and j and each $x_{-i} \in \prod_{j \neq i} \Delta(A_i)$, $u_j(z_1, x_{-i}) = u_j(z_2, x_{-i})$. A pure strategy is redundant if the player has another (pure or mixed) strategy that is equivalent. From a game G one obtains its reduction G^* by deleting redundant pure strategies (that is, strategies for which an equivalent pure or mixed strategy exists) until none remain; the reduction is unique (apart from names of pure strategies). Two games are equivalent if their reductions are the same. If σ is a profile of players' strategies in G then its reduction σ^* is the profile of equivalent strategies of G^*. For each set C of strategy profiles for game G the corresponding set C_0 for an equivalent game G_0 consists of the profiles of equivalent strategies (Govindan and Wilson, 2005).

2. *Strategically Essential set:* A closed set of Nash equilibria S is strategically essential if for any small payoff perturbation of the normal form, there exists a Nash equilibrium close to S. Formally, for each neighbourhood V of S, there is $\varepsilon > 0$, such that for any game G' with payoffs $|G'_i(s) - G_i(s)| < \varepsilon$ for all players i and strategy profiles s, the set of Nash equilibria of G' intersects V. A Nash equilibrium s^* is strategically essential if the set $\{s^*\}$ is strategically essential.

3. *Hyperstability:* A closed set S of Nash Equilibria of game G is hyperstable if, for any game G' equivalent to G, the set S' of Nash equilibria of G' equivalent to S is strategically essential. Every game has a hyperstable set of equilibria contained in a single connected component of the set of Nash equilibria (Kohlberg and Mertens, 1986).

4. *Extreme equilibria:* The set of Nash equilibria of a bimatrix game is a finite union of products of convex polytopes. The vertices (extreme points) of these polytopes are called extreme equilibria (see e.g. Audet, Hansen, Jaumard, and Savard (2001) and the references within).

5. *Pseudo-game*: A pseudo-game, with a set of players, each with a set of available actions, is one in which each player's payoff U_i is defined so that it depends on the mixed actions of the other players, σ_{-i}, in a general continuous way. That is, $(N, (A_i), (\tilde{U}_i))$, where $\tilde{U}_i : \prod_i \Delta(A_i) \to \mathbb{R}$, such that $\tilde{U}_i$ is affine only in strategy of Player i. This generalises the simple linear expectation calculation over pure strategies typically found in a standard normal-form game.



	L	C	R
T	5,4	-15,0	-11,2
M	-23,0	-1,8	3,1
B	-1,2	-21,-2	1,3

Figure 2.1 HH game in extensive form

	L	C	R
T	5,4	-15,0	-11,2
M	-23,0	-1,8	3,1
B	-1,2	-21,-2	1,3
Out	0,0	0,0	0,0

Table 2.1 HH game: normal form

2.3.3 Hauk and Hurkens Game

One approach presented in Hauk and Hurkens (2002), henceforth *HH*, is the starting point of this chapter. *HH*, in their paper, analyse solution concepts that capture the extensive form induction using examples from a class of generic two Player normal form games preceded by an outside option. In an example, *HH* demonstrate a game with the outside option, whose normal form is given in Table 2.1, to show that the game admits two equilibrium components, each of which is strategically essential, but only one of which is hyperstable equilibrium (see definitions).

If the game is considered without an outside option (*Out*), the only Nash equilibrium that pays the row player a better payoff compared the outside option is (T,L). As (T,L) is strict it constitutes a singleton stable strategically essential set. However, when the game includes an outside option, the resulting game admits a strategically essential set where the outside option is chosen. There are mixed strategies for Player 2 for which '*Out*' becomes the best reply for Player 1. It can be shown that there is no equilibrium in which Player 1 strictly mixes between '*Out*' and his other strategies.

Extreme equilibria	Payoffs
Out, (31/282, 10/282, 241/282)	0,0
Out, (0,3/4,1/4)	0,0
(Out, C)	0,0
Out, (3/4, 1/4, 0)	0,0
Out, (1/2, 0, 1/2)	0,0
Out, (11/16, 0, 5/16)	0,0
(T, L)	5,4

When normal form game is perturbed by a small ϵ, it is shown in Hauk and Hurkens (2002) that an equilibrium component where Player 1 plays pure '*Out*' persists, thus establishing that '*Out*' is payoff robust (part of a strategically essential set). Nevertheless, if such an equilibrium component were to be hyperstable, the component should also be a solution to equivalent games, obtained by adding a mixed strategy as an additional pure strategy. In this specific example, a new action $d = \frac{61}{81}L + \frac{20}{81}C$ is added to Player 2's set of actions, followed by a perturbation to the payoffs (in this case, only the payoffs of the strategy profile (*Out*, *d*)) resulting in the following game:

	L	C	R	d
T	5,4	-15,0	-11,2	5/81, 244/81
M	-23,0	-1,8	3,1	-1423/81, 160/81
B	-1,2	-21,-2	1,3	-481/81, 82/81
Out	0,0	0,0	0,0	0, ϵ

When $\epsilon = 0$, there are 11 extreme equilibria, 10 of which have Player 1 choosing *Out*. HH shows that when $\epsilon > 0$, there is a unique equilibrium: (T,L). '*Out*' is not part of the hyperstable set of equilibria of the game, as it is not part of the strategically essential set of every equivalent game. That is, there exists a neighbourhood U containing all equilibria of the original game where Player 1 plays *Out* with positive probability (precisely, all equilibria except (T,L)), as well as equilibria of sufficiently small perturbations of the game, but such that by adding a duplicate mixed action for Player 2 and by perturbing the game slightly, the component with Player 1 playing *Out* are eliminated so that there are no equilibria in the equivalent neighbourhood U' of the modified game.

2.3.4 Levy Game

The main theorem from Levy (2016) provides a tool to our extension of the approach used in Hauk and Hurkens (2002). For the purposes of this chapter, we only utilise the specific matching pennies game example, H_q from Levy (2016). Both the general result and the example game from Levy (2016) are presented below.

Theorem 3.2 (Levy, 2016):
Let $A \subseteq \mathbb{R}^N$ be bounded and semi-algebraic; and let $f : A \to [0,1]^K$ be a continuous semi-algebraic function. Then there exists an $\mathbb{R}^N$- parametrised game G on a set of binary players $\{\alpha_1, ..., \alpha_K\} \cup J$ such that for each $x \in A$, in any equilibrium z of $G[x]$, we have $(z^{\alpha_1}, ..., z^{\alpha_K}) = f(x)$.

We clarify the notation and concepts in Theorem 3.2. A set $A \subseteq \mathbb{R}^N$ is *semi-algebraic* if it can be expressed as a finite union of sets of the form

$$\left\{x \in \mathbb{R}^N \mid p_1(x) = 0, \ldots, p_r(x) = 0,\ q_1(x) \geq 0, \ldots, q_s(x) \geq 0\right\},$$

where $p_1, \ldots, p_r$ and $q_1, \ldots, q_s$ are real polynomial functions on $\mathbb{R}^N$. For instance, a closed interval $[0,1]$ or a polytope is semi-algebraic. A *semi-algebraic function* $f : A \to [0,1]^K$ is a continuous function whose graph $\{(x, f(x)) : x \in A\}$ is a semi-algebraic set. An $\mathbb{R}^N$- *parametrised game* G is a game whose payoff structure depends affinely in each coordinate on a parameter $x \in \mathbb{R}^N$; the notation $G[x]$ denotes the game with parameter value x. *Binary players* are players with exactly two actions available (hence "binary" in the sense of binary choice). In an equilibrium z of $G[x]$, for any player i, $z^i \in [0,1]$ denotes the probability that Player i assigns to one of their two actions (with probability $1 - z^i$ on the other action). Theorem 3.2 asserts that for any target function f, we can construct a $\mathbb{R}^N$-parametrised game where the equilibrium play of the subcollection of binary players exactly matches the output of f on the parameter x, i.e., we have singled out a collection $\alpha^1, \ldots, \alpha^k$ $(z^{\alpha_1}, \ldots, z^{\alpha_K})$ of mixed strategies they play in equilibrium z of $G[x]$ is precisely $f(x)$. The particular simple example of this theorem (as presented in Levy (2016)) given below is the leverage in our construction to discipline the mixing behaviour of players over duplicate pure actions.

$H_q =$

-1,1	4q-1, 4q-3
3-4q, 1-4q	-1,1

Added as an auxillary game, this game ensures that the player with duplicate actions mixes between the duplicates in the precise probability distribution ($\frac{61}{81}$, $\frac{20}{81}$ for

the game in 2.3.3) required to replicate the equilibrium-eliminating effect. At $q = \frac{1}{2}$, this becomes the matching pennies game. For $0 < q < 1$, the unique equilibrium is $(q, 1-q) \bigotimes (q, 1-q)$; for $q \in \{0, 1\}$, the set of equilibria is $\{ (q, 1-q) \bigotimes (w, 1-w) | w \in [0, 1]\}$.

2.3.5 Notation

To ensure clarity and consistency throughout the paper, this section introduces the notation and terminology used in the analysis using the table below.

Table 2.2 Summary of notation

Symbol	Description
Base Game Setup	
B	Base game with 2 players; has a singleton equilibrium.
$e = (x^*, y^*)$	Singleton equilibrium of B that is payoff-robust.
U	Neighbourhood around e^*; contains no other equilibria.
$\tilde{}$	A $\sim$ over a game means there has been an ϵ-perturbation.
$\hat{}$	A $\hat{}$ over a game refers to auxiliary pseudo-games which are perturbed by a function.
Strategy Duplication: Mixed	
$d = p^* a_1 + (1 - p^*) a_2$	Specific mixed strategy added as a pure duplicate for Player 2.
n	Support size of mixed strategy d added as a pure duplicate.
B^0	Equivalent game after adding d as a pure duplicate.
$\tilde{B}^0$	B^0 with payoffs perturbed by up to a small ε.
U'	Equilibrium neighbourhood in $\tilde{B}^0$ corresponding to U in B.
Strategy Duplication: Pure (Section 2.4)	
a'_1, a'_2	Pure duplicates of Player 2's actions a_1 and a_2.
H_{p^*}	Levy game with $q = p^*$, with equilibrium strategy $(p^*, 1-p^*)$.
$B^{1.0}$	Equivalent game after adding a'_1 and a'_2 as pure duplicates.

Continued on next page...

Table 2.2 – Continued from previous page

Symbol	Description
U''	Equilibrium neighbourhood in $B^{1.0}$ corresponding to U in B.
$\tilde{B}^{1.0}$	$B^{1.0}$ with payoffs perturbed by up to a small ε.
$\hat{B}^{1.0}$	$\tilde{B}^{1.0}$ with payoffs adjusted by a small function Δ such that Player 2 is indifferent between his pure duplicate actions.
$\tilde{B}^{1.1}$	An augmented game where Player 2's payoffs are the sum of payoffs from $\tilde{B}^{1.0}$ and $\delta \cdot H_{p^*}$ played against an auxiliary Player 3.
δ	The weight of H_{p^*} on Player 2's payoffs.
$\tilde{B}^{1.2}$	$\tilde{B}^{1.1}$ with binary players α^1 and α^2 who adjust Player 2's payoffs by a small function Δ to achieve indifference.
Recursive Notation Convention (For Sections 2.5, 2.6)	
l	Level of recursion.
N	Number of players in the original game B.
$B^{l.0}$	Equivalent game of B: Take game B and add duplicate pure actions.
$\tilde{B}^{l.0}$	$B^{l.0}$ with payoffs perturbed by up to a small ε.
$\hat{B}^{l.0}$	$\tilde{B}^{l.0}$ with Player 2's payoffs adjusted by a small function Δ so that Player 2 is indifferent between the 2 duplicates.
$\tilde{B}^{l.1}$	An augmented game where Player 2's payoffs are the sum of payoffs from $\tilde{B}^{l.0}$ and an auxiliary game H played against an auxiliary Player.
$\tilde{B}^{l.2}$	$\tilde{B}^{l.1}$ with Player 2's payoffs adjusted by binary players α_l^1 and α_l^2 so that Player 2 is indifferent between the 2 duplicates.

2.4 Demonstrating Hyperstability Using Pure Duplicates in 2-Player Games

In this section, we take the approach from Hauk and Hurkens (2002) and modify it. Rather than adding a specific mixed strategy as duplicate to the action set of a player, we introduce pure strategy duplicates.

2.4.1 Case: Pure Duplicates in the Hauk-Hurkens Game

Let us start from the original Hauk and Hurkens (2002) game shown in Section 2.3.3, naming the game as B. We have modified this game to $B^{1.0}$ by adding L' and C' instead of adding the specific mix d as in Section 2.3.3. After a perturbation of payoffs by up to an ε, the payoff table of the equivalent game $\tilde{B}^{1.0}$ has the following changes:

$\tilde{B}^{1.0}$	L	C	R	L'	C'
T	5,4	-15,0	-11,2	5,4	-15,0
M	-23,0	-1,8	3,1	-23,0	-1,8
B	-1,2	-21,-2	1,3	-1,2	-21,-2
Out	0,0	0,0	0,0	0,ϵ	0,ϵ

So far with this modified game $\tilde{B}^{1.0}$, '*Out*' remained in the set of extreme equilibria of the game following slight perturbation to the payoffs by up to ε. Note that in this specific game with these specific payoffs, Player 2 is already indifferent between playing L' and C' against Player 1's '*Out*'. This removes the need for us to make Player 2 indifferent between the two duplicates by introducing binary players α^1 and α^2, whose role is to adjust Player 2's payoffs by a small function Δ to get this indifference. This means to say that in this case, the game $\tilde{B}^{1.0}$ and $\tilde{B}^{1.1}$ (with the binary players α^1 and α^2) are one and the same, since Δ (small adjustment to cause indifference) is 0.

The next objective is to ensure that Player 2 will play the desired mixture over his pure duplicates. Player 2's behaviour is disciplined by manipulating his payoff further - by introducing a second game H_{p^*} (in this case $p^* = 61/81$) against an auxiliary Player 3. This game is a version of the matching pennies game borrowed from Levy

(2016), where the optimal strategy for both players is to mix between their two actions with probabilities $(p^*, 1-p^*)$. Player 2 is the row player and Player 3 is the column player in H_{p^*}.

The resulting three-player game is called $\tilde{B}^{1.1}$. Here we choose any small $\delta > 0$ to control the perturbations to the payoffs from this game. This δ can be made small so as to make the perturbations to the original payoffs minimal. The payoffs for each player from the three-player game $\tilde{B}^{1.2}$ are as follows:

- Player 1's Payoff: $\tilde{B}_1^{1.2}(x, y, z) = \tilde{B}_1^{1.0}(x, y)$
- Player 3's Payoff:
  $\tilde{B}_3^{1.2}(x, y, z) = (y[L']+y[C'])\cdot\left[\delta\cdot\left[\left(H_{p^*}\right)_3\left[\left(\frac{y[L']}{y[L']+y[C']}, \frac{y[C']}{y[L']+y[C']}\right), z\right] - \left(H_{p^*}\right)_3(p^*)\right]\right]$
  Player 3's payoff is constructed to incentivise him to play the equilibrium strategy of the H_{p^*} game, forcing Player 2 to do the same.
- Player 2's total payoff $\tilde{B}_2^{1.2}(x, y, z)$ is the aggregate of his payoff from the original game with Player 1 and his payoff from the auxiliary game H_{p^*} with Player 3:
  $\tilde{B}_2^{1.2}(x, y, z) =$
  $\tilde{B}_2^{1.0}(x, y) + (y[L'] + y[C'])\cdot\left[\delta\cdot\left[\left(H_{p^*}\right)_2\left[\left(\frac{y[L']}{y[L']+y[C']}, \frac{y[C']}{y[L']+y[C']}\right), z\right] - \left(H_{p^*}\right)_2(p^*)\right]\right]$
  This equation consists of three main parts that clarify Player 2's strategic incentives:
  1. Payoff from the base game ($\tilde{B}_2^{1.0}(x, y)$): This is the payoff Player 2 receives from the perturbed equivalent of the original game against Player 1's strategy x.
  2. The weighting factor ($y[L'] + y[C']$): This is the total probability weight Player 2 assigns to his duplicate actions L' and C' in his mixed strategy y. This term ensures the disciplinary mechanism from the auxiliary game H_{p^*} is only activated if Player 2 places a positive probability on playing the duplicate actions.
  3. The disciplinary term from H_{p^*}: This is the payoff adjustment derived from Player 2's play in the auxiliary game H_{p^*} against Player 3's strategy z. The conditional mixed strategy over the duplicates $\left(\frac{y[L']}{y[L']+y[C']}, \frac{y[C']}{y[L']+y[C']}\right)$ is used as the input for Player 2's play in the Levy game. The subtraction of the equilibrium payoff, $-(H_{p^*)_2}(p^*)$, along with the small weight δ, ensures that Player 2's best response is to mix between L' and C' in the precise ratio dictated by the equilibrium of H_{p^*} (i.e., p^* and $1-p^*$), thereby replicating the equilibrium-eliminating effect of the single mixed duplicate action d.

The following claims and proofs will systematically establish the conditions under which the addition of pure duplicates and auxiliary players can replicate the equilibrium-eliminating effect of the complex mixed-strategy duplicate.

2.4.2 Main Theorem: General Case of the Hyperstability Construction

The result we demonstrate, by beginning from simpler cases and progressing onto the more general, is as follows:

Theorem 1. *Fix a game B with players I, and a non-hyperstable connected component E of equilibria on which the payoffs are constant (i.e., $B(e') = B(e'')$ for any $e', e'' \in E$). Fix $\delta > 0$. Then there exists a neighbourhood U of E and, for each $\varepsilon > 0$, a game $\tilde{B}$ with players and actions derived by:*

(i) Adding to the players in I actions which are duplicates of their pure strategies.

(ii) Adding binary players to I, making a larger set J,

with the properties that:

(a) If U'' denotes the equivalent neighbourhood of U in $\tilde{B}$ (formally, whose projection to the original players I is an equivalent neighbourhood of U), then $\tilde{B}$ has no equilibrium in U''.

(b) For any strategy profile $e = (e^I, e^J) \in U$, the payoffs of $\tilde{B}(e)$ to the players in I are δ-close to those in $B(e^I)$.

(c) Furthermore, if such an $e \in U''$ is an equilibrium of $\tilde{B}$, then for any profile ω^I for the players in I, the payoffs for players in I in $\tilde{B}(\omega^I, e^J)$ are ε-close to $B(\omega^I)$.

Hence, although we cannot choose perturbation size δ to be arbitrarily small without shrinking the neighbourhood that we eliminate equilibria in, if the additional players are playing their part in an equilibrium profile, the payoffs that the original players expect to get *for any actions they play are* ε-close to the original payoffs, where $\varepsilon > 0$ can be made arbitrarily small without revising the neighbourhood.

Proof Strategy

We start with the simplest generic game with two players, in which a singleton equilibrium is eliminated by adding a simple mixture, i.e. a new strategy that is a mixture of just two of Player 2's original actions. The construction proceeds in three conceptual stages.

Stage 1: Leveraging Non-Hyperstability. Since E is non-hyperstable, by definition there exists an equivalent game B' and a specific payoff perturbation of B' under which the component E' (equivalent to E) is no longer present in its equivalent neighbourhood U'. This construction reveals that E is vulnerable to a specific form of payoff modification: the addition of a mixed duplicate strategy (a redundant action mixing over existing pure strategies) combined with a small perturbation can eliminate E.

Stage 2: Translating Mixed Duplicates to Pure Duplicates. Rather than relying on arbitrary mixed strategies, we replace the mixed duplicate with multiple pure duplicate actions, one for each pure strategy in the support of the original mixed strategy. To ensure these pure duplicates together replicate the disciplinary effect of the original mixed duplicate, we invoke Levy's Theorem 3.2: we construct auxiliary games whose equilibrium mixing behavior is controlled to force the pure duplicates to be played in exactly the proportions specified by the original mixed duplicate.

Stage 3: Preservation of Indifference. The player whose action space is expanded with pure duplicates must remain strictly indifferent among those duplicates when the other players play their equilibrium strategy. To preserve this indifference under local deviations by the opposing player, we introduce auxiliary binary players into the game construction. These binary players enforce the required indifference condition within a local ϵ-neighborhood of the equilibrium.

Stage 4: Final perturbation. Finally, we apply the original payoff perturbation identified in Stage 1 onto the newly constructed game, so that the perturbations applied to the mixed duplicate in Stage 1 are also applied on the duplicate pure strategies. Consequently, the equivalent equilibrium component E' is eliminated from its neighborhood, demonstrating the non-hyperstability of the equilibrium.

The sections that follow contain claims and proofs that work through increasingly general cases to build the full construction and verify that the conditions of *Theorem 1* are satisfied at each stage:

- From singleton equilibria eliminated via simple mixtures (2.4.3) (which is actually the step that requires the most innovation),
- onto components of equilibria eliminated via simple mixtures (2.4.4),
- to N-player games (2.4.5),
- onto elimination via more complex mixtures (2.5), and
- to where multiple such strategies are added (2.6).

2.4.3 Case: Singleton Equilibrium, Simple Mixture Added

Game B is a generic 2-player game similar to the example above whose singleton equilibrium $e^* = (x^*, y^*)$ is payoff-robust (but not hyperstable). Let us assume that the lack of hyperstability is demonstrated by adding, say for Player 2, an additional mixed strategy that mixes over two original pure strategies, like in the *HH* example (the difference from the HH example is that, for simplicity, we assume that the equilibrium component is an isolated singleton. We remark how to generalise this to a class of non-hyperstable components like that in the *HH* example in Section 2.4.4). Let U be a small neighbourhood around e^* that contains no other equilibria except e^*. Game B, when a special mixed strategy $d = p^*a_1 + (1 - p^*)a_2$ is added to Player 2's action set as a duplicate pure strategy becomes game B^0, an equivalent game to B. B^0 with payoffs perturbed by up to ϵ becomes game $\tilde{B}^0$. Let U' be a neighbourhood of the strategy in $\tilde{B}^0$ equivalent to the equilbrium strategy e^* such that U' has no equilibria of $\tilde{B}^0$ in it. By shrinking U if needed, we may assume U' is equivalent to U.

We start again from base game B, this time adding two pure duplicate actions of a_1 and a_2 for Player 2, namely a_1' and a_2' and call this version of the game $B^{1.0}$. Let U'' be a neighbourhood of the strategy in $B^{1.0}$ equivalent to the equilibrium strategy e^* in B. By shrinking U if needed, we may assume U'' is equivalent to U. When the payoffs for Player 2's pure duplicate strategies in $B^{1.0}$ are subject to a perturbation up to a small ϵ, the resulting game is called $\tilde{B}^{1.0}$.

Claim 1. In game $B^{1.0}$, against Player 1's equilibrium strategy x^*, Player 2 is indifferent between the pure duplicate actions a_1' and a_2', i.e.,

$$B_2^{1.0}(x^*, a_1') = B_2^{1.0}(x^*, a_2')$$

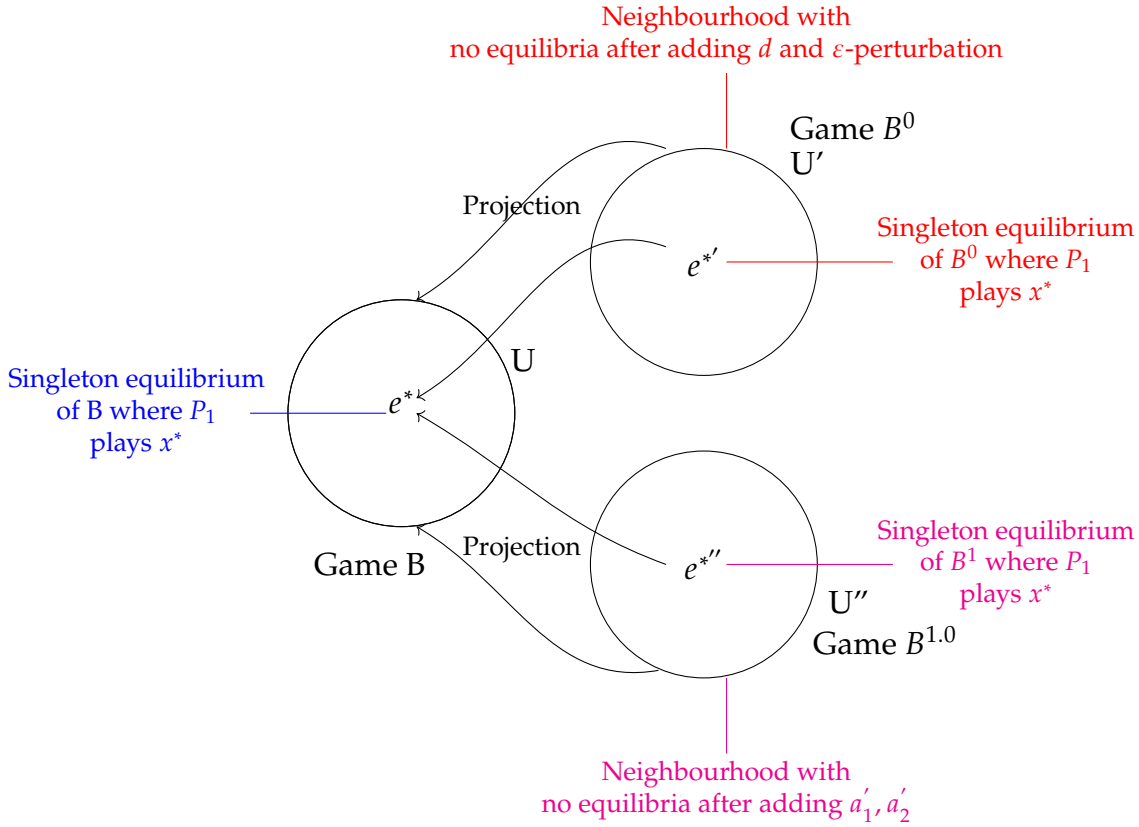


Figure 2.2 Neighbourhoods of e* in Equivalent Games

Proof. Suppose not. Without loss of generality, $B_2^{1.0}(x^*, a_1') > B_2^{1.0}(x^*, a_2')$; for $\varepsilon > 0$, let $\tilde{B}^{1.0}$ be a perturbation such that there is no equilibrium in a neighbourhood U' of e^* (reminder: without loss of generality, U'' is equivalent to a neighbourhood U in original game B). If $\varepsilon > 0$ is small enough, $\tilde{B}_2^{1.0}(x, a_1') > \tilde{B}_2^{1.0}(x, a_2')$ for any $x \in U'$ and hence in game $\tilde{B}^0$, $\tilde{B}_2^0(x, a_1) > \tilde{B}_2^0(x, d)$ (possibly by shrinking U'' further) for $x \in U''$.

Let the original game B be perturbed by ε (say, $\tilde{B}$) in the same way as $\tilde{B}^0$, except that there is no d to perturb in this case; so $\tilde{B}$ is the restriction of $\tilde{B}^0$ to original strategies. We contend that in U there is no equilibrium of $\tilde{B}$ either. This will be a contradiction, because we assume B cannot be perturbed to remove nearby equilibria.

If there were such an equilibrium $e' = (x', y') \in U$ of $\tilde{B}$, we contend that it would also be an equilibrium of $\tilde{B}^0$ in U', which is a contradiction. Indeed, since $\tilde{B}_2^0(x', a_1) > \tilde{B}_2^0(x', d)$ and a_1 is already present as one of the original strategies, we deduce that d is not used with positive probability in any equilibrium inside U; and since $y'[d] = 0$, the fact that this strategy has been added for Player 2 cannot induce a profitable

deviation for Player 1.

■

Claim 2. Let us consider again the game $\tilde{B}^{1.0}$ with pure duplicate actions for Player 2 and payoffs perturbed by a small ε. Let us modify $\tilde{B}^{1.0}$ to get $\hat{B}^{1.0}$ by changing only the payoffs to Player 2, only for the actions a_1', a_2', by $\hat{B}_2^{1.0}(\cdot, a_i') = \tilde{B}_2^{1.0}(\cdot, a_i') + \Delta_i$, for $i = 1, 2$. Then, Δ_1, Δ_2 , which depend on x, are defined by:

$$\Delta_1(x) = (1 - p^*)[\tilde{B}_2^{1.0}(x, a_2') - \tilde{B}_2^{1.0}(x, a_1')]$$

$$\Delta_2(x) = p^*[\tilde{B}_2^{1.0}(x, a_1') - \tilde{B}_2^{1.0}(x, a_2')]$$

where $x \in U''$ satisfy:

(a) $\hat{B}_2^{1.0}(x, a_1') = \hat{B}_2^{1.0}(x, a_2')$, i.e., Player 2 is indifferent between a_1', a_2'.

(b) $\hat{B}_2^{1.0}(x, p^*a_1' + (1 - p^*)a_2') = \tilde{B}_2^{1.0}(x, p^*a_1' + (1 - p^*)a_2')$, i.e. the expected payoff of $p^*a_1' + (1 - p^*)a_2'$ against x is the same in either game.

Proof. Working from condition (a): $\hat{B}_2^{1.0}(x, a_1') = \hat{B}_2^{1.0}(x, a_2')$, this is equivalent to $\tilde{B}_2^{1.0}(x, a_1') + \Delta_1(x) = \tilde{B}_2^{1.0}(x, a_1') + \Delta_2(x)$. Rearranging this equation, we get

$$\Delta_1(x) - \Delta_2(x) = \tilde{B}_2^{1.0}(x, a_2') - \tilde{B}_2^{1.0}(x, a_1') \tag{2.1}$$

Let us now look at condition (b): $\tilde{B}_2^{1.0}(x, p^*a_1' + (1 - p^*)a_2') = \hat{B}_2^{1.0}(x, p^*a_1' + (1 - p^*)a_2')$. In game $\tilde{B}^{1.0}$ when Player 2 plays $a_1'$with probability p^* and a_2' with probability $1 - p^*$ against x, then Player 2's payoff $\tilde{B}_2^{1.0}(x, p^*a_1' + (1 - p^*)a_2') = p^*[\tilde{B}_2^{1.0}(x, a_1')] + (1 - p^*)[\tilde{B}_2^{1.0}(x, a_2')]$. Similarly, $\hat{B}_2^{1.0}(x, p^*a_1' + (1-p^*)a_2') = p^*[\hat{B}_2^{1.0}(x, a_1')] + (1-p^*)[\hat{B}_2^{1.0}(x, a_2')]$. Substituting this equation in condition (b) and replacing $\hat{B}_2^{1.0}(x, a_i')$ with $\tilde{B}_2^{1.0}(x, a_i') + \Delta_i(x)$, we get

$$\tilde{B}_2^{1.0}(x, p^*a_1' + (1 - p^*)a_2') = p^*[\tilde{B}_2^{1.0}(x, a_1') + \Delta_1(x)] + (1 - p^*)[\tilde{B}_2^{1.0}(x, a_2') + \Delta_2(x)]$$

$$= p^*[\tilde{B}_2^{1.0}(x, a_1')] + (1 - p^*)[\tilde{B}_2^{1.0}(x, a_2')] + p^*(\Delta_1(x)) + (1 - p^*)(\Delta_2(x))$$

$$= \tilde{B}_2^{1.0}(x, p^*a_1' + (1 - p^*)a_2') + p^*(\Delta_1(x)) + (1 - p^*)(\Delta_2(x))$$

Rearranging, we get

$$p^*(\Delta_1(x)) + (1 - p^*)(\Delta_2(x)) = 0 \tag{2.2}$$

Substituting equation (2.2) in (2.1) and solving for $\Delta_1(x)$ and $\Delta_2(x)$:

$$\Delta_2(x) = p^*[\tilde{B}_2^{1.0}(x, a_1') - \tilde{B}_2^{1.0}(x, a_2')]$$

and

$$\Delta_1(x) = (1 - p^*)[\tilde{B}_2^{1.0}(x, a_2') - \tilde{B}_2^{1.0}(x, a_1')]$$

Since, from Claim 1, $B_2^{1.0}(x^*, a_1') = B_2^{1.0}(x^*, a_2')$ and ε in $\tilde{B}_2^{1.0}$ is small (perturbing the payoffs in $B_2^{1.0}$ by at most ε), we can say that

$$\tilde{B}_2^{1.0}(x^*, a_1') - B_2^{1.0}(x^*, a_1') = \varepsilon_1$$

$$\tilde{B}_2^{1.0}(x^*, a_2') - B_2^{1.0}(x^*, a_2') = \varepsilon_2$$

where $|\varepsilon_1|, |\varepsilon_2| \leq \varepsilon$. Using Claim 1,

$$\tilde{B}_2^{1.0}(x^*, a_2') - \tilde{B}_2^{1.0}(x^*, a_1') = \varepsilon_2 - \varepsilon_1 = O(\varepsilon)$$

where $|\varepsilon_2 - \varepsilon_1| \leq 2\varepsilon$. As $x \in \varepsilon$-neighbourhood of x^*, then

$$\tilde{B}_2^{1.0}(x, a_2') - \tilde{B}_2^{1.0}(x, a_1') = O(\varepsilon) + O(\|x - x^*\|)$$

and therefore $\Delta_1(x)$ and $\Delta_2(x)$ are small:

$$\Delta_1(x) = (1 - p^*)(O(\varepsilon)) + O(\|x - x^*\|) \tag{2.3}$$

$$\Delta_2(x) = -p^* O(\varepsilon) + O(\|x - x^*\|) \tag{2.4}$$

If U'' is chosen small enough, then Δ_1, Δ_2 are smaller than δ in U''. This follows from the equalities (2.3) and (2.4). This will give the conclusion (b).

■

In our context, the pseudo-game $\hat{B}^{1.0}$ is used as an intermediate step where Player 2's payoffs are adjusted by such a small continuous function Δ to induce indifference between duplicate actions. [2]

Claim 3. Consider the pseudo-game $\hat{B}^{1.0}$. Let us add a Player 3 to the mix, who plays the game $\delta \cdot H_{p^*}$ (Levy's matching-pennies game with the parameter $q = p^*$) against

[2]The game $\hat{B}^{1.0}$ is referred to as a pseudo-game because the adjustments (Δ function) to Player 2's payoffs are made to ensure a specific condition (indifference) holds, which is then formally simulated by introducing auxiliary players and the Levy game H_η in the construction of the final game $\tilde{B}^{1.2}$ (see Claim 5 and Corollary 1). This approach essentially makes the game's payoff structure non-standard until the final transformation is complete.

Player 2. Let this game be called $\tilde{B}^{1.1}$. If $(\bar{x}, \bar{y}, \bar{z})$ is an equilibrium of $\tilde{B}^{1.1}$ where $(\bar{x}, \bar{y}) \in U''$, then either

$$\bar{y}[a'_1] = \bar{y}[a'_2] = 0$$

or

$$\frac{\bar{y}[a'_1]}{\bar{y}[a'_2]} = \frac{p^*}{1-p^*}$$

Proof. If Player 2 plays the pair $\{a'_1, a'_2\}$ with positive probability, but not in the ratio $\frac{p^*}{1-p^*}$, then Player 3 plays pure, say L. By Claim 2, since the payoff from Player 2 is the same for a'_1, a'_2 under $\hat{B}^{1.0}$ against $\bar{x}$ and since $(\bar{x}, \bar{y}) \in U''$, we have $\tilde{B}^{1.1}(x, a'_1, z) < \tilde{B}^{1.1}(x, a'_2, z)$, where z is the action of Player 3. But if then Player 2 puts positive weight on a'_1 and none on a'_2, Player 3 should deviate to R, which would yield $\tilde{B}^{1.1}(x, a'_1, z) > \tilde{B}^{1.1}(x, a'_2, z)$, causing Player 3 to deviate to L, contradiction. ■

Claim 4. In the quasi-game $\tilde{B}^{1.1}$, no equilibria exist in which Player 1 and Player 2 play in the neighbourhood U''.

Proof. By Claim 3, if the game $\tilde{B}^{1.1}$ has an equilibrium (x^*, y^*, z^*) where $x^* \in U''$, then y^* plays actions a'_1, a'_2 with the correct ratio $p^*, 1-p^*$, if they are played at all - any such equilibrium would induce an equilibrium (x^*, y^{**}) in game $\tilde{B}^0$, where $y^{**}[d] = y^*[a'_1] + y^*[a'_2]$ and $y^{**}[b] = y^*[b]$ for each original action b. We would then have (x^*, y^{**}) is in U' of game $\tilde{B}^0$. Since there is no equilibrium in U', either Player 1 has a profitable deviation x' or Player 2 has a profitable deviation y'. In the first case, since y^* plays a'_1, a'_2 with the correct ratio $p^*, 1-p^*$, and payoffs under profiles including a'_1, a'_2 are perturbed in the same manner as d, we see that $\tilde{B}^{1.1}_1(x', y^*) = \tilde{B}^0_1(x', y^{**}) > \tilde{B}^0_1(x^*, y^{**}) = \tilde{B}^{1.1}_1(x^*, y^*)$. In the second case, define the strategy for $\tilde{B}^{1.1}$ of Player 2 where $y''[b] = y'[b]$ for each original action b, $y''[a'_1] = p^* y'[d]$, $y''[a'_2] = (1-p^*)y'[d]$, so then $\tilde{B}^{1.1}_2(x^*, y'') = \tilde{B}^0_2(x^*, y') > \tilde{B}^0_2(x^*, y^*) = \tilde{B}^{1.1}_2(x^*, y^*)$. ■

Claim 5. Let $\tilde{B}^{1.1}$ be an ε-perturbation of $B^{1.0}$ in which Player 2 plays the game $\delta \cdot H_{p^*}$ against Player 3. Let strategy x of Player 1 be in U, a small neighbourhood around x^*. Denote the function η as

$$\eta(x) = 0.5 + \Delta_1(x)$$

There exists a *Levy* game H_η (recall from Section 2.3.4) in which two new binary players $\{\alpha^1, \alpha^2\}$ play against Player 1 such that the game simulates the function $\eta(x)$. The payoffs depend upon x (Player 1's strategy in $\tilde{B}^{1.0}$), and in any equilibrium θ of

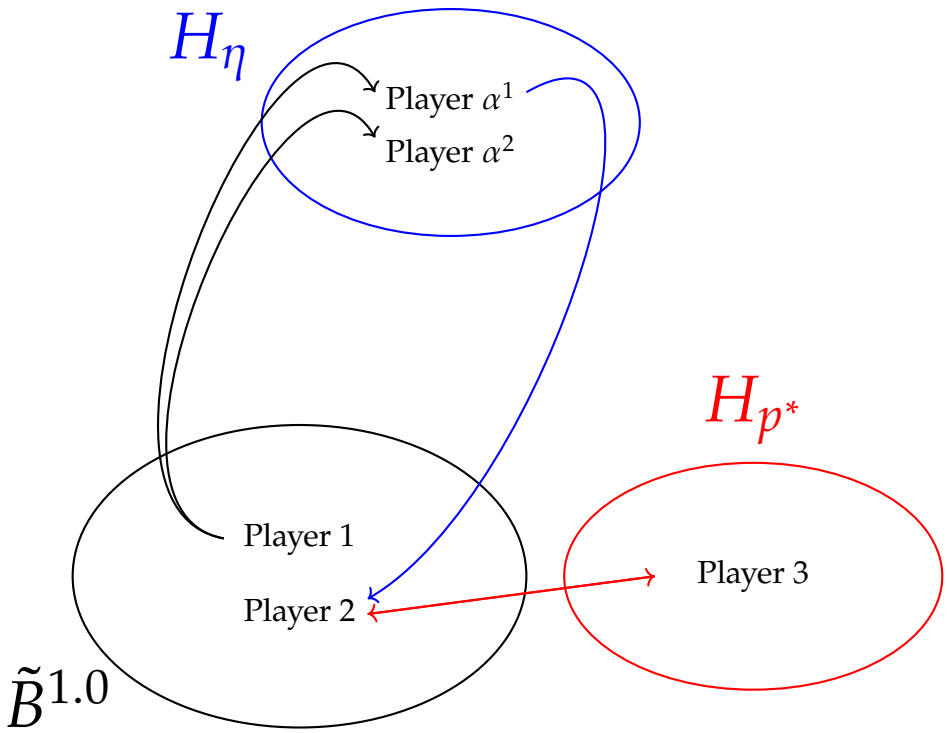


Figure 2.3 Games and Players in $\hat{B}^{1.1}$. The arrows point to the players' effects on each other.

H_η, we have $\theta^{\alpha^1} = \eta(x)$.

Proof. Using Theorem 3.2 from Levy (2016) (see section 3.3 for the Theorem) in our context, the function η_i can be realised as a projection of the equilibrium correspondence of the game H_η in which payoffs depend on the function's domain, i.e., the strategy x of Player 1. If θ is an equilibrium of game H_η, then the equilibrium strategy of the binary player α^1, uniquely determined by x, will be: $\theta^{\alpha^1} = \eta(x)$. In other words, in equilibrium θ of game H_η, α^1 plays $\eta(x)$.

■

Corollary 1. *Let $\tilde{B}^{1.1}$ be an ε-perturbation of $B^{1.0}$ in which Player 2 plays the game $\delta \cdot H_{p^*}$ against Player 3. Let strategy x of Player 1 be in U, a small neighbourhood around x^*. Let us modify the payoffs of Player 2 using the outcome from game H_η to obtain game $\tilde{B}^{1.2}$ where there are 3 players and 2 binary players such that:*

$$\tilde{B}_2^{1.2}(x, a_1', z, u, v) = \tilde{B}^{1.1}(x, a_1', z) + u - 0.5$$

and

$$\tilde{B}_2^{1.2}(x, a_2', z, u, v) = \tilde{B}^{1.2}(x, a_2', z) + \left(\frac{-p^*}{1-p^*}(u - 0.5)\right)$$

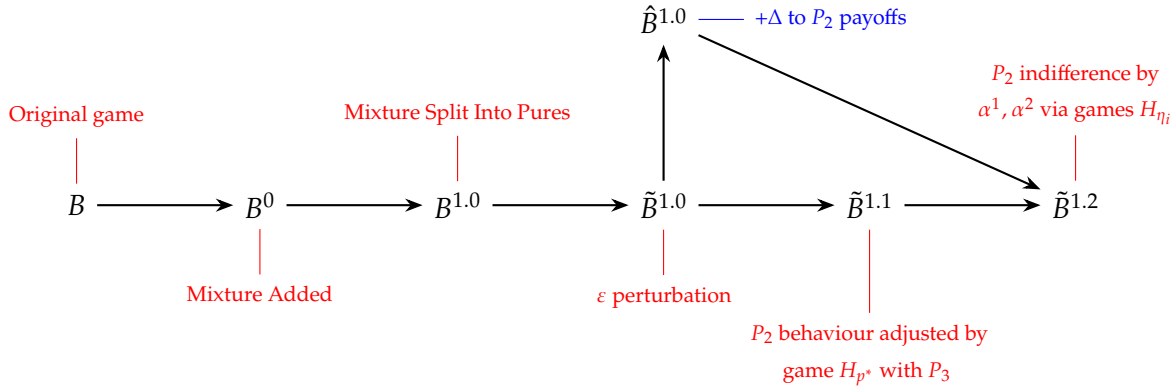


Figure 2.4 Construction of game $\tilde{B}^{1.2}$

where u is the equilibrium strategy of the binary player α^1 in game H_η. In any equilibrium profile of the game $\tilde{B}^{1.2}$, Player 2 is indifferent between the two pure duplicate actions a'_1 and a'_2.

Proof. *For the game $\tilde{B}^{1.2}$, in any equilibrium profile $(x^*, y^*, z^*, u^*, v^*)$, it holds that $\tilde{B}_2^{1.2}(x^*, a'_1, z^*, u^*, v^*) = \tilde{B}_2^{1.2}(x^*, a'_2, z^*, u^*, v^*)$, due to Claim 2 and Claim 3.* ■

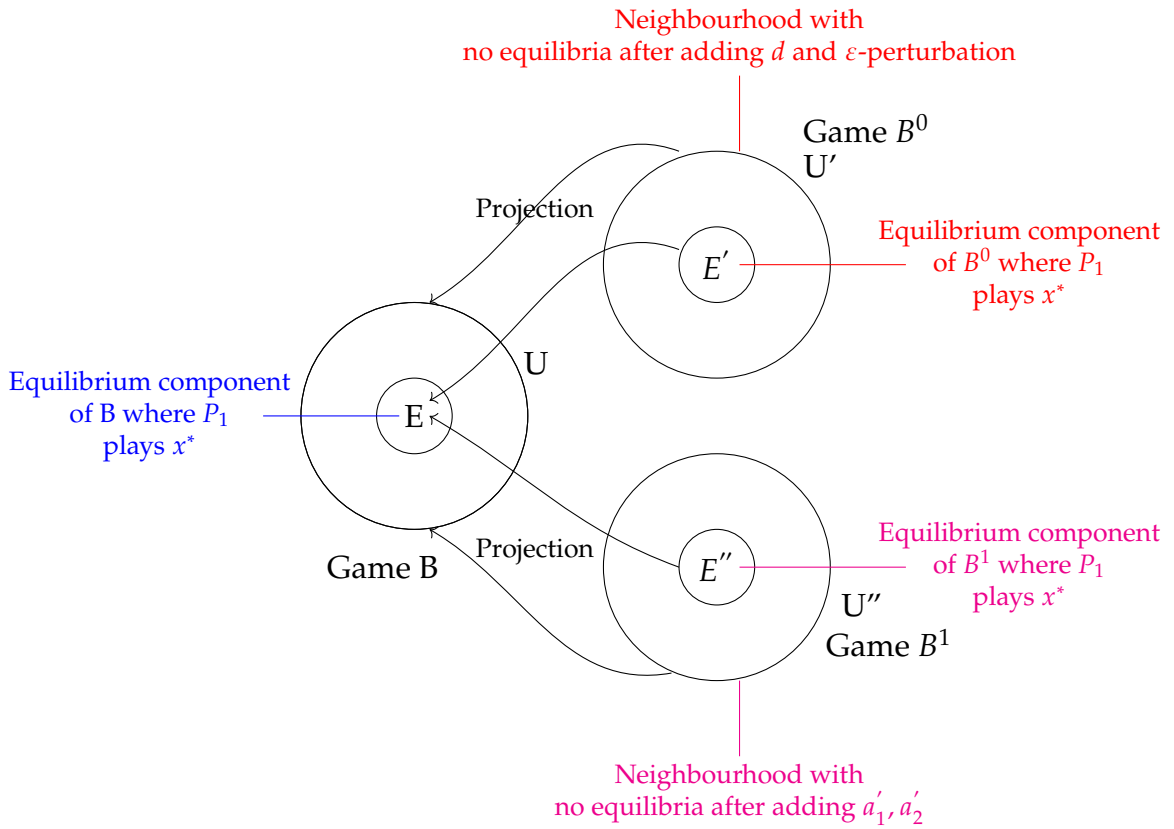


Figure 2.5 Neighbourhoods of E in Equivalent Games

2.4.4 Case: 2-Player Games and Equilibrium Component Eliminated by a Simple Mixture of 2 Actions

In games like the HH game, adding $d = p^* a_1 + (1 - p^*)a_2$ as a duplicate pure action eliminated not just a singleton equilibrium, but a connected component of equilibria that awards the same payoffs to both the players.

Claim 6. A connected component E^* of game B can be eliminated by adding pure duplicates and an additional player, parallel to *Corollary 2* following Claim 4, as long as the payoffs of all the players remain the same in all the equilibria within this connected component E^*:
$\exists \rho \in \mathbb{R}^2$ s.t. $\forall e \in E^*$, $B(e) = \rho$, i.e., the payoffs to both players are unchanged in all e within the connected component E^*. This implies that $\forall e \in E^*$, $\forall i \in 1, 2$ (players), $\forall a_i \in A_i$ (actions), if $a_i \in supp(e_i) \Rightarrow B_i(a_i, e_{-i}) = \rho$.
Proof. The proof is identical to that of the previous case, as shown in figure 2.5. ■

2.4.5 Case: Player 1 Is a Collection of N players

When we state that Player 1 is replaced by a collection of N players, we are expanding the game from a two-player setting to an $(N + 1)$-player setting, where the original Player 2 now acts as Player $N + 1$.

Throughout this analysis so far, we have not used the fact that Player 1 is a single player. We could perform the same manipulations to Player $N + 1$ that we have done to Player 2 in Section 2.4.3. Therefore *Claims 1-6* do apply to cases where Player 1 is a collection of N players. In this case the difference is that the equilibrium strategy x^* in e^* from the previous section would be an equilibrium strategy profile x^* of N players, while the rest of the logic follows in the same way as in the 2-player version.

2.5 Inductive Extension: N-Player Games and an Equilibrium Component Is Eliminated Using a Complex Mixture

Game C is a N-player game, with an equilibrium component E^* that is payoff-robust and the equilibrium payoffs of all N players remain the same in each $e^* \in E^*$. We assume the equilibrium component E^* can be eliminated by adding a specific strategy d to Player N's set of actions and applying a arbitrarily small ϵ payoff perturbation.

Game C^0 is the game C after adding the mixed duplicate action $d = p_1^* a_1 + p_2^* a_2 + ... + p_n^* a_n$ for Player N where $\sum p_i^* = 1$, and where the coefficients are defined as:

$$p_1^* = \beta_1 \beta_2 ... \beta_{n-1}$$

$$p_2^* = (1 - \beta_1)\beta_2 ... \beta_{n-1}$$

$$p_3^* = (1 - \beta_2)\beta_3 ... \beta_{n-1}$$

$$\vdots$$

$$p_{n-1}^* = (1 - \beta_{n-2})\beta_{n-1}$$

$$p_n^* = 1 - \beta_{n-1}$$

Let $\tilde{C}^0$ be the game C^0 with payoffs perturbed by an arbitrarily small amount ϵ to eliminate the component E^*. Let U' be the neighbourhood in $\tilde{C}^0$ where no equilibria persist under this perturbation, and let U be the equivalent component in the original game C. We assume that the mixed strategy d which removes all equilibria in U' is such that no simpler mixture (i.e., a mixture with smaller support) will do the job of eliminating the component.

Now, we approach the problem of eliminating E^* by adding duplicate pure actions for Player N, by recursively unpacking d. This unpacking is achieved by replacing one mixed duplicate with a pure duplicate and a simplified mixed duplicate action at each step:

1. At each level l, the modified game is labeled $C^{l.0}$. The perturbed version is $\tilde{C}^{l.0}$.

2. At the final level ($l = n - 1$), the mixed duplicate is fully unpacked into two pure duplicate actions.

3. At each level of recursion l, we add *three* new players: an auxiliary player k (where $k = N + l$) and two binary players α_l^1 and α_l^2.

At the first level ($l = 1$), we begin from game C, adding the pure duplicate a'_n (a copy of a_n) and the simplified mixed duplicate:

$$d_1 = (\beta_1\beta_2 \dots \beta_{n-2})a_1 + ((1-\beta_1)\beta_2 \dots \beta_{n-2})a_2 + \dots + (1-\beta_{n-2})a_{n-1}$$

This creates game $C^{1.0}$. This initial split into one pure and one mixed duplicate follows the same logic as the construction of $B^{1.0}$ in Section 2.4.3. Let U'' be the neighbourhood in $C^{1.0}$ equivalent to U in C. In the perturbed version $\tilde{C}^{1.0}$, Player N's payoffs for these duplicates are perturbed by ϵ. The pseudo-game $\hat{C}^{1.0}$ is then derived, where Player N is made indifferent between a'_n and d_1 by balancing his payoffs using a small function Δ_1 (a function of the other players' strategies $x \in U''$). The derivation of this indifference and the properties of Δ_1 are analogous to Claim 2. The small function Δ_1 is simulated by the binary players α_1^1 and α_1^2 playing the Levy game H_{η_1}. Next, in game $\tilde{C}^{1.1}$, the auxiliary Player $k = N + 1$ is introduced to play the game $\delta \cdot (H_{\beta_{n-1}})$ against Player N to discipline the mixing ratio between a'_n and d_1. As in Claim 3, the equilibrium properties of this auxiliary game force Player N into the specific mixing ratio required by β_{n-1}. In the final augmented game $\tilde{C}^{1.2}$, Player N's payoffs are the sum of his payoff from the base game $\tilde{C}^{1.1}$ and his discounted payoffs from the Levy game $H_{\beta_{n-1}}$ (via the auxiliary Player k) and the H_{η_1} game (via the binary players α_1^1 and α_1^2) as we did in Claim 5.

Generalising for any level of recursion l, we start at game $C^{l.0}$. We remove the duplicate action from the mixed strategy d_{l-1}, and instead add a pure duplicate action a'_j and a new mixed duplicate action d_l, where $j = n + 1 - l$ and

$$d_l = (\beta_1\beta_2 \dots \beta_{j-2})a_1 + ((1-\beta_1)\beta_2 \dots \beta_{j-2})a_2 + \dots + (1-\beta_{j-2})a_{j-1}$$

The game $\tilde{C}^{l.0}$ is derived from $C^{l.0}$ by perturbing payoffs for Player N by ϵ when playing a'_j and d_l. The game $\hat{C}^{l.0}$ is derived by adding the payoff function Δ_l (simulated by α_l^1 and α_l^2 playing H_{η_l}) to ensure Player N is indifferent between d_l and a'_j. After this, the game $\tilde{C}^{l.1}$ is created by adding the auxiliary Player k (where $k = N + l$) who plays the game $\delta \cdot (H_{\beta_{j-1}})$ against Player N. In the final augmented game $\tilde{C}^{l.2}$, Player N's payoffs are the sum of his payoff from the base game $\tilde{C}^{l.1}$ and the adjustments

determined by the Levy game $H_{\beta_{j-1}}$ (via the auxiliary Player) and the H_{η_l} game (via the binary players α_l^1 and α_l^2).

The unperturbed game $C^{l.0}$ is the equivalent game of $C^{(l-1).0}$ where the mixed duplicate d_{l-1} is further unpacked into a pure duplicate a'_j and a new mixed duplicate d_l. The payoffs in $C^{l.0}$ for the newly introduced actions are inherited from the indifferent payoffs in the adjusted game of the previous level, $\tilde{C}^{(l-1).2}$. By induction, since $C^{1.0}$ already contains the initial mixture (split as d_1 and a'_n) of the complex strategy d, each subsequent $C^{l.0}$ continues this process of substituting one mixed strategy with a pure strategy and a simpler mixture until $l = n-1$, where only pure duplicates remain.

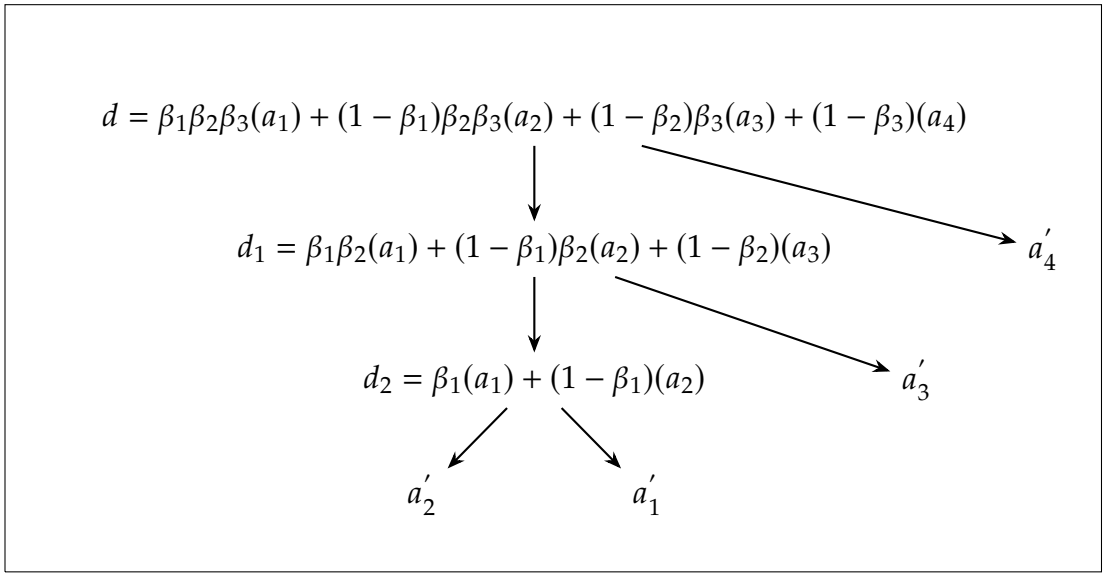


Figure 2.6 An example

Claim 7. Let C be a game with a payoff-robust equilibrium component where each $e^* = (x^*, y^*)$. At each recursive stage l, against profile x^* of Player 1 to Player $N-1$, Player N is indifferent between the two new duplicate actions introduced in level l:

$$C_N^{l.0}(x^*, d_l) = C_N^{l.0}(x^*, a'_j)$$

where $j = n + 1 - l$.

Proof. Suppose not. Assume the two payoffs are not equal, meaning one action is strictly better for Player N against the opponents' equilibrium strategy x^* in the unperturbed game $C^{l.0}$. The action a'_j is a pure duplicate of an original action a_j. The action d_l is a mixture of original actions $a_1, \ldots, a_{j-1}$ (by construction, d_l is equivalent to a mixture of $j-1$ of Player N's original actions). The equilibrium $e^* = (x^*, y^*)$ is a Nash equilibrium of the original game C. We consider two cases:

- **Case 1: The pure duplicate is strictly better.**
  Let $C_N^{l.0}(x^*, a_j') > C_N^{l.0}(x^*, d_l)$; for $\varepsilon > 0$, let $\tilde{C}^{l.0}$ be a perturbation such that there is no equilibrium in a neighbourhood U' of e^* (reminder: without loss of generality, U' is equivalent to a neighbourhood U in original game C). If $\varepsilon > 0$ is small enough, $\tilde{C}_N^{l.0}(x, a_j') > \tilde{C}_N^{l.0}(x, d_l)$ for any $x \in U'$ (possibly by shrinking U') and hence in the game $\tilde{C}^0$, $\tilde{C}_N^0(x, a_j) > \tilde{C}_N^0(x, d)$ (possibly by shrinking U'' further) for $x \in U'$.

  Let the original game C be perturbed by ε (say, $\tilde{C}$) in the same way as $\tilde{C}^0$, except that there is no d to perturb in this case; so $\tilde{C}$ is the restriction of $\tilde{C}^0$ to original strategies. We contend that in U there is no equilibrium of $\tilde{C}$ either. This will be a contradiction, because we assume C cannot be perturbed to remove nearby equilibria.

  If there were such an equilibrium $e' = (x', y') \in U$ of $\tilde{C}$, we contend that it would also be an equilibrium of $\tilde{C}^0$ in U', which is a contradiction. Indeed, since $\tilde{C}_N^0(x', a_j) > \tilde{C}_N^0(x', d)$ and a_j is already present as one of the original strategies, we deduce that d is not used with positive probability in any equilibrium inside U; and since $y'[d] = 0$, the fact that this strategy has been added for Player N cannot induce a profitable deviation for the other $N - 1$ players.

- **Case 2: The Mixed Duplicate is Strictly Better.**
  Assume $C_N^{l.0}(x^*, d_l) > C_N^{l.0}(x^*, a_j')$. Since d_l is a convex combination of original pure strategies $a_1, \ldots, a_{j-1}$, and a_j' is a pure duplicate of a_j, the strict inequality means that playing the mixture d_l yields a payoff strictly greater than playing a_j. For $\epsilon > 0$, let $\tilde{C}^{l.0}$ be a perturbation such that there is no equilibrium in the neighbourhood U' of e^*. If ϵ is small enough, the strict inequality persists for any $x \in U'$: $\tilde{C}_N^{l.0}(x, d_l) > \tilde{C}_N^{l.0}(x, a_j')$. This means that in the equivalent game $\tilde{C}^0$, the mixed strategy d is strictly preferred to a_j against $x \in U'$: $\tilde{C}_N^0(x, d) > \tilde{C}_N^0(x, a_j)$, for $x \in U'$.

  If there were an equilibrium $e' = (x', y') \in U$ of the perturbed original game $\tilde{C}$ (the restriction of $\tilde{C}^0$ to original strategies), then y' must be a best response for Player N against x'. Since d is a mixture of strategies in the original support, the strict inequality $\tilde{C}_N^0(x', d) > \tilde{C}_N^0(x', a_j)$ implies that Player N's mixed strategy y' would not assign positive weight to a_j. If d_l (a mixture with smaller support) is strictly better than a_j and therefore d, it would imply d_l could have eliminated the equilibrium in U' in game $\tilde{C}^0$ on its own. This violates our assumption of no simpler mixture than d could do the job of eliminating all equilibrium near $e^* \in U'$.

More importantly, the existence of the strictly better mixture d_l means that Player N has a profitable deviation from y' if a_j is in the support, or that the support itself is not optimal otherwise which contradicts e' being an equilibrium of $\tilde{C}$ in U. Since d_l is made up of original strategies, this contradiction implies that in U there is no equilibrium of $\tilde{C}$ either, which violates the assumption that the component E^* is payoff-robust, i.e., C cannot be perturbed to remove the nearby equilibrium component.

In either case, assuming that $C_N^{l.0}(x^*, d_l) \neq C_N^{l.0}(x^*, a'_j)$ leads to a contradiction of the initial premise that $e^* = (x^*, y^*)$ is a Nash equilibrium of the base game C, which can be eliminated throughout a neighbourhood U' of an equivalent game derived by adding a pure action which is a mixture of actions in the original game, but not by adding any simpler mixture. Therefore, Player N must be indifferent between the two duplicate actions:

$$C_N^{l.0}(x^*, d_l) = C_N^{l.0}(x^*, a'_j)$$

If the mixed duplicate d_l is itself a pure duplicate action (which occurs at the final level of recursion, $l = n - 1$), then the structure of the proof simplifies and is analogous to the argument used in Claim 1, as both a'_j and d_l are copies of original pure strategies.

■

Claim 8. Let us consider again the game $\tilde{C}^{l.0}$ with pure duplicate actions for Player N and payoffs perturbed by a small ε. Let us modify $\tilde{C}^{l.0}$ to get $\hat{C}^{1.0}$ by changing only the payoffs to Player N, only for the actions d_l, a'_j, by $\hat{C}_N^{l.0}(\cdot, d_l) = \tilde{C}_N^{l.0}(\cdot, d_j) + \Delta_{d_l}$ and $\hat{C}_N^{l.0}(\cdot, a'_j) = \tilde{C}_N^{l.0}(\cdot, a'_j) + \Delta_{a_j}$. Then, $\Delta_{d_l}, \Delta_{a_j}$, which depend on x, are defined by:

For $l = 1$:

$$\Delta_{d_1}(x) = (1 - \beta_{n-1})[\tilde{C}_N^{1.0}(x, a'_n) - \tilde{C}_N^{1.0}(x, d_1)]$$
$$\Delta_{a'_n}(x) = \beta_{n-1}[\tilde{C}_N^{1.0}(x, d_1) - \tilde{C}_N^{1.0}(x, a'_n)]$$

For $l > 1$:

$$\Delta_{d_l}(x) = \Delta_{d_{l-1}}(x) + (1 - \beta_{j-1})[\tilde{C}_N^{l.0}(x, a'_j) - \tilde{C}_N^{l.0}(x, d_l)]$$
$$\Delta_{a'_j}(x) = \Delta_{d_{l-1}}(x) - \beta_{j-1}[\tilde{C}_N^{l.0}(x, a'_j) - \tilde{C}_N^{l.0}(x, d_l)]$$

and satisfy the properties:

(a) $\hat{C}_N^{l.0}(x, d_l) = \hat{C}_N^{l.0}(x, a'_j)$, i.e., Player N is indifferent between d_l and a'_j.

(b) $\hat{C}_N^{l.0}(x, \beta_{j-1}d_l + (1-\beta_{j-1})a'_j) = \tilde{C}_N^{l.0}(x, \beta_{j-1}d_l + (1-\beta_{j-1})a'_j)$, i.e. the expected payoff of $\beta_{j-1}d_l + (1-\beta_{j-1})a'_j$ against x is the same in either game.

(c) For $l > 1$, $\hat{C}_N^{l.0}(x, \beta_{j-1}d_l + (1-\beta_{j-1})a'_j) = \hat{C}_N^{(l-1).0}(x, d_{l-1}) = \tilde{C}_N^{(l-1).0}(x, d_{l-1}) + \Delta_{d_{l-1}}$, i.e. the expected payoff before and after unpacking a mixed duplicate is the same against x.

Proof. The proof is established by demonstrating that the required forms of $\Delta_{d_l}(x)$ and $\Delta_{a'_j}(x)$ satisfy properties (a) and (b), and that property (c) holds by construction.

We require $\hat{C}_N^{l.0}(x, d_l) = \hat{C}_N^{l.0}(x, a'_j)$, which means:

$$\tilde{C}_N^{l.0}(x, d_l) + \Delta_{d_l}(x) = \tilde{C}_N^{l.0}(x, a'_j) + \Delta_{a'_j}(x)$$

Rearranging the indifference condition yields:

$$\Delta_{d_l}(x) - \Delta_{a'_j}(x) = \tilde{C}_N^{l.0}(x, a'_j) - \tilde{C}_N^{l.0}(x, d_l)$$

Subtracting the formulas given for $l > 1$:

$$\Delta_{d_l}(x) - \Delta_{a'_j}(x) = \left(\Delta_{d_{l-1}}(x) + (1-\beta_{j-1})[\tilde{C}_N^{l.0}(x, a'_j) - \tilde{C}_N^{l.0}(x, d_l)]\right)$$

$$- \left(\Delta_{d_{l-1}}(x) - \beta_{j-1}[\tilde{C}_N^{l.0}(x, a'_j) - \tilde{C}_N^{l.0}(x, d_l)]\right)$$

$$\Delta_{d_l}(x) - \Delta_{a'_j}(x) = (1 - \beta_{j-1} + \beta_{j-1}) \cdot [\tilde{C}_N^{l.0}(x, a'_j) - \tilde{C}_N^{l.0}(x, d_l)]$$

$$\Delta_{d_l}(x) - \Delta_{a'_j}(x) = \tilde{C}_N^{l.0}(x, a'_j) - \tilde{C}_N^{l.0}(x, d_l)$$

This confirms that the formulas satisfy the indifference condition.

Condition (c) requires that the expected payoff of the mixture $\beta_{j-1}d_l + (1-\beta_{j-1})a'_j$ in the current adjusted game ($\hat{C}^{l.0}$) must be equal to the adjusted payoff of d_{l-1} from the previous level ($\hat{C}^{(l-1).0}$):

$$\hat{C}_N^{l.0}(x, \beta_{j-1}d_l + (1-\beta_{j-1})a'_j) = \hat{C}_N^{(l-1).0}(x, d_{l-1})$$

The Left-Hand Side (LHS) term (current expected adjusted payoff) can be expanded using the definition of $\hat{C}^{l.0}$ ($\hat{C}_N^{l.0}(\cdot) = \tilde{C}_N^{l.0}(\cdot) + \Delta$):

$$\hat{C}_N^{l.0}(x, \beta_{j-1} d_l + (1-\beta_{j-1}) a'_j) = \beta_{j-1} \hat{C}_N^{l.0}(x, d_l) + (1-\beta_{j-1}) \hat{C}_N^{l.0}(x, a'_j)$$

$$= \beta_{j-1} \left(\tilde{C}_N^{l.0}(x, d_l) + \Delta_{d_l} \right) + (1-\beta_{j-1}) \left(\tilde{C}_N^{l.0}(x, a'_j) + \Delta_{a'_j} \right)$$

$$= \tilde{C}_N^{l.0}(x, \beta_{j-1} d_l + (1-\beta_{j-1}) a'_j) + \beta_{j-1} \Delta_{d_l} + (1-\beta_{j-1}) \Delta_{a'_j}$$

The Right-Hand Side (RHS) term (previous adjusted payoff) can be expanded using the definitions $\hat{C}^{(l-1).0}(\cdot) = \tilde{C}^{(l-1).0}(\cdot) + \Delta_{d_{l-1}}$. Since $\tilde{C}_N^{(l-1).0}(x, d_{l-1})$ is equivalent to the unadjusted payoff of the mixture $\tilde{C}_N^{l.0}(x, \beta_{j-1} d_l + (1-\beta_{j-1}) a'_j)$:

$$\hat{C}_N^{(l-1).0}(x, d_{l-1}) = \tilde{C}_N^{(l-1).0}(x, d_{l-1}) + \Delta_{d_{l-1}}$$

$$= \tilde{C}_N^{l.0}(x, \beta_{j-1} d_l + (1-\beta_{j-1}) a'_j) + \Delta_{d_{l-1}}$$

Equating the LHS and RHS and canceling the identical $\tilde{C}$ expected payoff terms:

$$\tilde{C}_N^{l.0}(x, \beta_{j-1} d_l + (1-\beta_{j-1}) a'_j) + \beta_{j-1} \Delta_{d_l} + (1-\beta_{j-1}) \Delta_{a'_j} = \tilde{C}_N^{l.0}(x, \beta_{j-1} d_l + (1-\beta_{j-1}) a'_j) + \Delta_{d_{l-1}}$$

This isolates the Δ terms:

$$\beta_{j-1} \Delta_{d_l} + (1-\beta_{j-1}) \Delta_{a'_j} = \Delta_{d_{l-1}} \tag{2.5}$$

From the system formed by Equation (1) and Equation (2), we solve for Δ_{d_l} and $\Delta_{a'_j}$:

$$\Delta_{d_l}(x) = \Delta_{d_{l-1}}(x) + (1-\beta_{j-1})[\tilde{C}_N^{l.0}(x, a'_j) - \tilde{C}_N^{l.0}(x, d_l)]$$

$$\Delta_{a'_j}(x) = \Delta_{d_{l-1}}(x) - \beta_{j-1}[\tilde{C}_N^{l.0}(x, a'_j) - \tilde{C}_N^{l.0}(x, d_l)]$$

The term $[\tilde{C}_N^{l.0}(x, a'_j) - \tilde{C}_N^{l.0}(x, d_l)]$ is $O(\epsilon) + O(\|x - x^*\|)$ because the unperturbed game $C^{l.0}$ satisfies $C_N^{l.0}(x^*, d_l) = C_N^{l.0}(x^*, a'_j)$ (from Claim 7) and the perturbations on the payoffs from a'_j and d_l are bounded by ε. This yields:

$$\Delta_{d_l}(x) = \Delta_{d_{l-1}}(x) + (1-\beta_{j-1}) \left(O(\epsilon) \right) + O(\|x - x^*\|)$$

$$\Delta_{a'_j}(x) = \Delta_{d_{l-1}}(x) - \beta_{j-1} \left(O(\epsilon) \right) + O(\|x - x^*\|)$$

■

Claim 9. Suppose x is a strategy profile of Player 1 to N-1 close to x^* in $\hat{C}^{l.0}$. If a game $\tilde{C}^{l.1}$ is constructed from here by adding a Player k to play a game of $\delta \cdot (H_{\beta_{j-1}})$ for some $\delta > 0$ against Player N where $k = N + l$ and where $j = n + 1 - l$. If there is an equilibrium of $\tilde{C}^{l.1} = (\hat{x}^l, \hat{y}^l, \hat{z}_1^l ..., \hat{z}_k^l)$ where $\hat{x}^l = x$, then either

$$\hat{y}^l[d_l] = \hat{y}^l[a'_j] = 0$$

or

$$(\frac{\hat{y}^l[d_l]}{\hat{y}^l[a'_j]} = \frac{\beta_{j-1}}{1 - \beta_{j-1}}$$

and

$$\frac{\hat{y}^l[d_l]}{\hat{y}^l[a'_j] + \hat{y}^l[a'_{j+1}] + ... + \hat{y}^l[a'_n]} = \frac{\prod_{\gamma=j-1}^{n-1} \beta_\gamma}{1 - \prod_{\gamma=j-1}^{n-1} \beta_\gamma})$$

Note that if $l = n - 1$ and $j = 2$, then we are at the last level of recursion and $d_{n-1} = a'_1$, a pure duplicate.

Proof. The proof is identical to that of Claim 3.

■

Claim 10. In game $\tilde{C}^{1.1}$, no equilibria exist in which Player 1 to Player $N - 1$ and Player 2 play in the neighbourhood U''.

Proof. The proof is identical to that of Claim 4.

■

Claim 11. Let $\tilde{C}^{l.1}$ be an ε-perturbation of $C^{l.0}$ and in which Player N plays the game $\delta \cdot (H_{\beta_{j-1}})$ against Player k and let Player 1 to Player $N - 1$ play some profile x in the U'' neighbourhood of x^*. Fix a function η_l, where

$$\eta_l(x) = 0.5 + \Delta_{d_l}(x)$$

There exists a *Levy* game H_{η_l} with player set $N - 1 \cup \{\alpha_l^1, \alpha_l^2\}$ in which the new set of binary players $\{\alpha_l^1, \alpha_l^2\}$ play with Player 1 to Player $N - 1$ such that the game simulates the function $\eta_l(x)$. The payoffs depend upon x (the other $N - 1$ players' strategy profile in $\tilde{C}^{l.0}$), and in any equilibrium θ_l of H_{η_l}, we have $(\theta_l^{\alpha_l^1}, \theta_l^{\alpha_l^2}) = \eta_l(x)$.

Proof. Using Theorem 3.2 from Levy (2016), as highlighted in Section 2.3.4, the proof of this claim is identical to that of Claim 5.

■

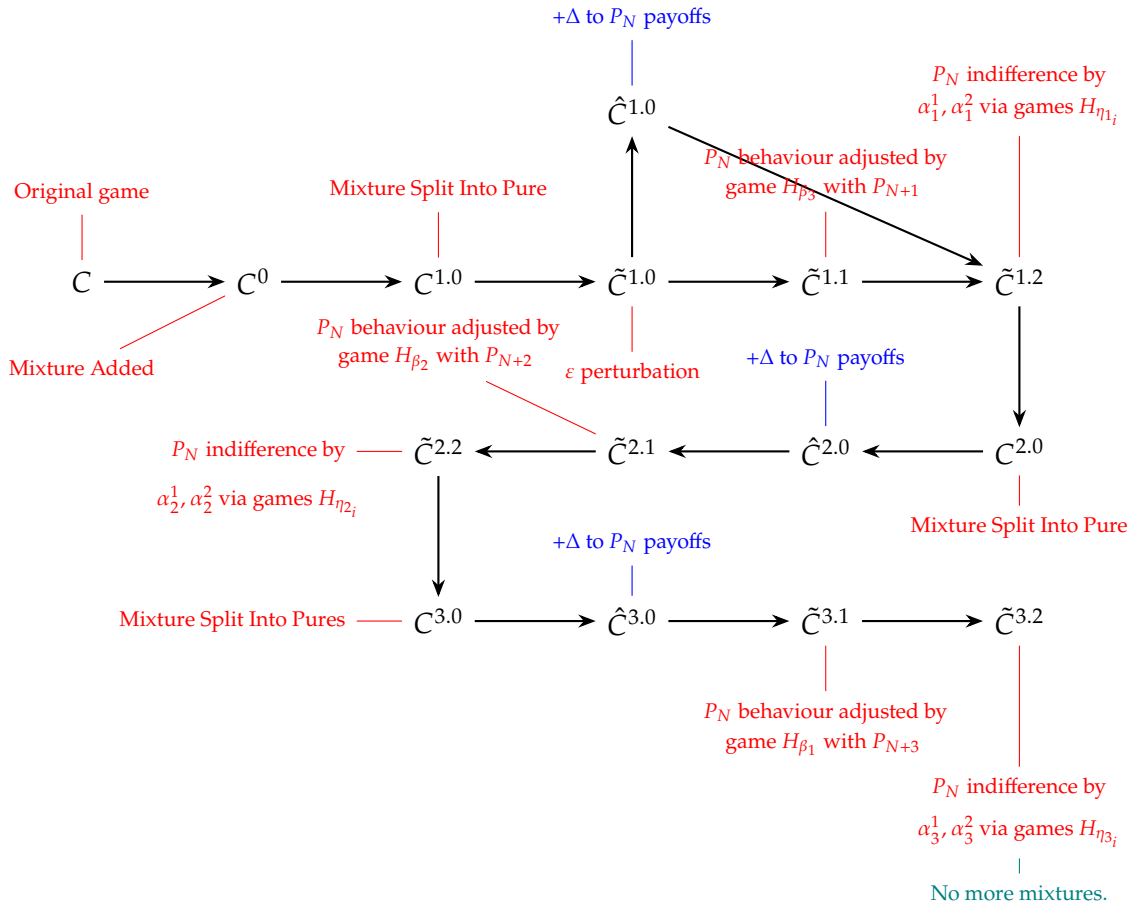


Figure 2.7 Construction of game $\tilde{C}^{l.2}$ when d is a mixture of 4 actions

Corollary 2. *Let $\tilde{C}^{l.1}$ be an ε-perturbation of $C^{l.0}$ and in which Player N plays the game $\delta \cdot (H_{\beta_{j-1}})$ against Player k and let Player 1 to Player $N-1$ play some profile x in the U'' neighbourhood of x^*, where x^* is the equilibrium response profile for Player 1 to Player $N-1$. Let us modify the payoffs of Player N using the outcome from game H_{η_l} to obtain game $\tilde{C}^{l.2}$ where there are $N+3l$ players (the full set of players at level l is: $N \cup \{k_1, k_2, ..., k_l\} \cup \{\alpha_1^1, \alpha_1^2, ..., \alpha_l^1, \alpha_l^2\}$) such that:*
for the mixed duplicate action d_l:

$$\tilde{C}_N^{l.2}(x, d_l, z_{aux}, u_l, v_l) = \tilde{C}_N^{l.1}(x, d_l, z_{aux}) + (u_l - 0.5)$$

and for the pure duplicate action a'_j:

$$\tilde{C}_N^{l.2}(x, a'_j, z_{aux}, u_l, v_l) = \tilde{C}_N^{l.1}(x, a'_j, z_{aux}) + \left(\frac{-p^*}{1-p^*}(u_l - 0.5)\right)$$

where:

- $j = n + 1 - l$,
- z_{aux} *collectively denotes the strategies of all previous auxiliary players (i.e., the* $l - 1$ *auxiliary players* k_i *and the* $2(l-1)$ *binary players* α_i^1, α_i^2 *for* $i < l$*, as well as the current level's auxiliary player* k*),*
- u_l *and* v_l *are the equilibrium strategies of the binary players* α_l^1 *and* α_l^2 *in game* H_{η_l}*.*

In any equilibrium profile of the game $\tilde{C}^{l.2}$*, Player* N *is indifferent between the two duplicate actions* a'_j *and* d_l*.*

Proof. *For the new game* $\tilde{C}^{l.2}$*, in any equilibrium profile* $(x^*, y^*, z^*_{aux}, u^*, v^*)$ *it holds that* $\tilde{C}^{l.2}_N(x^*, d_l, z^*_{aux}, u^*, v^*) = \tilde{C}^{l.2}_N(x^*, a'_j, z^*_{aux}, u^*, v^*)$*, due to Claim 7 and Claim 8.*

■

2.6 Inductive Extension: N-Player Games and an Equilibrium Component is Eliminated Using Multiple Complex Mixtures

In this case, we consider a game D whose payoff-robust equilibrium component E^* can only be eliminated by adding mixed strategy duplicates to more than one player. These players whose duplicate actions eliminate all equilibria within a neighbourhood U' around E^* belong to a set S and no subset of S, and no smaller mixture than the one specified to each player in S, can successfully eliminate the equilibrium component E^* from U'.

In such a scenario, we contend that applying the construction from Section 2.5 to each player's duplicate mixture strategy iteratively will solve the problem of eliminating equilibrium component E^* using pure duplicates. In this way, we have completed the construction from simple mixtures and singleton equilibria to full equilibrium components, and onward through the addition of more complex and multiple mixtures, thereby completing the proof of Theorem 1. ■

Chapter 3

Solving Polymatrix Games

3.1 Introduction

Polymatrix games, introduced by Yanovskaya (1968) and Howson (1972), are n-player games in strategic form with a particular compact description that allows expressing their Nash equilibria as solutions to a linear complementary problem (LCP), closely related to bimatrix games. Each player has a finite number of strategies, and chooses simultaneously with all other players one strategy (possibly by randomisation as a mixed strategy). For each pair of players, a separate *bimatrix* game describes their payoffs when they choose their respective strategies. Each player receives as overall payoff the *sum* over all other players of the payoffs from these pairwise interactions. The game is thereby specified by $n(n-1)$ payoff matrices of suitable dimensions.

Polymatrix games occupy an important position in the hierarchy of game representations. They generalise two-player bimatrix games (where $n = 2$) while forming a strict subclass of general n-player games. A polymatrix game can be seen as a special *graphical game* on a complete graph. For polymatrix games, the strategic interaction among the agents has the key property that the payoffs are linear in mixed strategy probabilities for any number of players. This linearity ensures that the expected utility for any individual can be decomposed into a sum of pairwise interactions and simplifies the computational complexity of the equilibrium set.

Due to this linearity property, the Nash equilibria of the game are solutions to a Linear Complementarity Problem (LCP). Howson (1972) demonstrated that an equilibrium for polymatrix games can be found by applying the algorithm of Lemke (1965). The main aim of this chapter is to show that a specific parametrisation of

Lemke's algorithm effectively implements the (linear) *tracing procedure* of Harsanyi and Selten (1988). The methodology presented in this article serves as a direct extension of the path-following approach already established for two-player games, now generalised for the multi-player polymatrix setting.

While finding an equilibrium remains a primary goal, to decide the existence of a pure Nash equilibrium in a polymatrix game G is an NP-hard problem. This classification implies that no polynomial-time algorithm is known to exist for this task unless P = NP. The contribution of this chapter is twofold. First, we present a formal proof that establishes the NP-completeness of deciding the existence of pure Nash equilibria in polymatrix games. Second, we develop and implement a tracing-based algorithm for equilibrium computation in polymatrix games that extends from the classical bimatrix tracing procedure. We apply this algorithm to random starting points.

The theoretical hardness results together with constructive algorithmic exploration aims to deepen understanding of the computational landscape of polymatrix games. Beyond their intrinsic interest in algorithmic game theory, the findings shed light on the broader challenge of designing practical algorithms for equilibrium computation in complex multi-agent systems.

The remainder of this chapter is organised as follows: Section 3.2 provides a comprehensive review of the literature that covers the development of computational methods for Nash equilibria from two-player bimatrix games to recent multi-player polymatrix settings. Section 3.3 formally defines polymatrix games, their pairwise payoff matrices and total payoff decomposition. In Section 3.4, we prove that it is NP-complete to decide if a polymatrix game has a pure ϵ-Nash equilibrium. Section 3.5 focuses on mixed Nash equilibria, describes the transformation of polymatrix games into standard Linear Complementarity Problems (LCPs), and details the implementation of Lemke's algorithm. Section 3.6 explains the Linear Tracing Procedure and our randomised choice of starting points. We applies the algorithm to a specific 6×6 bimatrix game with a large number of mixed Nash equilibria. Finally, Section 3.7 details a *Tracing Backwards* Breadth-First Search (BFS) algorithm, which allows us to find many more equilibria than from randomised starting points alone.

3.2 Literature

The development of computational methods for Nash equilibria began with two-player games. The Lemke-Howson algorithm, introduced in the 1960s, provided the first practical pivoting method for computing equilibria in bimatrix games by tracing complementary paths through best-response polyhedra (Lemke and Howson, 1964). This algorithm became the classical approach for two-player equilibrium computation and established the foundation for complementary pivoting techniques in game theory. However, early empirical studies revealed that the Lemke-Howson algorithm could require exponential time in the worst case: Savani and von Stengel (2006) constructed classes of games where the Lemke-Howson algorithm requires exponentially many pivoting steps for finding a Nash equilibrium. Crucially, their work shows that this exponential trajectory is not merely a consequence of a poor choice of initial parameterization; rather, the path length remains exponential even in the "best case," which corresponds to the most favorable choice of the initial "missing label".

Polymatrix games emerged as an extension of bimatrix games to multi-player settings. In a polymatrix game, each player's payoff is expressed as the sum of payoffs from pairwise interactions with other players. This representation is particularly valuable for modelling distributed systems, network interactions, and multi-agent scenarios where pairwise relationships dominate. The polymatrix form allows games with n players to be represented using $O(n^2)$ bimatrix payoff tables rather than the exponentially large tables required for general normal-form games.

The Linear Complementarity Problem (LCP) encodes (among other applications) the Nash equilibrium condition for bimatrix games and polymatrix games. An LCP is defined by a given square matrix M and vector q and asks to find vectors w and z that satisfy $w = Mz + q, w \geq 0, z \geq 0, w^\top z = 0$ (see also (3.1) below). Because w and z are nonnegative, the orthogonality condition $w^\top z = 0$ is equivalent to the *complementarity* condition that for each index i, either $w_i = 0$ or $z_i = 0$ (or both), that is, $w_i z_i = 0$.

Applied to games, the components z_i of z are decision variables that either denote payoffs (and are then dependent variables), or mixed strategy probabilities for pure strategies i with corresponding payoff *slacks* w_i. For a pure strategy i, the complementarity condition $w_i z_i = 0$ states that i is either played with positive probability ($z_i > 0$) because the strategy is a best response ($w_i = 0$) or the strategy is not a best response ($w_i > 0$) and so the strategy is not played ($z_i = 0$). This captures exactly the *Best Response Condition* of Nash equilibrium that a strategy profile is a Nash

equilibrium if and only if for every player, any pure strategy played with positive probability is a best response to the remaining players' strategies (von Stengel, 2022, Prop. 6.1, Prop. 6.3).

We focus on computing exact equilibria rather than approximations. It is important to note that while exact equilibria are fully tractable in bimatrix and polymatrix games due to the rational numbers generated by the LCP formulation, this does not hold widely. For general n-player games where $n \geq 3$, Nash equilibria can involve irrational coordinates. This representation obstacle creates a fundamental barrier for standard digital encoding. It is often the reason to use ϵ-approximate equilibrium definitions in non-linear settings (Etessami and Yannakakis, 2010).

Thus, the LCP framework offers several distinctive advantages for polymatrix games that make it the natural choice for exact equilibrium computation in this setting. The LCP conditions represent the equilibrium conditions. Unlike approximate methods such as gradient descent, LCP-based approaches compute exact equilibria, with solutions that can be verified to satisfy equilibrium conditions. The LCP framework connects polymatrix game computation to the broader theory of complementarity problems in operations research and mathematical programming.

Lemke's algorithm, a complementary pivoting method for solving LCPs, has played a central role in equilibrium computation for games. The algorithm operates by introducing an artificial variable and following a piecewise-linear path through the feasible region. It performs pivot operations that maintain complementarity while seeking a solution to the original problem. For LCPs with matrices in certain classes (including copositive matrices), Lemke's algorithm is guaranteed to find a solution or determine that none exists within the standard framework.

The relationship between Lemke's algorithm and the broader family of complementary pivoting methods has been extensively studied. A unified view of Lemke's algorithm and the Lemke-Howson algorithm reveals that both follow similar pivoting principles but differ in their initialisation and path-following strategies (von Stengel, 2023). The Lemke-Howson algorithm, specific to bimatrix games, allows only a restricted finite set of paths, while more general pivoting approaches can explore a richer space of solution trajectories.

The problem of *equilibrium selection* has been a central concern in game theory since the recognition that most games possess multiple equilibria. Harsanyi and Selten (1988) developed an equilibrium selection theory based on the tracing procedure, a mathematical construction that adjusts arbitrary prior beliefs into equilibrium beliefs.

The linear tracing procedure, a specific implementation of this concept, provides both a selection criterion and a computational approach.

Lemke in our version implements the (linear) Harsanyi-Selten tracing procedure. Harsanyi and Selten normally use the "centroid" uniform mixed strategies as a starting point to select an equilibrium. The starting point is the "prior" that players assume they play against first when starting the procedure, and then adjust their reactions to a mixture of the prior and actual play until the prior has weight zero, which finds the selected equilibrium of the game.

We choose the starting point randomly as a mixed strategy for each player uniformly from their mixed strategy simplex. By running this over a number of random starting points, we can identify which equilibria are found more often, as an additional and possibly better selection criterion than the uniform prior. In addition, if always the same equilibrium is found, then this is some sort of empirical "uniqueness" of the Nash equilibrium, which in general is NP-hard to decide (Gilboa and Zemel, 1989; Conitzer and Sandholm, 2008).

Lemke's algorithm is in practice running in a linear number of pivoting steps in the LCP dimension. An enumeration method for bimatrix games, such as *lrsnash* (Avis, Rosenberg, Savani, and von Stengel, 2010), considers all vertices of polytopes associated with the game, which is inherently exponential. It would be completely infeasible for games of size 100×100, where Lemke still runs in seconds. Furthermore, an algorithm for enumerating all mixed equilibria for polymatrix games has not yet been fully developed for games that may be degenerate. Finally, even for bimatrix games, complete enumeration does not give the information about equilibrium selection.

The van den Elzen and Talman (1999) algorithm implements the linear tracing procedure for bimatrix games through a complementary pivoting approach that can start from any mixed strategy pair (called a prior). The free choice of the starting pair allows the generation of infinitely many possible paths. This flexibility distinguishes it from the Lemke and Howson (1964) algorithm, which explores only a restricted finite set of paths. Balthasar (2010)'s analysis revealed important relationships between these algorithms. She proved that the van den Elzen and Talman (1999) algorithm is a special case of the global Newton method introduced by Govindan and Wilson (2003). However, the algorithm does not behave identically to the Lemke-Howson algorithm when started from the same pure strategy pair. In von Stengel, van den Elzen, and Talman (2002), it was shown that the algorithm of van den Elzen and Talman (1999) is a special case of Lemke's algorithm.

Polymatrix games are particularly relevant in the study of computational complexity and equilibrium discovery. Unlike general n-player games, which are difficult to solve, the computation of a mixed Nash equilibrium has variants of different computational complexity. For general n-player games with $n \geq 3$, if a Nash equilibrium is meant to be found exactly, Etessami and Yannakakis (2010) show that this problem belongs to their class FIXP that includes other fixed point problems related to major open problems in the complexity of numerical computation.

An approximate fixed point or Nash equilibrium is somewhat easier to compute, because an approximate fixed point may still be far away from an actual fixed point. The problem of ϵ-Nash equilibrium belongs to the class PPAD. This class, introduced by Papadimitriou (1994), stands for "polynomial parity argument with direction", of which the Lemke-Howson algorithm and Lemke's algorithm is a prime example.

A problem in PPAD is defined by an implicitly defined directed graph where each node (typically encoded by a k-place binary string and thus an element of an exponentially large set with 2^k elements) has indegree and outdegree at most 1 each. Isolated nodes (with indegree and outdegree zero) are disregarded. A node with indegree zero is called a source, a node with outdegree zero is called a sink, and one source is given. It is assumed that the outgoing and ingoing neighbours of a node can be found in polynomial time (the first "P" in "PPAD"). Clearly, there are as many sources as sinks (the parity argument). The problem is to find a source or sink other than the given source.

For Lemke-Howson, the given source is the artificial equilibrium, and the paths are pivoting steps, which can be computed in polynomial time. In Chen and Deng (2006), the problem of finding a Nash equilibrium of a 2-player game has been shown to be PPAD-complete. Solving an LCP by complementary pivoting as in Lemke's algorithm is a similar problem in PPAD.

Polymatrix games share the same complexity class as two-player bimatrix games. What truly distinguishes polymatrix and bimatrix games from general n-player games is their internal bilinear payoff structure. This allows the equilibrium conditions to be cast as a LCP and enables complementary pivoting paths. With polymatrix games, we target a class that is both expressive enough to describe complex social and economic networks and structured enough to allow for an exploration of the equilibrium set that is efficient in practice: Empirically, Lemke's algorithm seems to require a linear number of pivoting steps in the problem dimension, similar to the Simplex algorithm for Linear Programming (Dantzig, 1963) – although a systematic study and explanation of this observation has not yet been conducted.

3.3 Polymatrix Games

In a polymatrix game, the set of players $[n] = \{1, \ldots, n\}$ each possesses a finite set of pure strategies. The distinguishing feature of this model is that the payoff to player i is not a single value determined by the joint strategy profile of all n players. Instead, it is the sum of payoffs from $n - 1$ separate bimatrix games played simultaneously against every other opponent $j \in [n] \setminus \{i\}$.

Game Definition

Consider a set of n players, denoted by $[n] = \{1, 2, \ldots, n\}$.

1. **Strategy Sets:** Each player $i \in [n]$ has a finite set of pure strategies $[m^i] = \{1, 2, \ldots, m^i\}$. A pure strategy profile is a tuple $s = (s^1, \ldots, s^n)$, where $s^i \in [m^i]$.

2. **Mixed Strategies:** A mixed strategy for player i is a probability distribution over $[m^i]$, denoted by $x^i \in \Delta^i$, where

$$\Delta^i = \left\{ x^i \in \mathbb{R}^{m^i} \;\middle|\; \sum_{k=1}^{m^i} x^i_k = 1 \text{ and } x^i_k \geq 0 \text{ for all } k \in [m^i] \right\}.$$

   We denote a mixed-strategy profile by $x = (x^1, \ldots, x^n)$.

3. **Pairwise Payoff Matrices:** For every ordered pair of players (i, j) where $i \neq j$, there is an $m^i \times m^j$ payoff matrix A^{ij}. The entry A^{ij}_{kl} is the payoff player i receives when she plays strategy k and player j plays strategy l.

4. **Total Payoff:** The total payoff is the sum of the payoffs received from her pairwise interactions with all other players. If x^i is the mixed strategy of player i and x^j is the strategy of player j, the payoff to player i from the interaction with j is given by ${x^i}^\top A^{ij} x^j$, where A^{ij} is the payoff matrix for player i in their specific game against j. The total payoff for player i is then

$$U_i(x) = \sum_{j \neq i} {x^i}^\top A^{ij} x^j$$

   where A^{ij} is the $m^i \times m^j$ payoff matrix player i receives from player j.

   Because the payoff is additive across pairwise interactions, the utility of player i is determined by the sum of the strategies of all other players j. This property,

described as *linearity in the strategies of others*, is the critical feature that allows polymatrix games to be mapped directly into the Linear Complementarity Problem (LCP) framework.

5. A strategy profile is a Nash equilibrium if no player can strictly increase their expected payoff by unilaterally deviating.

3.4 NP-Completeness of Pure ϵ-Nash Equilibria of Polymatrix Games

In their Theorem 5, Simon and Apt (2015) have shown that existence of a (pure) Nash equilibrium in a polymatrix game is NP-complete, by showing that this property holds for their *social network games*, which can be encoded as polymatrix games. Their NP-completeness proof reduces the problem PARTITION to the existence of a Nash equilibrium in a social network game. Our proof is more direct, by a reduction of the canonical SAT problem to the existence of a pure Nash equilibrium in a polymatrix game. In addition, we prove the stronger statement that it is NP-complete to decide if a polymatrix game has a pure ϵ-approximate Nash equilibrium.

The following proposition is proved by a direct reduction from SAT. Similar to Chu and Halpern (2001) and von Stengel and Forges (2008, Figure 9) for payoff-sums in extensive games, we use a reduction from SAT with clauses and Boolean variables played by certain agents. The construction of Gottlob, Greco, and Scarcello (2005) for graphical games may possibly also apply or be adaptable to polymatrix games, but is not described as such. The stated bound $\frac{1}{5}$ for ϵ is not sharp but coarse enough to make this a strong result.

Proposition 1. *It is NP-complete to decide if a polymatrix game has a pure-strategy ϵ-approximate Nash equilibrium for $0 \le \epsilon \le \frac{1}{5}$.*

Proof. For a given pure strategy profile $s \in \prod_{i \in N} A^i$, it is easy to verify if no player $i \in N$ has an incentive to deviate unilaterally from their strategy s^i without gaining more than ϵ in utility, so the problem is in NP.

We first consider the case $\epsilon = 0$, which is the case of a standard pure Nash equilibrium. The proof is by reduction from a SAT formula in conjunctive normal form with n Boolean variables, given by the conjunction of m clauses, each of which is a disjunction of *literals*, each of which is a negated or non-negated variable. The

decision problem is to decide if all clauses can be made true by a suitable truth assignment to the n variables. SAT is one of the most widely used problems for NP-completeness proofs, usually in its form 3SAT where each clause has at most three literals (Garey and Johnson, 1979).

The polymatrix game $G = (N, \{A^i\}_{i \in N}, \{U_i\}_{i \in N})$ has $n + m + 1$ players. The player set N is the disjoint union of N_V, N_C, and $\{Out\}$. There are n variable players in N_V, furthermore m clause players in N_C, and one outside player Out. Each variable player $i \in N_V$ has action space $A^i = \{T, F\}$, where T and F correspond to setting the ith Boolean variable to true or false, respectively. Any combination of pure actions for the variable players N_V corresponds to a truth assignment of the Boolean variables. If the jth clause in the formula that has ℓ literals, the respective clause player $j \in N_C$ has $\ell + 2$ actions A^j. Of these actions, ℓ correspond to the literals in the clause, and the clause player j and the variable player i receive payoff $U_j = 1$ if the variable for player i makes that literal true (and $U_j = 0$ otherwise). The clause player j has two further actions H and T (for "heads" and "tails") where he interacts with Out in a constant-sum matching pennies game, as follows.

The outside player Out has three actions $A_{Out} = \{H, T, B\}$. In any pairwise interaction with a clause player $j \in N_C$, player Out when playing B gets a small payoff $U_{Out} = \frac{1}{3m}$ for a "literal" action of the clause player j and zero when the clause player j chooses H or T. Against the outside player Out playing B, each clause player $j \in N_C$ gets payoff $U_j = \frac{1}{4}$ when playing H or T and zero otherwise. If Out plays H or T, he gets payoff $U_{Out} = 1$ if the choice H or T of the clause player j matches, that is, for the pairs (H, H) and (T, T), and payoff $U_{Out} = 0$ for the non-matching pairs (H, T) and (T, H); for the clause player j the opposite payoffs apply. All other payoffs U_i are zero. Figure 3.1 shows an example. In fact, the payoffs to the variable players could even be set to zero so that these players are completely indifferent among their actions; their only purpose is to supply payoffs to the clause players via their actions as truth assignments.

We claim that *any* assignment of truth values, via the corresponding pure actions of the variable players N_V, can be extended to a Nash equilibrium of this polymatrix game G by best responses of the clause players N_C and Out. However, as we will show, this Nash equilibrium is in pure strategies if and only if all clauses are satisfied.

Consider any truth assignment defined by pure actions of the variable players N_V. Let $S \subseteq N_C$ and $U \subseteq N_C$ be the sets of clause players for the satisfied and unsatisfied clauses, respectively. The best response of any clause player $j \in S$ is to choose a literal

Out \ clause player	H	T	$\neg x_1$	x_3	x_5
H	0 1	1 0	0 0	0 0	0 0
T	1 0	0 1	0 0	0 0	0 0
B	$\frac{1}{4}$ 0	$\frac{1}{4}$ 0	0 $\frac{1}{3m}$	0 $\frac{1}{3m}$	0 $\frac{1}{3m}$

x_1 player \ clause player	H	T	$\neg x_1$	x_3	x_5
true	0 0	0 0	0 0	0 0	0 0
false	0 0	0 0	1 1	0 0	0 0

x_3 player \ clause player	H	T	$\neg x_1$	x_3	x_5
true	0 0	0 0	0 0	1 1	0 0
false	0 0	0 0	0 0	0 0	0 0

Figure 3.1 Example payoffs for the outside player *Out* and two variable players x_1 and x_3 in interaction with a clause player for the clause $(\neg x_1 \vee x_3 \vee x_5)$.

that makes that clause true, with resulting payoff $U_j = 1$ from the variable player $i \in N_V$.

The best response of any clause player $j \in U$ is *not* to choose any literal in the clause, which would give payoff zero, but instead to choose H or T, or possibly a mixture of H and T, for example with equal probability, which gives that player j at least payoff $U_j = \frac{1}{4}$ from the interaction with *Out*. This is higher than the payoff $U_j = 0$ achievable for any literal in the clause because none of these literals is true.

Against this behaviour of the clause players N_C, no variable player $i \in N_V$ can improve her payoff U_i (which is some nonnegative integer that counts the clause

players N_C that choose the respective literal) by changing the truth value of her variable, because no clause player $j \in N_C$ chooses the opposite literal.

The response of *Out* varies. Suppose $U = \emptyset$, that is, all clauses are satisfied. Then *Out* uniquely chooses B with cumulative payoff $U_{Out} = \frac{1}{3}$, because H or T would give *Out* payoff $U_{Out} = 0$. This defines the claimed pure strategy equilibrium.

Consider the other case that $u = |U| > 0$. Then *Out* when playing B receives $\sum_{j \in U} \frac{1}{3m} < \frac{1}{3}$, whereas playing H and T with equal probability gives him payoff $U_{Out} = \frac{u}{2}$, which is higher. We claim *Out* can only play this mixed strategy in any Nash equilibrium (when $u > 0$). First of all, a Nash equilibrium results only when *Out* is indifferent between H and T. This can be achieved in various ways, for example if up to half of the clause players $j \in U$ play H and the same number play T (recall that *Out* receives the sum of his payoffs against them) and the rest of the players in U play H and T with equal probability. Second, because *Out* plays the same matching pennies game against the clause players $j \in U$, any mixture of *Out* between H and T other than $\frac{1}{2}, \frac{1}{2}$ would induce a pure best response of the clause players $j \in U$ that is the opposite of what *Out* prefers. In particular, H or T is not a pure equilibrium strategy of *Out*.

Hence, if and only if the Boolean formula is unsatisfiable, the constructed polymatrix game G does not have a pure equilibrium.

The same construction applies for $\epsilon = \frac{1}{5}$ (or any smaller value of ϵ; the bound $\frac{1}{5}$ is not sharp). Clearly, if the SAT formula is satisfiable, we can obtain a pure Nash equilibrium as described, which is also an ϵ-Nash equilibrium. Conversely, suppose that the formula is not satisfiable, and consider a pure strategy profile. We claim that at least one clause player can deviate from their current strategy and gain at least $\frac{1}{5}$ in utility, which shows that the profile is not an ϵ-equilibrium. Let U be the nonempty set of unsatisfied clauses as before. Irrespective of the action of *Out*, a player in U will choose H or T to get payoff at least $\frac{1}{4}$, which is a gain of more than $\frac{1}{5}$ compared to choosing a literal in the clause. Player *Out* may well be indifferent between H and T (but will not choose action B) if exactly half of the players in U play H and the other half play T, but *Out* chooses a pure action H or T. But then at least one player in U can improve their payoff by 1 by changing their strategy, namely from H to T if *Out* plays H, and from T to H if *Out* plays T. This violates the condition of ϵ-equilibrium.

We have reduced the NP-complete problem SAT to the decision problem of whether a polymatrix game G has a pure ϵ-Nash equilibrium. Hence, this decision problem is NP-complete, as claimed. ■

3.5 Mixed Nash Equilibria: LCP and Lemke's Algorithm

The computation of mixed Nash equilibria in polymatrix games rests upon the ability to reformulate the equilibrium conditions into a mathematical framework known as the Linear Complementarity Problem (LCP). This section describes the transformation of a polymatrix game into a standard LCP, and details the execution of Lemke's algorithm as given in von Stengel (2023) for tracing equilibrium paths.

The LCP provides a unified algebraic framework for solving for Nash equilibria. The key is to transform the inequality and complementarity conditions into the standard LCP form: $w = q + Mz \geq 0$, with $w \geq 0$, $z \geq 0$, and $w^\top z = 0$ (Cottle, Pang, and Stone, 2009). The LCP is defined by finding z such that:

$$z \geq 0 \quad \perp \quad w = q + Mz \geq 0\,. \tag{3.1}$$

Here, we utilise the symbol $\perp$ to denote orthogonality and thus complementarity between two nonnegative difference vectors of equal dimension. That is, $a \geq b \perp c \geq d$ stands for $(a - b)^\top (c - d) = 0$. Equivalently, $(a_i - b_i)(c_i - d_i) = 0$ for every index i.

3.5.1 The LCP Formulation: w = q + Mz

A mixed Nash equilibrium of a polymatrix game is a strategy profile $(x^1, x^2, \ldots, x^n)$ in $\Delta^1 \times \Delta^2 \times \cdots \times \Delta^n$ such that for every player $i \in \{1, \ldots, n\}$, there exists a real number u_i that represents that player's equilibrium payoff according to

$$x^i \geq 0 \quad \perp \quad \sum_{j \neq i} A^{ij} x^j \leq \mathbf{1} u_i\,.$$

The complementarity condition ensures that if player i assigns a positive probability to some pure strategy k (i.e., $x^i_k > 0$), then that strategy is a best response against the other players' mixed strategy profile.

In a Linear Complementarity Problem (3.1), the variables w and z are nonnegative. In order to represent a polymatrix game as an LCP with nonnegative variables, we employ several variable transformations. This extends an approach known for the bimatrix case in von Stengel (2023) to n players. First, we address the unrestricted equilibrium payoff variables u_i for each player $i \in \{1, \ldots, n\}$. These can be represented

as the difference of two nonnegative variables:

$$u_i = u_i^+ - u_i^-, \quad u_i^+ \geq 0, \quad u_i^- \geq 0\,.$$

The normalisation equations for the mixed strategies of all n players, $\mathbf{1}^\top x^i = 1$, are represented as two inequalities for each player i:

$$\begin{aligned} u_i^+ \;&\geq\; 0 \quad \perp \quad -\mathbf{1}^\top x^i \;\geq\; -1\,, \\ u_i^- \;&\geq\; 0 \quad \perp \quad \mathbf{1}^\top x^i \;\geq\; 1\,. \end{aligned} \tag{3.2}$$

By choosing these specific complementarity conditions, the constraint matrix M captures the interactions between all players.

It is possible and useful to omit the first complementarity condition in (3.2) by guaranteeing all payoffs to be *negative*. This is achieved by a "*negshift*" change of subtracting a suitable constant from every payoff, which does not change the game, as follows.

This requirement to transform payoff matrices into strictly negative values is a technical step in mapping a polymatrix game to a standard Linear Complementarity Problem (LCP) suitable for Lemke's algorithm. In a standard LCP (q, M), the goal is to find a vector $z \geq 0$ such that $w = q + Mz \geq 0$ and $z \perp w$. However, in game theory, the equilibrium payoff variables (u_i) are generally unrestricted in sign; they could be positive, negative, or zero depending on the game data. Lemke's algorithm requires all variables in the z-vector to be nonnegative $(z \geq 0)$. If u_i is positive, it cannot be placed directly into z. While one could represent u_i as the difference of two nonnegative variables $(u_i = u_i^+ - u_i^-)$, this increases the dimension of the LCP and adds complexity to the pivoting logic. By ensuring payoffs are strictly negative, we can simplify the variable representation to a single nonnegative variable, u_i^-.

Any game A^{ij} can be transformed into an equivalent game with strictly negative payoff matrices by subtracting a sufficiently large constant from every entry in the matrices. For polymatrix games, each pairwise interaction matrix A^{ij} must be adjusted such that $A^{ij} < 0$ for all i, j. Note that this shift does not change the set of mixed Nash equilibria or the best-response conditions; it merely shifts the equilibrium payoff u_i by a constant, so that the utilities are guaranteed to be strictly negative. This allows us to simplify the variables significantly by setting the positive components to zero:

$$u_i^+ = 0, \quad u_i = -u_i^-\,.$$

The full polymatrix LCP system for n players then simplifies to the following conditions for each player i:

$$\begin{aligned} x^i \geq 0 \quad &\perp \quad \textstyle\sum_{j \neq i} -A^{ij} x^j - \mathbf{1} u_i^- \geq 0 \,, \\ u_i^- \geq 0 \quad &\perp \quad \mathbf{1}^\top x^i \geq 1 \,. \end{aligned}$$

In any solution $(x^1, \ldots, x^n, u_1^-, \ldots, u_n^-)$ to this system, the fact that all payoffs are negative implies that $u_i^- = -u_i > 0$. The complementarity condition then forces the last inequality to be tight ($\mathbf{1}^\top x^i = 1$), so that each x^i is a valid mixed strategy belonging to player i's strategy simplex.

The LCP is constructed over the total number of pure strategies $m = \sum_{i=1}^n m^i$ and n utility variables, where n is also the number of equality constraints for the mixed strategy variables (one equation per player). The LCP variables z and w are defined as:

- **Activity Variables (z):** z is partitioned into the vector of mixed strategies x and the vector of expected utility variables $v = -u$:

$$z = \begin{pmatrix} x \\ v \end{pmatrix} \in \mathbb{R}^{m+n}$$

- **Slack Variables (w):** This vector primarily represents the *utility slacks* (w^u), which are the payoffs player i *loses* by deviating from an optimal mixed strategy. The second set of variables, the *probability slacks* (w^y), are nonnegative slack variables corresponding to the probability constraints (as argued before, in any LCP solution these slacks w^y will have to be zero, but this is initially not the case):

$$w = \begin{pmatrix} w^u \\ w^y \end{pmatrix} \in \mathbb{R}^{m+n} \,.$$

The problem (3.1) is to find $w \geq 0, z \geq 0$ such that $w^\top z = 0$ and $w = q + Mz$.

The Complementarity Condition. The LCP requires that for every i, either $w_i = 0$ or $z_i = 0$. When applied to the NE, this means:

1. **For Strategy Probabilities and Utility Slacks:** For each pure strategy k of player i, we require that the probability x_k^i and the corresponding slack utility variable $w_{i,k}^u$ are complementary: $w_{i,k}^u \cdot x_k^i = 0$.

   - If $x_k^i > 0$ (the strategy is played), then $w_{i,k}^u$ is 0. This states that strategy k yields the maximum possible payoff (it is a best response).
   - If $w_{i,k}^u > 0$ (the strategy yields lower than maximum payoff), then x_k^i is 0 (the strategy is not played).

   This exactly captures the condition for a Nash Equilibrium.

2. **For Expected Utilities and Probability Slacks:** The complementary variables v and w^y relate to the fixed sum of probabilities constraint ($\sum_{k=1}^{m^i} x_k^i = 1$). The LCP uses $v_i = -u_i$ to calculate the slack utilities $w_{i,k}^u$ (or regret) for every other strategy. Because in a Nash equilibrium every player must have their probabilities on strategies sum to 1, then $w_i^y = 0$. This allows the complementary variable v_i, the (negative of the) expected utility, to be any positive value. The LCP cannot be solved until the probability slacks w_i^y are zero. This is guaranteed because the negshift transformation forces the equilibrium utility to be negative ($u_i < 0$), making the LCP expected utility variable v_i (where $v_i = -u_i$) strictly positive. By the complementarity condition $v_i \cdot w_i^y = 0$, a positive v_i necessitates a zero value of w_i^y, which in turn ensures the strategy probabilities sum exactly to one.

We now explicitly construct the matrix M and the vectors q for an n-player polymatrix game.

The Matrix M. The square matrix M is the heart of the LCP. It encodes all the interaction payoffs and probability constraints. For an n-player polymatrix game where each player i has m^i strategies, the dimensions of the blocks are determined by the total number of pure strategies, $m = \sum_{i=1}^{n} m^i$, and the number of players, n. The matrix M is of size $N \times N$, where $N = m + n$. Let A^{ij} be the $m^i \times m^j$ payoff matrix for player i when playing against player j. After applying the negshift transformation ($A^{ij} < 0$) and the variable substitution $u_i = -u_i^-$, the polymatrix-game LCP matrix M is defined as follows:

$$M = \begin{bmatrix} 0 & -A^{12} & -A^{13} & \dots & -A^{1n} & -\mathbf{1} & 0 & \dots & 0 \\ -A^{21} & 0 & -A^{23} & \dots & -A^{2n} & 0 & -\mathbf{1} & \dots & 0 \\ -A^{31} & -A^{32} & 0 & \dots & -A^{3n} & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -A^{n1} & -A^{n2} & -A^{n3} & \dots & 0 & 0 & 0 & \dots & -\mathbf{1} \\ \mathbf{1}^\top & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & \mathbf{1}^\top & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \mathbf{1}^\top & 0 & 0 & \dots & 0 \end{bmatrix}$$

The Vector q. The q vector represents the constants in the LCP equations. It is partitioned into two main sections:

$$q = \begin{pmatrix} q^u \\ q^y \end{pmatrix}$$

The first m rows of the LCP represent the utility constraints. For player i, the expected payoff for pure strategy k is $\sum_{j \neq i} A^{ij} x^j$. In the constraint $w = q + Mz \geq 0$, the respective constraints state $-\sum_{j \neq i} A^{ij} x^j - \mathbf{1} v_i \geq 0$. Hence, for these m inequalities we have $q^u = 0$. The bottom n rows enforce the probability sum $\sum_{k=1}^{m^i} x_k^i = 1$. This is handled by setting the constant terms to ensure $q^y = -\mathbf{1}$ (a vector of -1's in $\mathbb{R}^n$).

3.5.2 Lemke's Algorithm

Lemke's algorithm (Lemke, 1965) is a pivotal method, similar in style to the Simplex algorithm for linear programming (Dantzig, 1963), but adapted to maintain a state of "almost-complementarity" throughout the process. Lemke's algorithm finds a solution by traversing a sequence of almost-complementary basic solutions, expressed using a dictionary (or tableau), until the auxiliary variable z_0 is removed.

Basic and Non-Basic Variables

The LCP equation $w = q + Mz$ can be rewritten with the $N \times N$ idenity matrix $\mathbf{I}$ as

$$\mathbf{I}w - Mz = q\,.$$

A set of variables, known as the basic variables, are chosen to form the basis matrix composed of the corresponding columns. The remaining variables are non-basic and are set to zero. An LCP solution requires that the basic variables are all nonnegative and satisfy the complementarity condition ($w^\top z = 0$).

Lemke's algorithm maintains the following property: for almost every index i, either w_i or z_i is in the basis (i.e., they are complementary). The algorithm traces a path through a sequence of adjacent bases that are almost-complementary until a true complementary basis is found.

The Role of the Auxiliary Variable z_0

Since $q^y = -\mathbf{1}$, the vector q may contain negative entries, and the initial solution $z = 0, w = q$ yields $w^y = -\mathbf{1}$, which is not feasible as it violates $w \geq 0$. To overcome this, we introduce the auxiliary variable $z_0 \geq 0$ and a positive vector $d \geq 0$ (e.g., $d = \mathbf{1}$) to augment the system:

$$z \geq \mathbf{0}, \quad z_0 \geq 0, \quad w = q + dz_0 + Mz, \quad w^\top z = 0 \tag{3.3}$$

Any solution z, z_0 to this system is *almost complementary*, and *complementary* if $z_0 = 0$, which implies (3.1). The positive vector d is such that $q_i < 0 \Rightarrow d_i > 0 \quad (1 \leq i \leq N)$. This implies that for $z = \mathbf{0}$ and all sufficiently large z_0 we have $w = q + dz_0 \geq \mathbf{0}$ and $z^\top w = 0$, so that (3.3) holds. The set of almost complementary solutions where $z = 0$ is called the *primary ray* (Cottle, Pang, and Stone, 2009). The augmented system (3.3) has $(m + n) \times (m + n + 1)$ matrix entries in d and M.

The Starting Point and Pivoting

Initial Basic Solution: To initialise the algorithm, an initial value for z_0 is chosen to be large enough to make all w variables nonnegative:

$$z_0 = \max_i \left\{ \frac{-q_i}{d_i} \mid q_i < 0 \right\}$$

Let B denote the basis. The array of basic variables is defined as $x_B = (x_i)_{i \in B}$. The initial basis B has variables z_0 and all w_i for which $q_i > 0$. The variable w_r that

corresponds to the most negative q_r is the one that forces z_0 and is therefore the first *non-basic* variable to leave the basis as z_0 enters.

In any given basic solution of the augmented LCP system $w - Mz - dz_0 = q$, the basic variables x_B (a subset of $\{w, z, z_0\}$) are expressed in terms of the non-basic variables x_N, which are the remaining variables:

$$x_B = \bar{b} - \bar{A}x_N \tag{3.4}$$

where $\bar{A}$ represents the transformed coefficient matrix and $\bar{b}$ represents the current values of the basic variables if the non-basic variables x_N are set to zero. This basic solution is feasible if $\bar{b} \geq 0$. Importantly, the system (3.4), often called a *tableau*, is fully equivalent to the original system (3.3), which requires maintaining $\bar{A}$ and not just $\bar{b}$.

Pivoting. The algorithm proceeds by pivoting, changing the basis B (where $i \in B$ means x_i is a basic variable) by swapping one non-basic variable x_j (the entering variable) with one basic variable x_i (the leaving variable), while maintaining almost-complementarity. Pivoting means changing basis from B to $B - \{i\} \cup \{j\}$ for some $j \in N$, with entering variable x_j, and $i \in B$ indicating the leaving variable x_i.

1. **Entering Variable (j):** During the initial pivot, z_0 enters and w_r leaves. The next entering variable z_r is the complement of the variable w_r that was non-basic in the initial solution. Thereafter, an entering variable x_j is the complement of the variable x_i that just left the basis (in first pivot, $x_j = z_0$, later $x_j = z_i$ if w_i was leaving before or $x_j = w_i$ if z_i was leaving before).

2. **Leaving Variable (i):** In the execution of Lemke's algorithm for solving the polymatrix LCP, the *Minimum Ratio Test* is the critical mechanism used to identify which variable must leave the basis during a pivoting step. Once an entering variable is selected, either the auxiliary variable z_0 during initialization or the complement of the variable that just left the basis, the algorithm must determine how much that variable can increase before one of the current basic variables violates its nonnegativity constraint. ($x_B \geq 0$).

   The test operates on the current tableau representation of the system $w = q + Mz + dz_0$. Let B^{-1} be the basis matrix and $b = B^{-1}q$ be the current values of the basic variables. If j is the column in the tableau corresponding to the entering variable, the minimum ratio test calculates the ratio of the current value of each basic variable to its corresponding (positive) entry in the entering

column:

$$\theta = \min_{i:\, \bar{A}_{ij}>0} \left(\frac{b_i}{\bar{A}_{ij}} \right)$$

The row i that achieves this minimum ratio θ identifies the leaving variable. This variable is driven to zero and the entering variable takes the value θ in order to maintain the feasibility of the system ($z \geq 0, w \geq 0$).

3. **Path Following:** When a variable, say z_j, enters the basis:

   - Its complement, w_j, which was basic before, is forced out of the basis and becomes non-basic (zero).
   - z_j becomes positive (non-basic $\rightarrow$ basic).
   - The entry of z_j causes a basic variable, say w_k, to hit zero first (determined by the minimum ratio test) and leave the basis.
   - Because w_k just left the basis, its complement, z_k, is designated as the next variable to enter the basis.

   The algorithm then follows a path of almost-complementary basic solutions. In each step, a non-basic variable is introduced, and its complementary variable leaves, maintaining the almost-complementarity. The process terminates when z_0 leaves the basis, leading to a complementary solution where $z_0 = 0$. Once z_0 is gone, the resulting feasible basic solution for w and z is a complementary solution, which corresponds to a Nash equilibrium of the polymatrix game. And once an equilibrium is found, all the $v_i = u_i^-$ variables are positive because all payoffs u_i are negative, and by complementarity the current inequalities for the probabilities (probability slacks w^y) therefore have to be zero, meaning that they are actual probabilities. This is essential to solve the LCP, as Lemke algorithm cannot terminate (with z_0 leaving the basis) before $w_i^y = 0$.

3.5.3 Integer Pivoting

Integer Pivoting, as described by von Stengel (2007) is a specialised method for solving systems of linear equations with integer coefficients. Unlike traditional Gaussian elimination, which often introduces fractions or floating-point rounding errors, integer pivoting preserves all coefficients as integers throughout the computation. This exactness is critical for computational efficiency and numerical stability, as it ensures that Gaussian elimination remains a polynomial-time algorithm Grötschel,

Lovász, and Schrijver (1988, section 1.4). While the method is frequently attributed to Edmonds (1967), its conceptual origins likely date back to J. J. Sylvester.

The defining feature of integer pivoting is the use of a previous pivot to scale down coefficients. In standard cross-multiplication (eliminating variables without fractions), the magnitude of coefficients grows exponentially with each step, quickly becoming computationally unmanageable. Integer pivoting solves this by performing a clever division by the pivot element from the preceding iteration.

Note that these divisions are guaranteed to result in integers without remainders. This is because each pivot element is equivalent to the determinant of the submatrix (minor) consisting of the rows and columns used in previous steps. These determinants are the denominators in solutions derived via Cramer's rule.

Consider the following, randomly chosen set of linear equations in three variables x, y, z:

$$\begin{aligned} 7x + 2y + 5z &= 1 \\ x + 3y + 11z &= 3 \\ 13x - 2z &= 0 \end{aligned} \tag{3.5}$$

We now eliminate x from the second and third equation in two steps, first by multiplying each equation by the coefficient 7 of x, called the *pivot element*:

$$\begin{aligned} 7x + 2y + 5z &= 1 \\ 7x + 21y + 77z &= 21 \\ 13 \cdot 7x - 14z &= 0 \end{aligned}$$

and then subtracting the respective multiple of the first row from rows 2 and 3:

$$\begin{aligned} 7x + 2y + 5z &= 1 \\ 19y + 72z &= 20 \\ -26y - 79z &= -13 \end{aligned}$$

Next, we proceed similarly to eliminate y from the first and third row: Multiply those rows by the coefficient 19 of y:

$$\begin{aligned} 7\cdot 19x &+ 2\cdot 19y &+ 95z &= 19 \\ & 19y &+ 72z &= 20 \\ -\,26\cdot 19y &- 1501z &= -247 \end{aligned}$$

and then subtract the respective multiple of the second row from rows 1 and 3:

$$\begin{aligned} 7\cdot 19x & - 49z &= -21 \\ 19y &+ 72z &= 20 \\ & 371z &= 273 \end{aligned} \tag{3.6}$$

The key point of (3.6), and it is somewhat surprising that this works, is that all coefficients of the modified rows 1 and 3 are divisible by the previous pivot element 7, so that they simplify to

$$\begin{aligned} 19x & - 7z &= -3 \\ 19y &+ 72z &= 20 \\ & 53z &= 39 \end{aligned} \tag{3.7}$$

The final step is to eliminate z from rows 1 and 2 in the same manner: Multiply them by the coefficient 53 of z as the new pivot element:

$$\begin{aligned} 19\cdot 53x & - 7\cdot 53z &= -159 \\ 19\cdot 53y &+ 72\cdot 53z &= 1060 \\ & 53z &= 39 \end{aligned}$$

Then subtract the respective multiple of row 3 from rows 1 and 2:

$$\begin{aligned} 19\cdot 53x & &= 114 \\ 19\cdot 53y & &= -1748 \\ & 53z &= 39 \end{aligned}$$

and divide (without remainder) those rows by the previous pivot element 19:

$$\begin{aligned} 53x & &= 6 \\ 53y & &= -92 \\ & 53z &= 39 \end{aligned}$$

It is readily verified that $x, y, z = \frac{6}{53}, \frac{-92}{53}, \frac{39}{53}$ solve the original system (3.5).

Note that the pivot elements are *determinants* of the currently used variables, and are thus the denominators in the solutions using Cramer's rule. Namely, 7 is the coefficient of x, that is, the determinant of the 1×1 matrix of the first row and column in (3.5), and 19 in (3.7) is the determinant $\begin{vmatrix} 7 & 2 \\ 1 & 3 \end{vmatrix}$ of the first two rows and columns, and 53 is the determinant $\begin{vmatrix} 7 & 2 & 5 \\ 1 & 3 & 11 \\ 13 & 0 & -2 \end{vmatrix}$ of the entire 3×3 matrix of coefficients.

Integer pivoting ensures that every entry in the tableau remains an integer throughout the pivoting process, which ensures an *exact* computation that finds exact equilibria for games with rational payoffs.

Furthermore, the algorithm can reliably execute the lexicographic minimum ratio test, described next.

Degeneracy and the Lexicographical Rule. Degeneracy is a critical phenomenon in any method based on linear inequalities, such as the Simplex algorithm for linear programming. It also applies to the study of Nash equilibria, particularly within the LCP framework used for polymatrix games. It occurs when a vertex of the polytope is defined by more than the minimum number of intersecting hyperplanes. In practical terms, this means that during a pivot operation, multiple variables reach zero simultaneously, rendering the result of the standard minimum ratio test not unique and making the choice of the leaving variable ambiguous.

If the minimum ratio test yields multiple candidates for the leaving variable, the path traced by Lemke's algorithm may cycle: a state where the algorithm indefinitely pivots between different bases of the same degenerate vertex without making progress. To prevent this and guarantee termination, a perturbation technique called the *lexicographical minimum ratio test* is employed. This method examines the entire row of the tableau to determine the lexicographically smallest, or most limiting, ratio by comparing rows element-wise from left to right. This ensures a unique leaving variable is always chosen, effectively shielding the pivoting process from infinite loops.

The robustness of this lexicographical rule is enhanced by integer pivoting which maintains coefficients as exact integers and allows for a precise lexicographic tie-breaking rule. This exactness is essential; while standard floating-point methods might drift due to rounding errors and trigger incorrect tie-breaking, integer-based arithmetic ensures that the lexicographical test is performed on precise rational values, accurately identifying when a strategic set is closed.

3.6 The Linear Tracing Procedure and Equilibrium Discovery

The algorithm for our equilibrium discovery is the van den Elzen and Talman (1999) linear tracing procedure, which is a special case of Lemke's algorithm, as shown by von Stengel, van den Elzen, and Talman (2002). This method suggests that for any Linear Complementarity Problem (LCP), the set of solutions can be reached by following different paths determined by the covering vector d, which itself is determined by a *prior* strategy profile that serves as a starting point for the algorithm. Instead of using for d a fixed vector of ones (a common choice for Lemke's algorithm), the implementation samples starting vectors from a uniform distribution on the mixed-strategy simplices of all players.

The Tracing Procedure interprets this path-following as a process of reasoning from a *prior* belief, i.e., a preconception of how opponents might play, to a consistent Nash equilibrium of the actual game. Using Lemke's algorithm with a specific covering vectors derived from such a priors, it traces a unique path of adjacent vertices of the polyhedron defined by the LCP constaints $q + Mz + dz_0 \geq 0$ and $z \geq 0$. By generating a large number of random starting points, we can explore different regions of the equilibrium set and locate multiple equilibria.

Let $\bar{x} = (\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^n)$ be a fixed *prior* mixed-strategy profile in the mixed-strategy space. For a parameter $\tau \in [0, 1]$, we define a restricted game where each player i believes that every other player j will play a strategy $x^j(\tau)$ that is a weighted average of their actual choice $\tilde{x}^j$ and the prior $\bar{x}^j$:

$$x^j(\tau) = (1 - \tau)\tilde{x}^j + \tau\bar{x}^j$$

At $\tau = 1$, players respond only to the prior, which generically yields a unique best response. As τ decreases (not necessarily monotonically) to eventually 0, the weight

shifts from the prior to the actual strategies being played in the game, ending at the original polymatrix game when $\tau = 0$ (von Stengel, 2023).

To implement this procedure within the LCP framework, we map the parameter τ to the auxiliary variable z_0. We redefine the LCP variables x^i as scaled versions of the mixed strategies of the actual play $\hat{x}^i$:

$$x^i = (1 - z_0)\hat{x}^i \ .$$

Since the actual strategies sum to one ($\mathbf{1}^\top \hat{x}^i = 1$), the scaled variables x^i satisfy the condition $\mathbf{1}^\top x^i = 1 - z_0$. This transformation allows us to use the standard LCP format while modifying the covering vector d to reflect the prior. In the augmented system $w = q + Mz + dz_0$, the covering vector $d = (d^u, d^y)^\top$ must be chosen so that at $z_0 = 1$ (the prior), the system remains complementary.

- The row for the slack utility $w^u_{i,k}$ in our block matrix M is defined by $w^u_{i,k} = v_i - \sum_{j \neq i} A^{ij} x^j + d^u_{i,k} z_0$, where $v_i = u_i^-$. At the initialisation where $z_0 = 1$ and $x = 0$, we require the slack to represent the difference between the equilibrium payoff and the payoff against the prior. This is achieved by setting:

  $$d^u_{i,k} = -\sum_{j \neq i} A^{ij} \bar{x}^j \ .$$

  This vector represents the negative of the expected payoffs for player i when all other players j play their prior strategy $\bar{x}^j$. Because the payoff matrices A^{ij} themselves are negative, d is a positive vector.

- The rows for the probability slacks w^y in the LCP are defined as $w^y_i = -1 + \sum_k x^i_k + d^y_i z_0$. We require $w^y_i = 0$ along the path because its complement v_i is generally nonzero. Since $\sum_k x^i_k = 1 - z_0$, we have:

  $$w^y_i = -1 + (1 - z_0) + d^y_i z_0 = -z_0 + d^y_i z_0$$

  For w^y_i to remain zero, we must set:

  $$d^y_i = 1$$

  for all players $i \in \{1, \ldots, n\}$. Hence, d is positive in all components.

The procedure begins when z_0 enters the basis. At this point, the strategies x are zero. The equilibrium payoff variables $v_i = u_i^-$ are initialised to the maximum payoff

Algorithm 1: Linear Tracing Procedure (Complementary Pivoting)

Input : Game G, TraceCount T
Output: Equilibrium Solution x^*

```
TraceLTP(G, T):
enter ← z_0                          // Initialise auxiliary variable
leave, z_0leave ← lexminvar(enter)
while true do
    // 1. Execute Pivot Operations
    pivot(leave, enter)                          // Integer Pivoting
    // 2. Check Termination Condition
    if z_0leave is true then
        break          // Tracing complete, z_0 has left the basis
    // 3. Complementary Selection
    enter ← complement(leave)
    leave, z_0leave ← lexminvar(enter)     // Lexicographic tie-breaking
    pivotcount ← pivotcount + 1
// 4. Extract Solution from Tableau
return solution x*
```

each player can receive against the prior:

$$u_i^- = \max_{k \in [m^i]} \left(\sum_{j \neq i} A^{ij} \bar{x}^j \right)_k .$$

This makes at least one slack variable $w^u_{i,k}$ zero for each player, corresponding to their best response to the prior.

As Lemke's algorithm pivots, z_0 (the influence of the prior) decreases (although it may intermittently increase, see von Stengel, van den Elzen, and Talman, 2002 for an example). The algorithm traces the path of Nash equilibria for the restricted games parametrised by τ. When z_0 finally leaves the basis and reaches zero, the scaled strategies x^i become unscaled strategies ($\mathbf{1}^\top x^i = 1$), and the result is a mixed Nash equilibrium of the original polymatrix game.

We implement a systematic discovery process by choosing priors $\bar{x}$ from a uniform distribution over the mixed-strategy simplices of all players. For each player i, we sample a vector $\bar{x}^i$ such that $\bar{x}^i_k \geq 0$ and $\sum \bar{x}^i_k = 1$. By repeating the Tracing Procedure (LTP) with a large set of these random priors, we identify a diverse set of equilibria. This random sampling allows us to break out of the well-known path traced by a single prior. The samples also provide a statistical distribution of which equilibria are

the most accessible or stable from random initial conditions. By recording how often a specific equilibrium is reached from various random priors, we gain an empirical measure of that equilibrium's *prevalence* or stability.

Tracing Perfect Equilibria

In bimatrix games, it is well established that weakly dominated strategies cannot be part of a perfect equilibrium. Consequently, for two-player games, the set of perfect equilibria coincides exactly with the set of undominated equilibria (see van Damme (1991, Corollary 2.2.6). While van Damme established that this undominance property does not universally extend to general n-player games, Belhaiza (2014, Theorem 5) shows that the unique additive nature of polymatrix games allows this property to be recovered. Specifically, every perfect equilibrium of a polymatrix game is undominated, and every undominated equilibrium is perfect. The theorem explicitly concludes that a Nash equilibrium is perfect if and only if it is a best response to every pure strategic choice of the opponents – a condition that directly certifies the equilibrium as undominated.

This theoretical distinction has implications for the Linear Tracing Procedure. Because the LTP is typically initiated from a completely mixed strategy profile, it essentially simulates the trembling hand perturbations used to define perfectness, guiding the homotopy path toward perfect and undominated equilibria in polymatrix games. Even if the procedure is purposefully initiated at a non-perfect pure point to explore the set, the internal mechanics of the algorithm often favour stability over weak dominance.

During the pivoting process, the algorithm utilises the lexicographical minimum ratio test to resolve ties at degenerate vertices. If the order of strategies in the payoff matrices places a non-perfect strategy in a position where the lexicographic rule prevents its consideration, the tracing procedure may bypass a weak equilibrium entirely, even if initiated from a non-perfect pure starting point. The lexicographical rule will steer the path toward the perfect equilibrium. This makes the discovery of the strategic set sensitive to the initial indexing and the formal ordering of strategies within the LCP framework.

Consider a simple bimatrix example in Table 3.1 with the following payoffs:

In this game, the profile *(B, L)* yields a payoff of $(0, 0)$, while *(T, R)* yields $(1, 1)$. The strategy T and R for both players weakly dominates the alternative. The profile *(B, L)*

	L	R
T	0,0	1,1
B	0,0	0,0

Table 3.1 Bimatrix game with perfect and weak equilibria

where both players receive 0 represents a weak, non-perfect equilibrium. Because this equilibrium is weakly dominated, the LTP will typically fail to reach this point. Even if the algorithm is forced to start in the neighbourhood of this weak equilibrium, the first pivot under the lexicographical rule will identify the dominant strategy of the two choices as the more "limiting" or stable choice.

For the row player, the payoffs are identical (0) when the opponent plays L. This creates a tie in the algorithm's decision making process. When the standard minimum ratio test fails to identify a unique variable to leave the basis, the lexicographical rule looks deeper into the tableau rows and compares them element wise from left to right. In the presence of weak dominance, the row corresponding to the dominant strategy typically contains values that make it lexicographically smaller than the row of the weakly dominated strategy. The rule treats the dominant strategy as the boundary that must be hit first to maintain the mathematical integrity of the path toward a perfect equilibrium. Consequently, the LTP will pivot the strategies toward the perfect equilibrium at $(1, 1)$.

A special example: 6×6 game

-81, 72	36, 36	-126, 17	126, -3	-36, -36	90, -153
-180, -180	72, -81	-333, -30	297, 20	-153, 90	270, 270
20, 297	-3, 126	42, 42	-33, -33	17, -126	-30, -333
-30, -333	17, -126	-33, -33	42, 42	-3, 126	20, 297
270, 270	-153, 90	297, 20	-333, -30	72, -81	-180, -180
90, -153	-36, -36	126, -3	-126, 17	36, 36	-81, 72

Table 3.2 The full 6×6 bimatrix game (A, B)

The 6×6 bimatrix game given in Table 3.2 serves as a crucial benchmark for evaluating the Linear Tracing Procedure (LTP) in our polymatrix framework. This game was specifically selected as a rigourous benchmark because it is known to possess a complex strategic landscape containing 75 distinct equilibria. This high density of solutions provides an ideal environment to test the efficiency of the Linear

Tracing Procedure and our randomised discovery process in capturing both dominant and elusive points on the equilibrium set. By treating the two-player case as a special case of polymatrix games where $n = 2$, we can analyse how the selection of a prior strategy profile $(\bar{x}, \bar{y})$ influences the final equilibrium reached. In this specific instance, the payoff matrices A and B are non-zero-sum and exhibit a range of values from -333 to 297. Before processing, these matrices undergo a negshift to ensure all payoffs are strictly negative. This represents the use of u_i^- variables in the LCP tableau.

To map the basins of attraction within this strategic landscape, we performed 100 independent trials. In each trial, a prior $(\bar{x}, \bar{y})$ was sampled from a uniform distribution over the 5-dimensional strategy simplices. These priors define the initial direction of the homotopy path in the augmented system $w = q + Mz + dz_0$.

The table below summarises the findings from a random sample of 100 priors for the 6×6 test case. Prevalence is used as an indicator of the practical stability of each equilibrium.

	Player 1 Strategy	**Player 2 Strategy**	**Prevalence**
1	$(0, 0, 0, 0, 1, 0)$	$(1, 0, 0, 0, 0, 0)$	44
2	$(0, 1, 0, 0, 0, 0)$	$(0, 0, 0, 0, 0, 1)$	49
3	$(\frac{19}{32}, 0, 0, \frac{13}{32}, 0, 0)$	$(0, 0, 0, 0, \frac{70}{103}, \frac{33}{103})$	2
4	$(0, 0, 0, 0, \frac{19}{70}, \frac{51}{70})$	$(0, 0, 0, \frac{4}{27}, \frac{23}{27}, 0)$	1
5	$(\frac{70}{103}, \frac{33}{103}, 0, 0, 0, 0)$	$(0, 0, 0, \frac{13}{32}, \frac{19}{32}, 0)$	2
6	$(0, 0, 0, 0, \frac{33}{103}, \frac{70}{103})$	$(0, \frac{19}{32}, \frac{13}{32}, 0, 0, 0)$	1
7	$(0, 0, \frac{4}{27}, 0, 0, \frac{23}{27})$	$(0, 0, 0, 0, \frac{51}{70}, \frac{19}{70})$	1

Table 3.3 Strategies and Prevalence for the 7 identified equilibria out of 100 random priors.

The data reveals that some equilibria (like the pure strategy equilibria in this game) often possess significantly larger basins of attraction compared to the other equilibria (some completely mixed equilibria in this game). This observation aligns with our expectation that some mixed equilibria can be difficult to find using a generic prior. Identifying these low-prevalence equilibria is a key motivation for our *Tracing Backwards* Breadth-First search, which uses discovered points to jump toward less accessible regions of the equilibrium set.

3.7 Tracing Backwards

The initial discovery phase using the Linear Tracing Procedure (LTP) provides a *forward* map from random priors to Nash equilibria. However, as demonstrated by the 6×6 case study, forward tracing from generic priors is heavily biased toward equilibria with large basins of attraction, predominantly pure strategies or low-support mixed strategies. In games with high strategic complexity, such as the selected benchmark with 75 known equilibria, many solutions remain hard-to-reach. To navigate the high strategic complexity of a polymatrix game, we implement an exploration algorithm that traces backwards. This method uses the conceptual framework of the LTP, however, instead of treating an equilibrium as a terminal destination, we treat it as a base (or seed) to discover its strategic neighbours. By treating discovered equilibria as priors for new tracing runs, we can navigate the equilibrium set in a breadth-first manner, in which we jump from one equilibrium to the other.

The fundamental idea of tracing backwards is to utilise a known equilibrium x^* as a "prior" ($\bar{x} = x^*$), exploiting the linearity of payoffs in polymatrix games. In the standard LTP, we trace from an arbitrary prior $\bar{x}$ to an equilibrium x^*. In backwards tracing, we take that same equilibrium x^* and use it to define the initial conditions of a new tracing procedure. This means the prior $\bar{x}$ is no longer a random point in the simplex, but a point that already satisfies the best-response conditions of the original game. When the LTP starts from this specific point, the algorithm is significantly more likely to:

- Re-discover a known equilibrium: If the basin of attraction around x^* is stable, the path for $z_0 \in [0, 1]$ will simply track the existing equilibrium, confirming its robustness.
- Identify *Sibling* Equilibria: The homotopy path may lead to a previously undiscovered equilibrium that shares significant strategy support with the base equilibrium.

This process essentially performs a systematic *sweep* of the equilibrium set and identifies all the distinct equilibria without needing to rely on the luck of a generic prior.

The implementation of this backward step remains within the robust LCP framework defined in Section 3.5. When we set the prior $\bar{x}$ equal to a discovered equilibrium x^*, the extraction of the basis inverse matrix and the construction of the covering

vector d become the significant next steps. The basis inverse matrix (B^{-1}) is the mathematical key that unlocks the local geometry of the LCP at any given point in the algorithm. In Lemke's algorithm, we are dealing with a high dimensional system: $Iw - Mz - dz_0 = q$, where the matrix M encodes the $n(n-1)$ pairwise interaction tables of the polymatrix game. At any step, the algorithm is at a vertex of a polytope. This vertex is defined by a basis (B), a subset of columns that are linearly independent. To understand where we are and how we can move, we must express the entire system in terms of the current basic variables. Multiplying the original system by B^{-1} performs this transformation. We obtain the basis inverse matrix by extracting it directly from the transformed identity columns of the integer-based tableau. Because we represent the initial system as $[I \mid -M \mid -d \mid q]$, the columns that originally contained the identity matrix I are updated during each pivot to store the current basis inverse, scaled by the system's determinant. With the use of Integer Pivoting (see Section 3.5.3), we maintain these entries as exact integers and recover the precise B^{-1}.

The covering vector d acts as a *compass* for the algorithm. When tracing backwards from a known equilibrium x^*, we set the prior $\bar{x} = x^*$. We then calculate the prior payoffs, which are the expected payoffs each player receives within the polymatrix game at that specific equilibrium. The d vector is built by taking the negative of these payoffs for the utility rows and setting the probability constraint rows to 1 as described in Section 3.6. The resulting covering vector d is multiplied by a new scale factor, to ensure all entries remain consistent with the integer-based coordinate system used in the tableau.

The Breadth-First Search. The Breadth-First Search (BFS) mechanism views the set of Nash equilibria of a polymatrix game as nodes in a graph, where edges represent transitions facilitated by the pivoting rules. The BFS architecture ensures that we do not just find the easiest equilibria, but also uncover those hidden in deep or narrow basins of attraction.

The BFS loop functions as follows:

1. Queue Initialisation: The process begins with a set of *seed* equilibria found during the initial random sampling phase. These are labeled as *Stage 1* equilibria and added to a processing queue.

2. Node Expansion: For each polymatrix equilibrium in the queue, the algorithm extracts the basis inverse B^{-1} from the final integer tableau.

Algorithm 2: BFS-style Equilibrium Exploration

Input : Game G, Initial trace count T
Output: Set of discovered equilibria E

```
BFSExplore(G, T):
E ← traceLTP(G, T)                          // Stage 1: Initial forward traces
ID_count ← |E| + 1
S_1 ← {e ∈ E | e.stage = 1}                  // Pool of Stage 1 directions
i ← 0
while i < |E| do
    base ← E[i]
    for each direction ∈ S_1 do
        if base.ID = direction.ID then
            continue
        for each prior ∈ direction.priors do
            // Redundancy Check
            if ∃p ∈ base.parent s.t. p.direction.ID = prior then
                continue
            // Attempt Back-tracing
            Binv ← GetBInv(game, base)          // Extract basis inverse
            d ← GetCoverVec(game, direction) // Construct covering vector
            tableau ← TraceFromEq(game, Binv, d) // Complementary Pivoting
            if tableau ≠ null then
                new_eq ← GetSolution(tableau)
                if ∃match ∈ E s.t. match.strategy ≈ new_eq then
                    // Add base to parent list
                else
                    child ← Equilibrium(new_eq, stage = base.stage + 1)
                    child.ID ← ID_count, child.parent ← [base]
                    Add child to E, ID_count ← ID_count + 1
    i ← i + 1
return E
```

3. Cross-Basis Testing: The algorithm then performs a *neighbour search*. The algorithm selects a polymatrix equilibrium x^* from the queue. It takes the priors that led to this known equilibrium to serve as the new prior $\bar{x}$ and tests them against the current basis. The LTP is executed starting from this equilibrium prior. Essentially, it asks: "If we started from the perspective of Equilibrium A, but ended up in the structural basin of Equilibrium B, would we discover a new transition?"

4. Discovery: If this process leads to a previously unknown Nash equilibrium, that point is marked as a *Stage 2* equilibrium and added to the back of the queue. The search continues layer by layer (*Stage 2* equilibria are used to find *Stage 3*, and so on) until the queue is empty.

This BFS approach is essential for polymatrix games because the solution components are often connected. Moving breadth-first, we ensure that we explore all immediate neighbours of a found polymatrix equilibrium before moving deeper into the search tree, which provides a more systematic map of the polymatrix game's equilibria.

Results of Backwards Exploration on the 6×6 Game

The transition from forward random sampling to backwards BFS exploration dramatically altered the discovery rate for the 6×6 benchmark. While 30 random priors only yielded 3 equilibria, the backwards procedure utilised those 3 as a foundation to unfold the rest of the strategic landscape.

#	**Player 1 Strategy (x)**	**Player 2 Strategy (y)**	**Prevalence**
1	$(0, 1, 0, 0, 0, 0)$	$(0, 0, 0, 0, 0, 1)$	15
2	$(0, 0, 0, 0, 1, 0)$	$(1, 0, 0, 0, 0, 0)$	14
3	$\left(0, 0, 0, 0, \frac{33}{103}, \frac{70}{103}\right)$	$\left(0, \frac{19}{32}, \frac{13}{32}, 0, 0, 0\right)$	1

Table 3.4 Strategy probabilities and prevalence for *Stage* 1 equilibria identified with 30 random priors.

Equilibria with very low forward prevalence, such as equilibrium #3 in Table 3.4 which appeared only once in 30 random starts, often turned out to be hubs for the more complex mixed equilibria. Using the priors that led to the other *Stage* 1 equilibria, the algorithm was able to find completely mixed equilibria that random priors never hit. The BFS successfully navigated through the strategic families of the 6×6 game, identifying the other 72 equilibria.

The final output of the exploration is a global transition map, as visualised in Figure 3.2. This directed graph illustrates the Breadth-First Search (BFS) discovery process of all 75 Nash equilibria. Nodes represent distinct equilibria, colour coded by discovery stage (1–8). Node diameters for *Stage* 1 reflect forward prevalence (the basin of attraction from 30 random priors), while *Stage* 2+ diameters scale by

parental density (the number of distinct equilibria from which a specific child could be discovered via backwards tracing). If a *Stage* 2+ node has a large diameter, it tells us that this equilibrium is a stable destination that can be reached from many different strategic starting points, even if it wasn't visible from the initial *Stage* 1 random sampling. The directed solid silver edges represent discovery paths: an edge from Node A to Node B indicates that a base tableau associated with A was used to reach B via the BFS basis expansion.

This visualisation provides several key insights:

1. Hub Equilibria: Nodes with high in-degree represent attractor equilibria that are reachable from many different parts of the strategy space.
2. Discovery Depth: The *Stage* of each node (colour coded in the map) shows how many levels of BFS were required to reach it, highlighting the most hidden solutions.

This two-stage process of forward discovery followed by backwards exploration allows us to quantify a new property of Nash Equilibria: *Fertility*.

- Forward Prevalence tells us how attractive an equilibrium is from a generic starting point.
- Fertility tells us how productive an equilibrium is as a base for finding other equilibria.

In our 6×6 study, we found that pure strategy equilibria have high prevalence but low fertility (they often just lead back to themselves). Conversely, mixed equilibria with support sizes of 2 or 3 often have low prevalence but high fertility, acting as bridges to the most complex, completely mixed regions of the game. By combining the Linear Tracing Procedure with this Breadth-First Backwards Search, we move beyond the simple goal of finding a Nash Equilibrium. We instead provide a comprehensive map of the entire strategic set of the polymatrix game, ensuring that even the most elusive of the 75 equilibria are brought into the light (see Table 3.5).

Table 3.5 Complete Discovery Lineage and Strategy Profiles for the 75 Equilibria

ID	Stage	Strategy Profile (x, y)	Parent
1	1	$(0,1,0,0,0,0),(0,0,0,0,0,1)$	Initial
2	1	$(0,0,0,0,1,0),(1,0,0,0,0,0)$	Initial
3	1	$(0,0,0,0,\frac{33}{103},\frac{70}{103}),(0,\frac{19}{32},\frac{13}{32},0,0,0)$	Initial
4	2	$(0,\frac{1}{2},0,0,\frac{1}{2},0),(\frac{1}{2},0,0,0,0,\frac{1}{2})$	[1, 2]
5	2	$(\frac{20}{33},\frac{13}{33},0,0,0,0),(0,0,0,0,\frac{20}{33},\frac{13}{33})$	[1]
6	2	$(0,0,\frac{23}{42},\frac{19}{42},0,0),(\frac{2}{7},\frac{5}{7},0,0,0,0)$	[1]
7	2	$(0,0,0,0,\frac{13}{33},\frac{20}{33}),(\frac{13}{33},\frac{20}{33},0,0,0,0)$	[2, 3]
8	2	$(0,0,0,0,\frac{2}{7},\frac{5}{7}),(0,0,\frac{23}{42},\frac{19}{42},0,0)$	[3]
9	2	$(0,\frac{2}{27},0,0,\frac{1}{3},\frac{16}{27}),(0,\frac{2}{11},\frac{25}{66},0,0,\frac{29}{66})$	[3]
10	2	$(0,0,\frac{11}{39},0,0,\frac{28}{39}),(0,\frac{28}{39},\frac{11}{39},0,0,0)$	[3]
11	2	$(\frac{1}{4},\frac{1}{4},0,0,\frac{1}{4},\frac{1}{4}),(\frac{1}{4},\frac{1}{4},0,0,\frac{1}{4},\frac{1}{4})$	[3]
12	3	$(0,\frac{2}{13},0,0,\frac{5}{13},\frac{6}{13}),(\frac{5}{13},\frac{6}{13},0,0,0,\frac{2}{13})$	[4, 7, 9, 11]
13	3	$(\frac{6}{13},\frac{5}{13},0,0,\frac{2}{13},0),(\frac{2}{13},0,0,0,\frac{6}{13},\frac{5}{13})$	[4, 5, 11]
14	3	$(\frac{70}{103},\frac{33}{103},0,0,0,0),(0,0,0,\frac{13}{32},\frac{19}{32},0)$	[5]
15	3	$(\frac{19}{32},0,0,\frac{13}{32},0,0),(0,0,0,0,\frac{70}{103},\frac{33}{103})$	[5, 11]
16	3	$(\frac{53}{193},\frac{84}{193},\frac{45}{193},0,0,\frac{11}{193}),(\frac{53}{189},\frac{82}{189},0,0,\frac{25}{189},\frac{29}{189})$	[5, 11]
17	3	$(\frac{51}{70},\frac{19}{70},0,0,0,0),(0,\frac{23}{27},\frac{4}{27},0,0,0)$	[5]
18	3	$(0,\frac{1}{3},\frac{19}{42},\frac{3}{14},0,0),(\frac{3}{10},\frac{2}{3},0,0,0,\frac{1}{30})$	[6]
19	3	$(0,0,\frac{13}{32},0,0,\frac{19}{32}),(\frac{33}{103},\frac{70}{103},0,0,0,0)$	[6, 7, 10]
20	3	$(\frac{23}{27},0,0,\frac{4}{27},0,0),(\frac{19}{70},\frac{51}{70},0,0,0,0)$	[6]
21	3	$(0,0,\frac{31}{59},\frac{28}{59},0,0),(0,\frac{15}{19},\frac{4}{19},0,0,0)$	[6, 10]
22	3	$(0,0,0,0,\frac{19}{70},\frac{51}{70}),(0,0,0,\frac{4}{27},\frac{23}{27},0)$	[8]
23	3	$(0,\frac{1}{30},0,0,\frac{3}{10},\frac{2}{3}),(0,0,\frac{19}{42},\frac{3}{14},0,\frac{1}{3})$	[8, 9]
24	3	$(0,0,\frac{4}{19},0,0,\frac{15}{19}),(0,0,\frac{31}{59},\frac{28}{59},0,0)$	[8, 10]
25	3	$(\frac{25}{189},\frac{29}{189},0,0,\frac{53}{189},\frac{82}{189}),(0,\frac{11}{193},\frac{45}{193},0,\frac{53}{193},\frac{84}{193})$	[9, 11]
26	3	$(0,\frac{1}{3},\frac{1}{3},0,0,\frac{1}{3}),(0,\frac{1}{3},\frac{1}{3},0,0,\frac{1}{3})$	[9, 10]
27	3	$(\frac{82}{189},\frac{53}{189},0,0,\frac{29}{189},\frac{25}{189}),(\frac{84}{193},\frac{53}{193},0,\frac{45}{193},\frac{11}{193},0)$	[11]
28	3	$(\frac{11}{193},0,0,\frac{45}{193},\frac{84}{193},\frac{53}{193}),(\frac{29}{189},\frac{25}{189},0,0,\frac{82}{189},\frac{53}{189})$	[11]
29	4	$(\frac{16}{27},\frac{1}{3},0,0,\frac{2}{27},0),(\frac{29}{66},0,0,\frac{25}{66},\frac{2}{11},0)$	[13, 14, 27]
30	4	$(\frac{2}{11},0,0,\frac{25}{66},\frac{29}{66},0),(\frac{2}{27},0,0,0,\frac{16}{27},\frac{1}{3})$	[13, 15, 28]
31	4	$(\frac{28}{39},0,0,\frac{11}{39},0,0),(0,0,0,\frac{11}{39},\frac{28}{39},0)$	[14, 15, 22]
32	4	$(\frac{5}{7},\frac{2}{7},0,0,0,0),(0,0,\frac{19}{42},\frac{23}{42},0,0)$	[14, 17]
33	4	$(\frac{10}{21},0,0,\frac{2}{7},\frac{5}{21},0),(\frac{53}{180},0,\frac{1}{4},\frac{41}{90},0,0)$	[14]
34	4	$(\frac{11}{15},\frac{4}{15},0,0,0,0),(\frac{4}{15},\frac{11}{15},0,0,0,0)$	[14, 17, 19, 20]
35	4	$(0,0,\frac{19}{42},\frac{23}{42},0,0),(0,0,0,0,\frac{5}{7},\frac{2}{7})$	[15]
36	4	$(0,0,0,0,\frac{4}{15},\frac{11}{15}),(0,0,0,0,\frac{11}{15},\frac{4}{15})$	[15, 22, 25]
37	4	$(\frac{53}{131},\frac{53}{131},\frac{87}{524},0,0,\frac{13}{524}),(\frac{53}{131},\frac{53}{131},0,\frac{87}{524},\frac{13}{524},0)$	[16, 27]

ID	Stage	Strategy Profile (x, y)	Parent
38	4	$(0, \frac{29}{66}, \frac{25}{66}, 0, 0, \frac{2}{11}), (\frac{1}{3}, \frac{16}{27}, 0, 0, 0, \frac{2}{27})$	[16, 18, 19, 26]
39	4	$(\frac{4}{21}, \frac{8}{21}, \frac{31}{98}, \frac{11}{98}, 0, 0), (\frac{25}{87}, \frac{50}{87}, 0, 0, \frac{5}{87}, \frac{7}{87})$	[16, 18, 19]
40	4	$(\frac{15}{71}, \frac{28}{71}, \frac{53}{213}, 0, 0, \frac{31}{213}), (0, \frac{31}{213}, \frac{53}{213}, 0, \frac{15}{71}, \frac{28}{71})$	[16, 18, 24, 25, 26]
41	4	$(\frac{93}{112}, 0, 0, \frac{19}{112}, 0, 0), (0, \frac{93}{112}, \frac{19}{112}, 0, 0, 0)$	[17, 20, 21]
42	4	$(\frac{5}{11}, \frac{4}{11}, \frac{5}{33}, \frac{1}{33}, 0, 0), (\frac{4}{11}, \frac{5}{11}, \frac{1}{33}, \frac{5}{33}, 0, 0)$	[17, 20, 21]
43	4	$(0, \frac{53}{180}, \frac{41}{90}, \frac{1}{4}, 0, 0), (0, \frac{10}{21}, \frac{2}{7}, 0, 0, \frac{5}{21})$	[18, 21, 26]
44	4	$(0, 0, \frac{1}{2}, \frac{1}{2}, 0, 0), (0, 0, \frac{1}{2}, \frac{1}{2}, 0, 0)$	[19, 21, 24]
45	4	$(\frac{13}{524}, 0, 0, \frac{87}{524}, \frac{53}{131}, \frac{53}{131}), (0, \frac{13}{524}, \frac{87}{524}, 0, \frac{53}{131}, \frac{53}{131})$	[21, 25, 28]
46	4	$(0, 0, \frac{19}{112}, 0, 0, \frac{93}{112}), (0, 0, 0, \frac{19}{112}, \frac{93}{112}, 0)$	[22, 24]
47	4	$(\frac{5}{87}, \frac{7}{87}, 0, 0, \frac{25}{87}, \frac{50}{87}), (0, 0, \frac{31}{98}, \frac{11}{98}, \frac{4}{21}, \frac{8}{21})$	[23, 25]
48	4	$(0, \frac{5}{21}, \frac{2}{7}, 0, 0, \frac{10}{21}), (0, 0, \frac{41}{90}, \frac{1}{4}, 0, \frac{53}{180})$	[23, 24, 26]
49	4	$(\frac{50}{87}, \frac{25}{87}, 0, 0, \frac{7}{87}, \frac{5}{87}), (\frac{8}{21}, \frac{4}{21}, \frac{11}{98}, \frac{31}{98}, 0, 0)$	[27]
50	4	$(\frac{31}{213}, 0, 0, \frac{53}{213}, \frac{28}{71}, \frac{15}{71}), (\frac{28}{71}, \frac{15}{71}, 0, \frac{53}{213}, \frac{31}{213}, 0)$	[27, 28]
51	4	$(0, 0, \frac{11}{98}, \frac{31}{98}, \frac{8}{21}, \frac{4}{21}), (\frac{7}{87}, \frac{5}{87}, 0, 0, \frac{50}{87}, \frac{25}{87})$	[28]
52	5	$(\frac{2}{3}, \frac{3}{10}, 0, 0, \frac{1}{30}, 0), (\frac{1}{3}, 0, \frac{3}{14}, \frac{41}{90}, 0, 0)$	[29, 32, 33, 49]
53	5	$(\frac{1}{3}, 0, 0, \frac{1}{3}, \frac{1}{3}, 0), (\frac{1}{3}, 0, 0, \frac{1}{3}, \frac{1}{3}, 0)$	[29, 30, 31, 33, 50]
54	5	$(0, 0, \frac{3}{14}, \frac{19}{42}, \frac{1}{3}, 0), (\frac{1}{30}, 0, 0, 0, \frac{2}{3}, \frac{3}{10})$	[30, 35, 50, 51]
55	5	$(0, 0, \frac{28}{59}, \frac{31}{59}, 0, 0), (0, 0, 0, \frac{4}{19}, \frac{15}{19}, 0)$	[31, 35, 44, 46]
56	5	$(\frac{15}{19}, 0, 0, \frac{4}{19}, 0, 0), (0, 0, \frac{28}{59}, \frac{31}{59}, 0, 0)$	[31, 32, 33, 38, 41, 44]
57	5	$(0, 0, \frac{2}{7}, \frac{16}{35}, \frac{9}{35}, 0), (\frac{9}{35}, 0, \frac{2}{7}, \frac{16}{35}, 0, 0)$	[33, 44]
58	5	$(\frac{50}{197}, 0, 0, \frac{50}{197}, \frac{67}{197}, \frac{30}{197}), (\frac{53}{151}, \frac{87}{604}, \frac{93}{604}, \frac{53}{151}, 0, 0)$	[33, 49, 50]
59	5	$(0, 0, \frac{13}{189}, \frac{5}{21}, \frac{8}{21}, \frac{59}{189}), (0, \frac{1}{128}, \frac{7}{64}, 0, \frac{33}{64}, \frac{47}{128})$	[35, 36, 45, 51]
60	5	$(0, 0, \frac{4}{27}, 0, 0, \frac{23}{27}), (0, 0, 0, 0, 0, \frac{51}{70}, \frac{19}{70})$	[35, 36, 46, 51]
61	5	$(\frac{1}{128}, 0, 0, \frac{7}{64}, \frac{47}{128}, \frac{33}{64}), (0, 0, \frac{5}{21}, \frac{13}{189}, \frac{59}{189}, \frac{8}{21})$	[36, 45, 47]
62	5	$(\frac{33}{64}, \frac{47}{128}, \frac{7}{64}, 0, 0, \frac{1}{128}), (\frac{8}{21}, \frac{59}{189}, \frac{13}{189}, \frac{5}{21}, 0, 0)$	[37, 42, 49]
63	5	$(\frac{59}{189}, \frac{8}{21}, \frac{5}{21}, \frac{13}{189}, 0, 0), (\frac{47}{128}, \frac{33}{64}, 0, \frac{7}{64}, \frac{1}{128}, 0)$	[37, 39, 42]
64	5	$(\frac{87}{604}, \frac{53}{151}, \frac{53}{151}, \frac{93}{604}, 0, 0), (0, \frac{50}{197}, \frac{50}{197}, 0, \frac{30}{197}, \frac{67}{197})$	[39, 40, 43, 44, 48]
65	5	$(\frac{30}{197}, \frac{67}{197}, \frac{50}{197}, 0, 0, \frac{50}{197}), (0, 0, \frac{53}{151}, \frac{93}{604}, \frac{87}{604}, \frac{53}{151})$	[40, 47, 48]
66	5	$(0, \frac{9}{35}, \frac{16}{35}, \frac{2}{7}, 0, 0), (0, 0, \frac{16}{35}, \frac{2}{7}, 0, \frac{9}{35})$	[43, 44, 48]
67	5	$(0, 0, \frac{93}{604}, \frac{53}{151}, \frac{53}{151}, \frac{87}{604}), (\frac{67}{197}, \frac{30}{197}, 0, \frac{50}{197}, \frac{50}{197}, 0)$	[44, 50, 51]
68	6	$(0, 0, \frac{1}{4}, \frac{41}{90}, \frac{53}{180}, 0), (\frac{5}{21}, 0, 0, \frac{2}{7}, \frac{10}{21}, 0)$	[53, 54, 55, 57, 67]
69	6	$(0, 0, \frac{50}{267}, \frac{100}{267}, \frac{29}{89}, \frac{10}{89}), (\frac{29}{89}, \frac{10}{89}, \frac{50}{267}, \frac{100}{267}, 0, 0)$	[57, 58, 61, 67]
70	6	$(0, 0, \frac{1}{33}, \frac{5}{33}, \frac{4}{11}, \frac{5}{11}), (0, 0, \frac{5}{33}, \frac{1}{33}, \frac{5}{11}, \frac{4}{11})$	[59, 61, 65]
71	6	$(\frac{10}{89}, \frac{29}{89}, \frac{100}{267}, \frac{50}{267}, 0, 0), (0, 0, \frac{100}{267}, \frac{50}{267}, \frac{10}{89}, \frac{29}{89})$	[64, 65, 66]
72	6	$(0, \frac{9}{58}, \frac{10}{29}, \frac{10}{29}, \frac{9}{58}, 0), (\frac{9}{58}, 0, \frac{10}{29}, \frac{10}{29}, 0, \frac{9}{58})$	[66]
73	7	$(\frac{10}{179}, \frac{39}{179}, \frac{60}{179}, \frac{50}{179}, \frac{20}{179}, 0), (\frac{20}{179}, 0, \frac{60}{179}, \frac{50}{179}, \frac{10}{179}, \frac{39}{179})$	[71, 72]
74	7	$(0, \frac{20}{179}, \frac{50}{179}, \frac{60}{179}, \frac{39}{179}, \frac{10}{179}), (\frac{39}{179}, \frac{10}{179}, \frac{50}{179}, \frac{60}{179}, 0, \frac{20}{179})$	[72]
75	8	$(\frac{1}{30}, \frac{1}{6}, \frac{3}{10}, \frac{3}{10}, \frac{1}{6}, \frac{1}{30}), (\frac{1}{6}, \frac{1}{30}, \frac{3}{10}, \frac{3}{10}, \frac{1}{30}, \frac{1}{6})$	[73, 74]

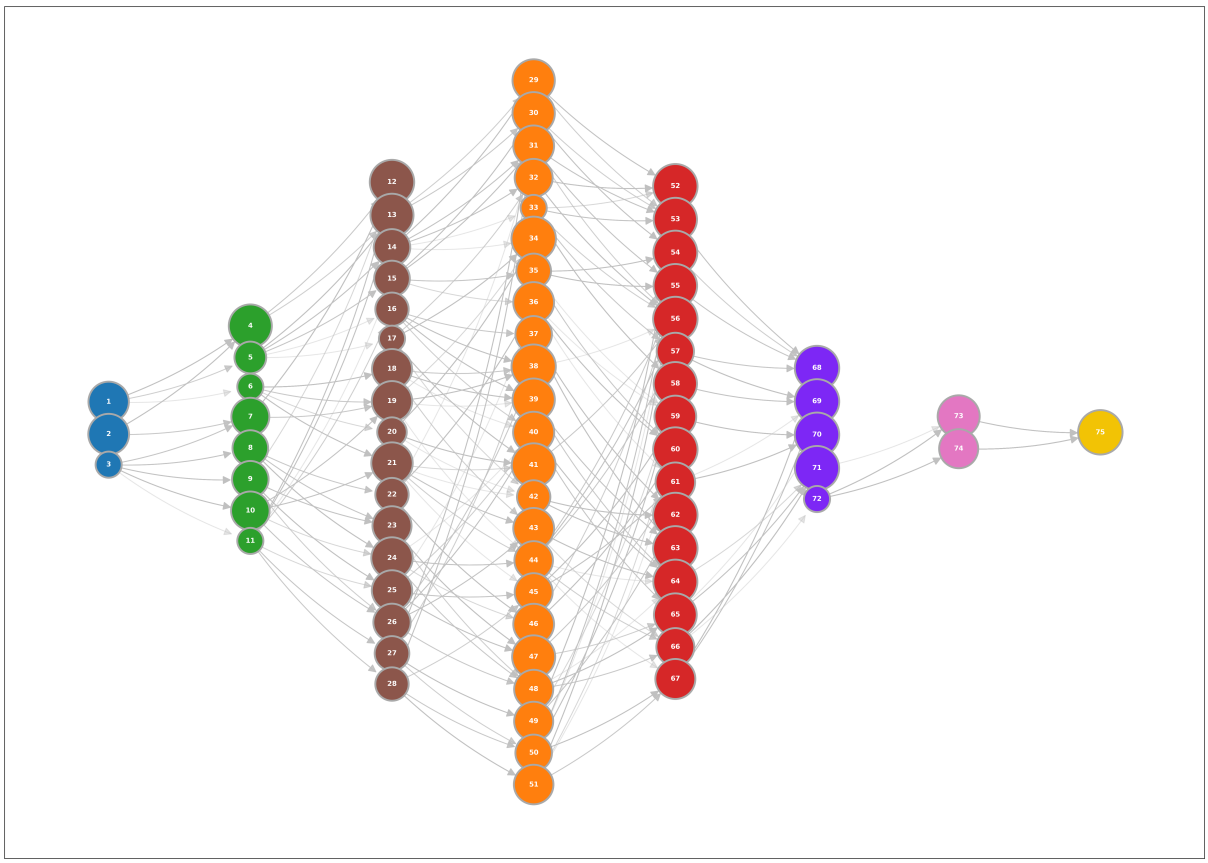


Figure 3.2 Equilibrium Stage Flow and Connection Topology of the 6×6 Game

A special example: A degenerate polymatrix $2 \times 2 \times 2$ game

To further illustrate the robustness of the *Tracing Backwards* algorithm, we present a second example: a 3-player polymatrix game with high symmetry and inherent degeneracy. In this game, each player has two pure strategies ($m = [2, 2, 2]$). The strategies are defined by the following pairwise payoff matrices, where identity matrices I represent coordination incentives for Players 1 and 2, while anti-coordination matrices ($J - I$) define the incentives for Player 3, as shown in Figure 3.3. Players 1 and 2 want to coordinate, but Player 3 wants to anti-coordinate.

Unlike the expansive 6×6 benchmark, this game is highly degenerate because multiple best-response constraints are satisfied simultaneously at the pure strategy profiles and the completely mixed profile. In such cases, standard pivoting can cycle, and we need integer pivoting and lexicographic minimum ratio test to ensure a unique leaving variable is chosen.

	P2: S_1	**P2:** S_2
P1: S_1	(1, 1)	(0, 0)
P1: S_2	(0, 0)	(1, 1)

(a) Player 1 vs Player 2 (Coordination)

	P3: S_1	**P3:** S_2
P1: S_1	(1, 0)	(0, 1)
P1: S_2	(0, 1)	(1, 0)

(b) Player 1 vs Player 3 (P3 Anti-coordination)

	P3: S_1	**P3:** S_2
P2: S_1	(1, 0)	(0, 1)
P2: S_2	(0, 1)	(1, 0)

(c) Player 2 vs Player 3 (P3 Anti-coordination)

Figure 3.3 Pairwise payoff matrices for the 3-player polymatrix game.

The initial discovery phase using LTP successfully identified three distinct Nash equilibria: a completely mixed profile $(\frac{1}{2}, \frac{1}{2})^3$ and two pure strategy coordination/anti-coordination states: $(0, 1), (0, 1), (1, 0)$ and $(1, 0), (1, 0), (0, 1)$. The second pure equilibrium is traced less often due to the lexicographic ordering at initialisation, because it does indeed resolve ties by considering the variables in the order they are arranged in the LCP tableau.

The most critical observation from the *Tracing Backwards* exploration was that every jump attempted between these equilibria resulted in a Return to *Stage* 1 equilibria. In a polymatrix context, these *returns* signify that the strategic set is completely closed. When the algorithm extracted a basis inverse (B^{-1}) and attempted to navigate to new regions using the priors of other equilibria, the paths consistently led back to the three already discovered during the random sampling phase. This behaviour confirms that these points are stable equilibrium destinations.

Finally, the stage structure uncovered by the BFS raises a question - the layered lineage visible in Table 3.5 shows that support size tends to grow monotonically with BFS depth: the support size of Stage 1 equilibria are sparse, the Stage 8 equilibrium is completely mixed, and the intermediate stages interpolate between them in a structured way. This is not surprising in retrospect. A completely mixed equilibrium requires every pure strategy of every player to be a best response simultaneously, placing it at the intersection of the maximum number of binding complementarity constraints in the LCP. Its basin of attraction under the linear homotopy is therefore very small relative to the full strategy simplex: a random prior drawn uniformly has negligible probability of inducing a Lemke path that passes through this region, which is why high-support equilibria have low forward prevalence despite being genuine Nash equilibria. The BFS sidesteps this problem entirely by constructing directions not from random points in the simplex but from the structured priors of already-discovered equilibria, which carry information about the game's payoff geometry.

3.8 Future Applications and Directions

We applied our empirical testing framework to the special classes of polymatrix games established by Deligkas, Fearnley, Igwe, and Savani (2016*a*). We generated a variety of adversarial and coordination games using the library of polymatrix game generators from Deligkas, Fearnley, Igwe, and Savani (2016*b*) to demonstrate the algorithm. The findings in Table 3.6 reveal a clear pattern: the BFS exploration yields the most significant gains, especially for the Coordination/Zero-sum polymatrix games family, adding between 1 and 7 new equilibria across every graph structure tested.

Table 3.6 Algorithmic Performance and Equilibria Counts using polymatrix games: Coordination/Zero-sum Games, Groupwise Zero-sum Games, Strictly Competitive Games, Weighted Cooperation Games, and the 6×6 special example (Table 3.2)

				LTP with 100 priors		BFS with 20 priors		
ID	**Game Type**	**Players**	**Graph**	**#Eq**	**Run Time**	**#Eq**	**Run Time**	**Diff.**
1	CoordZero	4	Complete	4	0.4s	11	0.7s	+7
2	GroupZero	5	Cycle	1	0.4s	1	0.2s	0
3	StrictComp	6	Grid	1	0.7s	1	0.3s	0
4	WeightCoop	4	Complete	1	0.5s	1	0.2s	0
5	CoordZero	4	Tree	3	0.7s	5	0.7s	+2
6	GroupZero	5	Complete	3	0.5s	5	0.4s	+2
7	StrictComp	2	Tree	1	5.2s	1	2.7s	0
8	WeightCoop	4	Cycle	1	0.4s	1	0.2s	0
9	CoordZero	4	Cycle	2	0.5s	3	0.3s	+1
10	GroupZero	5	Grid	1	0.6s	1	0.4s	0
11	StrictComp	2	Cycle	1	0.2s	1	0.1s	0
12	WeightCoop	4	Tree	1	0.4s	1	0.2s	0
13	CoordZero	4	Grid	3	0.5s	7	0.5s	+4
14	GroupZero	5	Tree	2	0.7s	3	0.3s	+1
15	StrictComp	3	Complete	2	0.6s	3	0.3s	+1
16	WeightCoop	4	Grid	1	0.2s	1	0.1s	0
17	6×6 game	2		4	0.4s	75	1.7s	+71

The runtime data in Table 3.6 further underscores the practical case for the BFS approach. The data for the Linear Tracing procedure was produced with 100 random priors, five times as many as the 20 used by the BFS, yet the BFS matches or exceeds it in equilibrium count on every game while running in equal or shorter time in all but

one case. The overhead of the backwards exploration, i..e., the construction the basis inverse, the injection the new covering vector is shown to be negligible relative to the cost of running 80 additional random traces. The most striking evidence, however, still comes from the 6×6 special example. Here, 100 random priors yield only 4 equilibria in 0.4 seconds, while the BFS Tracing Backwards procedure seeded with just 20 initial random priors discovers all 75 in 1.7 seconds. This is not a marginal improvement: it is a qualitative difference in what the two approaches are capable of finding. The standard LTP, however many random starts are attempted, cannot reach the high-support equilibria discovered only in the deep stages of the BFS tree, because their basins of attraction under the linear homotopy are too small to be hit by chance. The overarching conclusion is that tracing backwards exploration from a small number of equilibria re-utilised as structured starting points is strictly more efficient than exhaustive random sampling, both in the equilibria it finds and the time it takes to find them.

Chapter 4

A Comparison of Fair Division Algorithms with Money

4.1 Introduction

The problem of fair division involves distributing a set of objects among several involved parties in a way that ensures each recipient feels they have received their fair share of the set, even though each party values different subsets and individual objects differently. The nature of the problem is determined by the divisibility and homogeneity of each object, as well as the nature of utilities of the participating agents. In this chapter, we work with a version of fair division proposed by Bogomolnaia and Moulin (2025) that has in addition to indivisible goods, the possibility of cash compensation. This option for money transfer is part of the mechanism, allowing an agent that receives a nicer share of the bundle to compensate the other agents by paying them with some money. The main assumption in this setting is that the utilities are compensable as cash, i.e., the utilities are quasilinear. The introduction of money into the mechanism offers several benefits: cash transfer among agents smooths out the effects of the indivisibility of goods, and it is realistic, having applications like the classic rent-sharing among roommates and allocating indivisible bads within a community.

Crucial to evaluating a fair-allocation mechanism that allows cash compensation is the behavioral assumption of *safe play* (also known as a maximin strategy, formal definition in 4.2.4). Informally, playing safely means an agent acts completely conservatively to guarantee themselves a baseline level of fairness, entirely independent of

what the other participants choose to do. Instead of trying to predict or outsmart their opponents' strategies (which is practically impossible under strict information constraints), a safe player assumes a worst-case scenario. They choose an action or submit a bid that guarantees them a mathematically secure minimum utility threshold, ensuring they can never be exploited or left empty-handed, regardless of how unpredictably the other players behave in the economy.

A simple case of the setting of indivisible goods and quasi-linearity is where all players have additive utilities and the easiest solution would be to allocate each object in the set to the agent who values it the most (the *Bundled Auction* mechanism). The safe play for each player is to bid their true value; in this scenario, safe play ensures envy-freeness, makes sure that the utility each player receives does not exceed their utility from the entire set and maximises total utility. As the utilities of agents become more complex, it complicates how we measure fairness and how much information about utilities would a mechanism expect to be revealed by the agents. Consequently, achieving full efficiency becomes increasingly difficult. In their originating theoretical work, Bogomolnaia and Moulin (2025) propose two new division rules with efficient communication and privacy-preserving individual messages that hope to achieve a small loss in efficiency. A *Bid & Sell* mechanism (henceforth *B*&*S*) where each player reports a bid, a seller is chosen based on the bids and the remaining players purchase the optimal subset of goods at prices set by the seller. The second mechanism is an adaptation of *Divide & Choose* method (henceforth *D*&*C*) to include money. The main theoretical results from the paper posits that when every participant individually chooses safe play, the *Bid & Sell* mechanism will systematically outperform not only the brute-force *Bundled Auction*, but also the classic *Divide & Choose* mechanism by capturing a strictly higher share of the efficient social surplus due to its multidimensional messaging format.

This result provides the direct motivation for this chapter. Using extensive randomised experiments, we compare the performances of the mechanisms *B*&*S*, *D*&*C* presented in Bogomolnaia and Moulin (2025) against the benchmark *Bundled Auction* in terms of their efficiency when agents have monotonic and quasi-linear utilities over goods and money. We seek to check, if all participants play safely, whether or not the outcome in *B*&*S* collects on average a larger share of the efficient surplus than in *D*&*C*.

The overview of the rest of the paper is as follows: Section 4.3 discusses the relevant literature and situates this research. Section 4.4 introduces the fair division mechanisms, including the index used to measure fairness and efficiency. While

these opening sections establish the theoretical foundations derived from the work of Bogomolnaia and Moulin (2025), the remainder of the study transitions to our original empirical investigation and comparative numerical analysis. Section 4.5 covers the implementation choices and main results from numerical experiments using additive utilities. Section 4.6 discusses the methods and results for subadditive/superadditive utility configurations. Section 4.7 discusses the running time of different algorithms and their computational complexities. The programs and pseudocode are provided in Appendix B.

4.2 Definitions

This section introduces the key definitions and notation used throughout this chapter. We use these concepts to formally analyse and compare the three fair division mechanisms under investigation.

4.2.1 Utilities and Valuations

Definition 4.1 (Quasilinear Utility Function)**.** An agent i has a *quasilinear utility function* if her utility for a bundle of goods $S \subseteq A$ and a cash transfer $t_i \in \mathbb{R}$ can be expressed as:

$$u_i(S, t_i) = v_i(S) + t_i$$

where $v_i : 2^A \to \mathbb{R}_{\geq 0}$ is the agent's valuation of the goods bundle, and utilities are fully compensable through monetary transfers.

Definition 4.2 (Monotonic Utility)**.** An agent's utility function u_i is *monotonic* if for any two bundles $S, S' \subseteq A$ with $S \subseteq S'$, we have $u_i(S) \leq u_i(S')$. That is, receiving more goods does not decrease utility.

Definition 4.3 (Additive Utility)**.** An agent's valuation v_i over bundles is *additive* if for every bundle $S \subseteq A$:

$$v_i(S) = \sum_{j \in S} v_i(\{j\})$$

That is, the agent's value for a bundle is the sum of values for individual items, with no interaction effects or synergies between goods.

Definition 4.4 (Subadditive and Superadditive Utility)**.** An agent's valuation v_i is:

- *subadditive* if for all $S, T \subseteq A$ with $S \cap T = \emptyset$, we have $v_i(S \cup T) \leq v_i(S) + v_i(T)$. Intuitively, the agent experiences diminishing returns: adding items to a bundle provides less marginal value than separately valuing those items.
- *superadditive* if for all $S, T \subseteq A$ with $S \cap T = \emptyset$, we have $v_i(S \cup T) \geq v_i(S) + v_i(T)$. The agent experiences complementarities: items are more valuable together than separately.

4.2.2 Allocations and Partitions

Definition 4.5 (Allocation)**.** An *allocation* is a partition of the set of goods A among n agents such that each good is assigned to exactly one agent. Formally, an allocation is a collection of disjoint bundles $(S_1, S_2, \ldots, S_n)$ where $\bigcup_{i=1}^{n} S_i = A$ and $S_i \cap S_j = \emptyset$ for all $i \neq j$.

Definition 4.6 (Partition (Divider's Perspective))**.** In the context of *Divide & Choose*, a *partition* π is a division of the goods A into n disjoint bundles $(S_1, S_2, \ldots, S_n)$ proposed by the divider. The remaining agents then choose from these bundles according to the mechanism's rules.

4.2.3 Performance and Efficiency

Definition 4.7 (Maximal Surplus)**.** An allocation is *efficient* if there exists no alternative allocation that yields a strictly higher utility for at least one agent without reducing the utility of another. For a given utility profile $\vec{u} \in (\mathcal{M}^+)^n$, the *maximal efficient surplus* $\mathcal{W}(\vec{u})$ represents the first-best, unconstrained total welfare achievable in the economy, formally defined as:

$$\mathcal{W}(\vec{u}) = \max_{\pi \in \mathcal{P}(A;n)} \sum_{i=1}^{n} u_i(\pi_i)$$

where $\mathcal{P}(A; n)$ denotes the set of all valid n-partitions of the asset set A across the n agents.

Definition 4.8 (Percentage of Maximal Surplus Captured)**.** To evaluate and compare the performance of different fair division mechanisms under varying information structures, we introduce the *Percentage of Maximal Surplus Captured*, denoted by $\mathcal{V}(M, \vec{u})$. For a mechanism M operating on a utility profile $\vec{u}$, the metric is defined

as:

$$\mathcal{V}(M, \vec{u}) = \frac{\sum_{i=1}^{n} u_i(M_i(\vec{u}))}{\text{Maximal Surplus } \mathcal{W}(\vec{u})}$$

where $M_i(\vec{u})$ denotes the bundle allocated to agent i under mechanism M.

This ratio is bounded within $[0, 1]$, where a value of 1 signifies full economic efficiency. Throughout this chapter, $\mathcal{V}(M, \vec{u})$ serves as a comparison metric that quantifies a mechanism's capacity to capture the maximal efficient surplus available while satisfying its structural fairness criteria. In algorithmic game theory and computer science, the worst-case lower bound of this ratio corresponds to the *approximation ratio*, while its inverse is conceptualised as the *price of fairness* (Bertsimas, Farias, and Trichakis, 2011; Caragiannis, Kaklamanis, Kanellopoulos, and Kyropoulou, 2009).

Definition 4.9 (Information Elicitation)**.** *Information elicitation* in fair division refers to the problem of extracting sufficient information about agents' preferences to compute a fair and efficient allocation. When agents have general (e.g., combinatorial) valuations over bundles, fully specifying utilities requires 2^m parameters for m items, which is cognitively infeasible.

An allocation mechanism M is defined as *informationally economical* if it restricts individual communication to low-dimensional message actions while preserving baseline fairness and efficiency guarantees. Mechanisms designed for practical use must limit the information requested (e.g., single bids in $B\&S$ or partition proposals in $D\&C$) to be informationally economical.

4.2.4 Strategic Behaviour and Guarantees

Definition 4.10 (Safe Strategy)**.** A safe strategy for agent i is one that maximises her worst-case utility in a given model when she has no information about the preferences or strategic choices of other agents. More formally, if σ_i denotes agent i's strategy and $u_i(\sigma_i, \sigma_{-i})$ denotes the utility under strategy profile (σ_i, σ_{-i}), then a strategy σ_i^* is a safe strategy if it satisfies:

$$\min_{\sigma_{-i}} u_i(\sigma_i^*, \sigma_{-i}) \geq \min_{\sigma_{-i}} u_i(\sigma_i, \sigma_{-i})$$

for all alternative strategies σ_i.

Definition 4.11 (Guarantee and Maximal Guarantee)**.** A *Guarantee* in an n-person problem is a mapping Γ_i that assigns each agent i's utility function $u_i \in \mathcal{M}^+$ to a

guaranteed baseline utility level $\Gamma_i(u_i) \in \mathbb{R}_+$ satisfying the following inequality:

$$\sum_{i=1}^{n} \Gamma_i(u_i) \leq \mathcal{W}(\vec{u})$$

for all utility profiles $\vec{u} = (u_1, \ldots, u_n) \in (\mathcal{M}^+)^n$ and $\mathcal{W}(\vec{u})$ as in Definition 4.7.

The guarantee represents the utility an agent is assured to receive under safe play, independent of others' strategies, subject only to knowing her own utility function. This guarantee is *maximal* if we cannot raise its value at any utility u_i without violating the above inequality at some profile of utilities for the agents other than i.

Definition 4.12 (Proportional Share)**.** The *Proportional Share* guarantee Γ_n^{PS} awards each agent i a fraction $1/n$ of their utility for the entire bundle:

$$\Gamma_n^{PS}(u_i) = \frac{1}{n} u_i(A)$$

This is the standard, though coarse, interpretation of ex-ante fairness when agents have additive utilities. It is insufficient as a fairness measure for non-additive utilities (Bogomolnaia and Moulin, 2025).

4.2.5 Agent Typologies

Definition 4.13 (Frugal Agent)**.** A *Frugal* agent is one whose utility function exhibits subadditive structure: each item contributes diminishing value to the bundle. Formally, for small item sets or individual items, the marginal value per item is high, but adding more items yields progressively lower marginal utility. Intuitively, a frugal agent is easy to satisfy: even a small bundle can yield high utility relative to the full set.

In the absolute subadditive limit, a *Frugal* agent is fully satisfied by acquiring any single object from the set of identical goods A, expressed as:

$$u_F(S) = 1 \quad \forall\, \emptyset \subset S \subseteq A, \quad \text{and} \quad u_F(\emptyset) = 0$$

Definition 4.14 (Greedy Agent)**.** A *Greedy* agent is one whose utility function exhibits superadditive or complementarity structure: items are more valuable together than separately. The agent requires many items in combination to achieve substantial utility. Intuitively, a greedy agent is hard to satisfy: many items are needed together to provide reasonable utility.

In the absolute superadditive limit, a *Greedy* agent operates on an all-or-nothing requirement, deriving utility exclusively from the full bundle A, expressed as:

$$u_G(A) = 1, \quad \text{and} \quad u_G(S) = 0 \quad \forall\, S \subsetneq A$$

4.2.6 Fairness Axioms

Definition 4.15 (Positivity)**.** A guarantee Γ is *positive* if, for any agent whose utility for the full bundle is strictly positive ($u_i(A) > 0$), the guarantee is also strictly positive: $\Gamma(u_i) > 0$. Positivity ensures that no agent is entirely excluded from the benefit of common property due to their utility structure.

Definition 4.16 (Responsiveness)**.** A guarantee Γ is *responsive* if it distinguishes between agents based on how difficult they are to satisfy. Specifically:

- An agent with *Frugal* utility as in Definition 4.13 (subadditive, no strong complementarities) should receive strictly more than the Proportional Share: $\Gamma(u_{\text{Frugal}}) > \frac{1}{n}$.
- An agent with *Greedy* utility as in Definition 4.14 (superadditive, strong complementarities) should receive strictly less than the Proportional Share: $\Gamma(u_{\text{Greedy}}) < \frac{1}{n}$.

Responsiveness is motivated by a normative principle: mechanisms should reward agents with utility structures that are easier to satisfy and implicitly penalise those requiring expensive complementarities. It should be noted that while Positivity is a strong notion, the fairly minimal definition of Responsiveness serves primarily to highlight the sharp contrast between the extreme *Frugal* and *Greedy* utility profiles and the mechanism's reactiveness to such extreme points.

Definition 4.17 (Envy-Freeness)**.** An allocation $(\pi, t) \in \mathcal{P}(A; n) \times T(n)$ is *envy-free* if no agent prefers another agent's bundle-plus-transfer to her own. Formally, for all agents $i, j \in n$:

$$u_i(S_i) + t_i \geq u_i(S_j) + t_j$$

Envy-freeness is an ex-post fairness concept, evaluated after allocation and comparing allocations between specific agents. It requires interpersonal utility comparisons and is more stringent than guarantee-based fairness, which is based solely on maximising worst-case utility without such comparisons.

Remark: This chapter focuses on guarantee-based fairness rather than envy-freeness, as guarantees require only cardinal utility maximisation and no comparisons of allocations. Envy-freeness is relational and ex-post: after allocation, it asks "does agent i prefer agent j's bundle?" This requires comparing utilities across agents and knowing the full allocation before you can verify fairness. Guarantees are individual and ex-ante: Before allocation (even before knowing what others receive), agent i asks "what's the worst-case utility I can secure?" This is purely about maximising her own worst case, with no interpersonal comparisons needed.

4.3 Literature Review and Research Context

The problem of fair division, particularly the allocation of indivisible objects, has historically been a central theme in mathematics, economics, and computer science. The introduction of cash compensation, often referred to as quasi-linear utilities, simplifies the challenge of indivisibility but introduces new questions about mechanism design and optimal guarantees.

4.3.1 Fair Division with Money (Quasi-Linear Utilities)

Early work on fair division largely ignored the possibility of cash compensation. However, the economic discussion of the *assignment problem* (where agents receive at most one object) established a key precedent, demonstrating that monetary compensation can restore concepts like *Envy Freeness* and competitive equilibrium (Bogomolnaia and Moulin, 2025; Tadenuma and Thomson, 1993). This led to practical applications which routinely uses price-based reporting to allocate goods like rent shares, inheritance splits etc.

The challenge addressed by Bogomolnaia and Moulin (2025) and continued in this work stems from moving beyond the simple additive and assignment problems to scenarios involving arbitrarily complex externalities over bundles of goods. In this setting, an agent's utility requires a complex vector of dimension $2^{|A|} - 1$ to fully describe, which is cognitively infeasible to report in a real-world setting. This impossibility necessitates the design of mechanisms based on simpler, information-economical messages.

4.3.2 Fairness Guarantees in Indivisible Division

In models without cash transfers, defining a suitable fairness criterion is difficult due to un-smoothable indivisibilities. Consequently, the literature approaches these strict criteria by focusing on threshold-based guarantees that secure an agent's position against worst-case outcomes. The standard benchmark used to evaluate fair division under indivisible goods is the *MaxMinShare* (MMS) (Kurokawa, Procaccia, and Wang, 2018; Kurokawa, Procaccia, and Shah, 2018), which serves as a personalised baseline guarantee for each participant. Formally, for an agent i with utility function u_i, their MaxMinShare threshold MMS_i is defined as:

$$\text{MMS}_i = \max_{\pi \in \mathcal{P}(A;n)} \min_{k \in 1,\ldots,n} u_i(S_k^\pi)$$

where $\mathcal{P}(A;n)$ denotes the set of all valid n-partitions of the complete set A, and S_k^π represents the k-th individual bundle within a specific partition π. Conceptually, this assumes a worst-case scenario where agent i partitions the items into n subjective bundles, but is forced to choose their share last, receiving the bundle they value least. The *MaxMinShare* and and the related *MinMaxShare* (Bouveret and Lemaître, 2016) concepts seek to define a floor guarantee for agents.

In the quasi-linear setting used in this chapter, the Maxmin utility (the counterpart to MMS) serves as an upper bound on all reasonable guarantees, while the minMax utility serves as a lower bound on reasonable guarantees. This framework allows for a formal normative critique of the simple *Familiar Share*, $\frac{1}{n}u(A)$, arguing that it is "repugnant" because it fails to appropriately reward agents with subadditive utilities (like the *Frugal* agent in 4.2.5) or penalise those with superadditive utilities (like the *Greedy* agent in 4.2.5) that characterise complex externality issues. Guarantees should be both *Positive* and *Responsive* (see Definitions 4.15 and 4.16).

4.3.3 Research Contribution

This research directly contributes to the literature initiated by Bogomolnaia and Moulin (2025). We conduct a numerical comparison of the efficiency of the key proposed mechanisms – *Bid & Sell (B&S)*, *Divide & Choose (D&C)*, and the benchmark *Bundled Auction (BA)* – under independent safe play by participants. The core hypothesis originating from the theoretical framework of Bogomolnaia and Moulin (2025) posits that the *B&S* mechanism achieves a strictly higher share of efficient surplus than the *D&C* mechanism under independent safe play. The results of

this chapter confirm this foundational hypothesis across utility domains under structured information constraints. However, a notable divergence occurs when agent preferences are strictly additive: when valuation functions are completely independent and free of both subadditive and superadditive interactions, the performance of *B*&*S* fluctuates relative to *D*&*C* depending on the number of agents and the objects to be divided among them. This early finding underscores that while *B*&*S* provides superior strategic leverage globally, its comparative advantage is fundamentally bounded by the presence of non-additive preference structures in agent valuations. Specifically, this chapter delivers four primary contributions:

1. Computational validation of safe play: We provide computational evidence that independent safe play in the *B*&*S* mechanism consistently collects a larger average share of the efficient surplus than in *D*&*C*, especially as object complexity or quantity increases, thereby confirming the core theoretical hypothesis.

2. Delineation of informational limits: We demonstrate robustness of *Bid & Sell (B*&*S)* and the informational limitations of *D*&*C*, showing that the single-value bid in *D*&*C* is insufficient to achieve high efficiency, particularly when utilities are non-additive or heterogeneous.

3. Testing with heterogeneous populations: We test how these mechanisms perform when players possess a mix of utility families (additive, subadditive, and superadditive), a crucial test of resilience in real-world division problems. We demonstrate that even with quasilinear utilities, the structural *slack* created by a well-chosen partition (in *D*&*C*) or a strategic price vector (in *B*&*S*) remains a critical driver of total social welfare.

4. Discovery of the additive anomaly: We uncover a surprising insight in Section 4.5.7 regarding strictly additive preferences, showing that while the full transferability of utility via cash transfers is intended to smooth over the indivisibility of goods, it does not eliminate the collective structural benefit of choosing an allocation format that balances those transfers.

4.4 The Fair Division Mechanisms

To evaluate the trade-offs between efficiency, structural fairness and information elicitation restrictions, we analyse three distinct mechanisms. Let A denote the complete set of indivisible objects, n represent the total number of participating

agents, and u_i define the quasilinear utility function for each player $i \in N$, with an assumption that for a subset S: $\phi \subseteq S \subseteq A$, $u_i(\emptyset) = 0 \leq u_i(S)$ for all $S \subseteq A$.

We now formally define the algorithmic steps and safe strategies associated with each respective mechanism environment.

4.4.1 Bundled Auction

The Proportional Share (see definition 4.12) is implemented by the *Bundled Auction* (*BA*) mechanism, where agents bid truthfully for the entire set. *BA* implements the following algorithm: each agent i submits a non-negative bid β_i that represents the agent's utility for the entire set A. The highest bidder i^* then gets A and pays $\frac{1}{n}\beta_{i^*}$ to each of the $n-1$ other agents (pseudocode presented in Appendix B). The only safe bid (a bid that maximises worst case utility, with no clue about the opponents' behaviour) in *BA* is to bid truthfully, $\beta_i = u_i(A)$.

BA's Information Elicitation From an informational perspective, *BA* represents an extreme compression of the preference space. It completely bypasses the exponential $2^m - 1$ combinatorial elicitation barrier by aggregating an agent's entire utility matrix into a single, one-dimensional scalar representing their valuation for the grand bundle A. While this drastically minimises the communication burden on the agents, this severe informational restriction renders the mechanism entirely non-responsive to individual item preferences, serving as a coarse efficiency benchmark for our comparative analysis.

4.4.2 Bid & Sell

The definition of $B\&S$ algorithm for two players ($B\&S_2$):

1. the agents bid to become the Seller: the one with the lowest bid x wins.
2. The Seller chooses a price $p \in \mathbb{R}_+^A$ from the simplex of prices $\Delta(x)$ such that $p(A) = x$ (the whole bundle costs x).
3. The Buyer buys any share S of A (possibly ϕ) at the price $p(S)$ and the Seller receives $A \setminus S$ along with $p(S)$.

The n-player version has a recursive definition (refer to Bogomolnaia and Moulin (2025) for the recursive definition). The $B\&S$ mechanism systematically reduces how much of participants' preferences are revealed - from the exponential combinatorial space to a linear vector space of single bids. Rather than forcing participants to disclose complex preference values for every potential package combination, $B\&S$ shows nuanced information elicitation where all participants are only required to submit a single, one-dimensional price bid for each individual object. And when given the role of a Seller, each agent's informational task is reduced to establishing a single, linear vector of itemised prices.

4.4.3 Divide & Choose

Before going into the definition of our $D\&C$ algorithms, it is necessary to introduce a sub-step involved called the π-*auction*.

π-auction The π-auction implements the π-guarantee:

$$\Gamma_n^{\pi}(u) = \frac{1}{n}u(\pi) = \frac{1}{n}\sum_{k=1}^{n} u(S_k)$$

for all u. Given a partition π, each agent reports a vector of money transfer over each share. The winning assignment of shares to agents is the one that minimises the total money transferred. After the assignment of shares and money, the remaining surplus is equally divided between the agents. The safe play in π- auction is to report the transfers equalising one's utility across the shares of π. When all players play safe, the shares from π-auction are assigned efficiently.

Understanding Balanced Transfers In a π-auction for n agents, agents use money to equalise the subjective value of different shares in a partition $\pi = \{S_1, S_2, S_3, ..., S_n\}$. Each agent i reports a vector of balanced transfers $(t_i^{S_1}, t_i^{S_2}, t_i^{S_3}, ..., t_i^{S_n})$ on the n shares such that they sum to zero ($\sum_{k=1}^{n} t_i^{S_k} = 0$).

- Negative Transfer ($t < 0$): This is a tax. It indicates that the agent finds the share highly desirable and is willing to pay to receive it.
- Positive Transfer ($t > 0$): This is a subsidy. It indicates that the agent finds the share less desirable and needs to be paid to accept it.

The mechanism finds an assignment that minimises the sum of these transfers, creating a "negative slack" (surplus) that is rebated to all players. An illustrative example to demonstrate π- auction is given in Appendix B.

***D&C* procedure** The first version of $D\&C$ algorithm has the following steps, $D\&C_n^1$:

1. simple auction to choose the Divider. Agents bid β_i, winner becomes the Divider.
2. Divider pays $\frac{1}{n}\beta_i$ to all other players, then picks a partition π in $\mathcal{P}(A; n)$
3. run the π-auction

In the second version, the first step is eliminated, and each agent proposes a partition π_i. An *averaging auction* is implemented on this set of π-guarantees. Each agent reports a vector of money transfers over these guarantees. The auction selects a π-guarantee that minimises total money transfer. Each agent i receives $t_r^i - \frac{\sum_{j\in N} t_r^j}{n}$ where r is the winning guarantee. $D\&C_n^2$:

1. each agent i picks a π_i in $\mathcal{P}(A; n)$ and are revealed.
2. run the averaging auction over the profile of partitions π_i and select a partition that minimises total money transfer.
3. the π-auction is implemented on the selected partition.

Safe bid in $D\&C_1$ is to bid $\beta_i = Maxmin_n(u_i) - minMax_n(u_i)$ i.e., the difference between the utility obtained from the most favourable partition's worst share (as a Divider) minus the utility obtained from the least favourable partition's best share (as a Chooser). And in $D\&C_2$, player i chooses a maxi-minimising partition. In both cases the guaranteed utility is $\frac{1}{n}$ Maxmin + $\frac{n-1}{n}$ Minmax.

The $D\&C$ mechanism manages information elicitation by firstly concentrating the communication burden onto a single participant. Instead of demanding a multi-dimensional preference report from all players, the rule requires only the designated divider to compress their private valuation space into a low-dimensional action: dividing the objects and cash requirements into balanced, fair bundles. For the remaining agents, the elicitation process is reduced to a simple sequential evaluation and selection over a small, discrete set of options, making it informationally more economical for the choosing population.

4.4.4 Illustrative Example: *Frugal* vs. *Greedy*

To understand how *B&S* and *D&C* are designed to improve upon *BA* by providing guarantees that respond to these different utility structures, we consider two extreme agents, *Frugal* and *Greedy*, sharing $m \geq 2$ identical goods. As defined in 4.2.5, we assume the agents represent polar opposite utility boundaries:

- *Frugal* (u_F): Maximally subadditive utility where any single object provides full satisfaction:
$$u_F(S) = 1 \quad \forall\, \emptyset \subset S \subseteq A, \quad \text{and} \quad u_F(\emptyset) = 0$$
- *Greedy* (u_G): Maximally superadditive utility requiring an all-or-nothing allocation:
$$u_G(A) = 1, \quad \text{and} \quad u_G(S) = 0 \quad \forall\, S \subsetneq A$$

A standard Proportional Share would simply guarantee each agent a utility of $\frac{1}{2}$. However, the newer mechanisms provide a more nuanced fair outcome by recognising that *Frugal* is easier to satisfy than *Greedy*.

The *Bundled Auction* Outcome Each agent submits a single bid β_i for the whole bundle; the highest bidder wins all objects and compensates the other agent with a fraction of their bid. *Frugal*'s Safe Play: *Frugal* values any non-empty set at 1, including the full bundle A. The only safe bid in *BA* is to bid truthfully, $\beta_F = u_F(A) = 1$. If she wins, she gets the goods (value 1) and pays $\frac{1}{n}$ of her bid to *Greedy*. Her guaranteed utility is the Proportional Share (PS): $\Gamma_n^{PS}(u_F) = \frac{1}{2}$. *Greedy*'s Safe Play: *Greedy* only values the full bundle A at 1. Following the safe strategy of truthful bidding, he bids $\beta_G = u_G(A) = 1$. Like *Frugal*, his guaranteed utility is also exactly the Proportional Share (PS): $\Gamma_n^{PS}(u_G) = \frac{1}{2}$. *BA* is Positive (every agent gets a non-zero share if they value the goods) but it is not Responsive. It fails to differentiate between the agents, offering the same $\frac{1}{2}$ guarantee to *Frugal* (who is 'easy to satisfy') and *Greedy* (who is 'hard to satisfy').

The *Divide & Choose* Outcome : In the *D&C* rule, agents bid for the role of Divider. To play safely, they must calculate a bid that balances their worst-case utility as a Divider versus a Chooser. *Greedy*'s Safe Play:

- **As Divider**: If *Greedy* wins with bid x (which he pays to the other), he offers a choice between all the goods (with a tax) or no goods (with a subsidy): $(A, -\frac{1}{2})$ or $(\emptyset, \frac{1}{2})$. This ensures his net utility is $\frac{1}{2} - x$ regardless of the Chooser's pick.
- **As Chooser**: If *Greedy* loses, he receives x. In the worst case, he faces a partition where both shares are non-empty, yielding him zero object utility. His net utility is simply x.
- **Safe Bid**: Solving $\frac{1}{2} - x = x$ yields $x^* = \frac{1}{4}$. *Greedy*'s $D\&C$ guarantee is $\frac{1}{4}$.

Frugal's Safe Play:

- **As Divider**: She offers two non-empty shares $(S, 0)$ and $(A \setminus S, 0)$, securing a utility of 1 and a net utility of $1 - x$.
- **As Chooser**: If she loses and becomes the Chooser, she receives x. Facing any choice (S, t) and $(A \setminus S, -t)$, she can always pick a share worth at least $\frac{1}{2}$. Her net utility is $\frac{1}{2} + x$.
- **Safe Bid**: Solving $1 - x = \frac{1}{2} + x$ yields $x^* = \frac{1}{4}$. *Frugal*'s $D\&C$ guarantee is $\frac{3}{4}$.

Frugal can guarantee a utility of $\frac{3}{4}$ because she only needs one object or a small cash transfer to be happy. The result is that $D\&C$ rewards *Frugal*'s subadditive preferences by guaranteeing her three times more utility ($\frac{3}{4}$) than *Greedy* ($\frac{1}{4}$). $D\&C$ is Responsive, but its guarantee is fixed at $\frac{3}{4}$ and $\frac{1}{4}$ regardless of the number of objects m.

The *Bid & Sell* Outcome : The $B\&S$ mechanism creates an even sharper contrast as the number of goods m increases. Agents bid to become the Seller who sets the prices. *Frugal*'s Safe Play:

- **As Seller**: With bid x, she posts a uniform price $\frac{1}{m} \cdot x$ for each good. She gets at least $\min\{x, 1\}$.
- **As Buyer**: If she loses to a smaller bid, the Seller must offer at least one good at a price $\leq \frac{1}{m} \cdot x$. *Frugal* buys one good for a net utility of $1 - \frac{1}{m} \cdot x$.
- **Safe Bid**: $x^* = \frac{m}{m+1}$. *Frugal*'s $B\&S$ guarantee is $\frac{m}{m+1}$.

Greedy's Safe Play:

- **As Seller**: He posts a uniform price $\frac{1}{m} \cdot y$, guaranteeing $\min\{\frac{1}{m} \cdot y, 1\}$.
- **As Buyer**: He must pay up to y to buy all goods, guaranteeing $1 - y$.
- **Safe Bid**: $y = \frac{m}{m+1}$. *Greedy*'s *B&S* guarantee is $\frac{1}{m+1}$.

While Safe Bidding, both agents determine that the safe bid is $x = \frac{m}{m+1}$. *Frugal* secures a high guarantee of $\frac{m}{m+1}$. *Greedy* is penalised for his *all-or-nothing* requirement, securing only $\frac{1}{m+1}$.

While *D&C* recognises the difference in utility types, *B&S* is more responsive. As the set of goods m grows, it provides the most sophisticated correction to the Proportional Share by utilising the pricing stage to reveal more information about the agents' preferences and collects a much larger share of the surplus for the *Frugal* agent, whereas the *D&C* guarantee remains relatively coarse.

The following table summarises the guarantees $\Gamma_n(u)$ for the two agents across the three mechanisms discussed (for $n = 2$[1]):

Table 4.1 Comparison Table: Individual Guarantees

Mechanism	**Frugal (u_F)**	**Greedy (u_G)**	**Positive?**	**Responsive?**
Bundled Auction (BA)	1/2	1/2	Yes	No
Divide & Choose (D&C)	3/4	1/4	Yes	Yes
Bid & Sell (B&S)	$\frac{m}{m+1}$	$\frac{1}{m+1}$	Yes	Yes (More responsive than D&C)

4.5 Comparison Under Additive Utilities

Having established the theoretical framework of fairness, positivity, and responsiveness proposed by Bogomolnaia and Moulin (2025), we now transition from these formal definitions to our own computational analysis, where we empirically evaluate how effectively these mechanisms capture social welfare through randomised simulations under varying utility configurations.

While the extreme case of *Frugal* & *Greedy* highlight the stark behaviour differences of the *B&S*, *D&C* and *BA* mechanisms, the following simulations test these mechanisms under more normal, less extreme utility configurations to evaluate how much

[1]Refer to Appendix B for an illustrative example with $n > 2$.

of the efficient surplus they capture on average when players follow safe strategies. The simulated random agents have monotonic and quasi-linear utilities over goods and money. We seek to check if all participants play safely, whether the outcome in $B\&S$ collects on average a larger share of the efficient surplus than in $D\&C$ and where BA stands compared to these two mechanisms. The programs and pseudo-code are attached in Appendix B.

4.5.1 Generating Players with Additive Utilities

In our numerical experiments to compare the efficiency of fair division rules, we simulate randomised players by generating randomised utilities for each player at each instance. In the additive case, we use a uniform random distribution $U(1, 100)$ and draw m random values as utilities and repeat it for n players. In our tests, the range of m lies between 2 to 100 objects. The range of n is comparatively inflexible due to the π- auction executions, which involves evaluating all permutations of share allocation. As n grows, the number of possibilities to assign shares to players grows factorially (for $n = 10$, 10! permutations = 3628800). Due to the computational complexity of the π-auction, we limit n to a maximum of 7 (since 7! = 5040 which is numerically tractable) in our numerical experiments. See Section 4.7 for a full complexity analysis.

4.5.2 Additive: Maximal Surplus

The maximal surplus available in the economy serves as the absolute efficiency benchmark against which the performance of each allocation mechanism is evaluated. When agent preferences are strictly additive, the optimisation problem simplifies significantly. Instead of evaluating all the valid n-partitions, the maximal efficient surplus $\mathcal{W}(\vec{u})$ can be computed independently for each item. Formally:

$$\mathcal{W}(\vec{u}) = \sum_{a \in A} \max_{i \in N} u_i(\{a\})$$

In this setting, the aggregate surplus $\mathcal{W}(\vec{u})$ is simply the summation of maximum valuations across the set A, attained by directly allocating each individual object to the specific agent who values it the most.

As defined previously, the total utility realised under any given mechanism M is normalised relative to this this maximal surplus. This ratio yields the percentage of

efficient surplus captured by the mechanism, denoted by $\mathcal{V}(M, \vec{u})$, which is used to compare performance across mechanisms.

4.5.3 Additive: Bundled Auction

When a player has an additive utility function, the only safe bid is to bid truthfully: $u_i(A)$. The total surplus after the winner pays other agents is equal to the winning bid.

4.5.4 Additive: Bid & Sell

When utilities are additive, safe bid in $B\&S$ is to play $\frac{1}{n}u(A)$. This bid is considered safe because it represents the intersection of an agent's worst-case utility as a Seller and a Buyer. If agent bids x^* and becomes a Seller, they must set a price vector such that $p(A) = x^*$. By setting a uniform price for each object a as $p_a = \frac{1}{n}u_i(a)$, the Seller ensures that if the Buyer purchases an item, the Seller is compensated at exactly the rate of their own valuation. If the Buyer purchases all items, the Seller receives their bid $x^* = \frac{1}{n}u_i(A)$. If the Buyer purchases nothing, the Seller retains the full bundle worth $u_i(A)$. For additive utilities, this strategy guarantees a worst-case utility of $\frac{1}{n}u_i(A)$. If chosen as Buyer, then Buyer buys every object a that satisfies $u_{buyer}(a) > \frac{1}{n}u_{seller}(A)$. By purchasing every such object, the Buyer ensures their net utility (value of goods minus prices paid) is at least $\frac{1}{n}u_i(A)$. Because the worst-case utility in both roles is equalised at $\frac{1}{n}u_i(A)$, this value represents the safe bid that maximises the agent's guarantee.

4.5.5 Additive: Divide & Choose

In the additive case for $D\&C$, safe play is significantly simplified because the MinMax and MaxMin utilities both converge to the Proportional Share, $\frac{1}{n}u(A)$. Consequently, the safe bid in the initial auction is 0 (the difference between Minmax and Maxmin is 0), as there is no utility gap between the roles of Divider and Chooser; any partition π is safe. For this reason, the Divider is picked at random in our simulations.

The core of the $D\&C$ mechanism is the π-auction that follows after the Divider proposes a partition, which relies on agents submitting balanced transfers to equalise the subjective value of shares. As illustrated in the π-auction example (Appendix

B), these transfers act as taxes or subsidies - where negative values indicate a desire to pay for a share and positive values indicate a requirement for compensation - to ensure each agent receives their fair share.

For the Divider, choosing a partition the choice of a partition is equally arbitrary. However, following the balanced transfer logic established in Section 4.4.3, a *perfect* Divider aims to create a partition π where every share is exactly equal to $\frac{1}{n}u(A)$. The brute force choice would be to compute all possible n-partitions from a set and arrive at the partition whose shares provide an almost equal value for the player. But as the number of objects m grows, the permutations of objects into n shares also grows, proving computationally very expensive. To approximate this balance without the computational burden of a brute-force search, we implement a few partitioning algorithms that aim to minimise the magnitude of these cash transfers. The corresponding algorithms are of independent interest.

We introduce two ways of dividing the set of objects A, both of which start by arranging the objects in ascending order of utility for the player.

- *Sorted Partition or Box-Filling Partition:*

  1. Arrange m objects in the ascending order of utility to $player_i$, the Divider.
  2. Create n boxes S_i, one for each player's share.
  3. Add each object supplied in the sorted order into the first box S_1 until the total weight of utilities in the box $\sum u_i(S_1) = \frac{1}{n}u_i(A)$, at which point fill the next box.
  4. If this sum $\sum u_i(S_1)$ is close to $\frac{1}{n}u_i(A)$ but adding the next object will overflow the box beyond this limit, move to the next box.
  5. Repeat step 3 for the next box until all objects are placed in some box. If the last box is reached, add all remaining objects to it.

  The best-case scenario is where there are n bundles of objects in set A such that the utility of each bundle S_j is $u_i(S_j)_{j\in N} = \frac{1}{n}u_i(A)$. The Sorted Partition algorithm then creates shares that are exactly equal. In the worst case scenario, there is much variance between the smaller and the larger objects in terms of utilities, and the algorithm performs poorly, due to the indivisibility of objects, by giving away objects with higher utilities to the last pile.

- *Round-Robin partition:*

  1. Arrange m objects in the ascending order of utility to $player_i$, the Divider.

2. Create n boxes S_i, one for each player's share.
3. Add the first object in the sorted order to the first box, second object to the second box and so on.

One best case scenario for *Round-Robin partition* is when the objects have identical utilities and the number of objects m is a multiple of n or if the utilities of objects are balanced in a way that the sum of all $i + n$'th objects equal $\frac{1}{n}u_i(A)$. The worst case occurs when the objects with higher utilities are too high, worsened if there are not enough such objects for every box.
The total utility from running the rule is equal to the sum of each player's utility after receiving shares through the π-auction. Note that the vectors of money transfer over guarantees submitted by agents in the averaging auction (step 2 of $D\&C_n^2$) will all be zero vectors due to utilities being additive. Therefore, $D\&C_n^2$ reduces to $D\&C_n^1$ in the additive utility scenario.

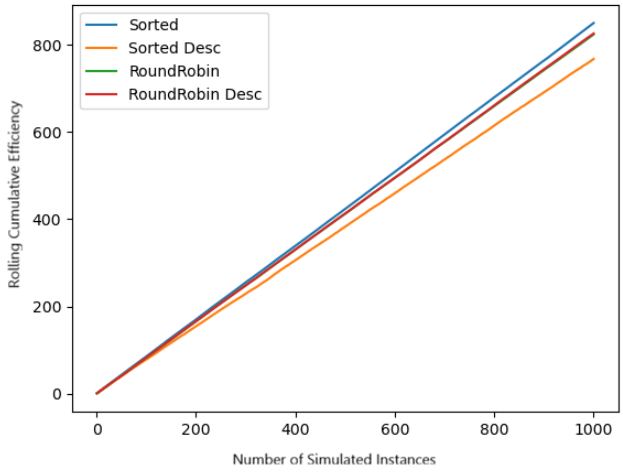
4.5.6 Optimisation of the Divider's Role in *D&C*:

The efficiency of the $D\&C$ mechanism is highly sensitive to the initial partitioning method used by the Divider. Unlike the coarse bundling found in *BA*, $D\&C$'s responsiveness is maximised when the partition successfully equalises shares.

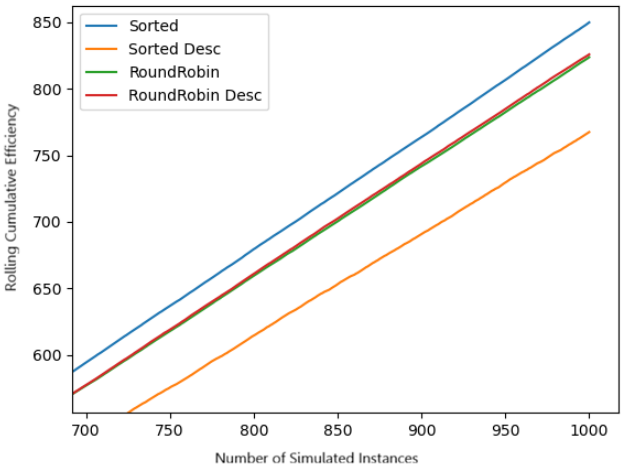
In our numerical experiments, we generated 1000 samples of uniform random utilities while fixing the number of players and objects at $n = 4$ and $m = 8$. We calculate the cumulative percentage of efficiency achieved by four algorithms: *Round-Robin partition* and *Sorted Partition*, in ascending and descending order of sorting. As seen in Figure 4.1, *Sorted Partition* when arranged in ascending order performs the best for a fixed number of players and objects. *Round-robin partition* falls behind but behaves almost the same way in both sorting directions.

Before we pick one, we tested the two methods for varying number of players and objects. Figure 4.2 showcases the normalised total value from $D\&C_n^1$ for $n \in \{3, 5, 7\}$ and $m \in [2, 30)$ averaged over a 100 iterations in each combination of players and objects. Under both dividing methods, the average total value grows as the number of players grow, but is more pronounced under *Round-Robin partition*. As n increases, the increase in $\mathcal{V}$ is weak with respect to the *Sorted partition*, but the graph traces a mild U shaped curve as m increases. The next graph shown in Figure 4.3 is the comparison between *Round-Robin partition* and *Sorted partition* arranged per object to show the effect of different sizes of set A on the same number of players n.

Numerical experiments show that the Sorted Partition (ascending order) consistently yields higher efficiency than Round-Robin for fixed numbers of players and objects. Efficiency grows as the number of players (n) increases, with the Round-Robin method showing more pronounced gains at higher n counts. Observations indicate that Round-Robin performs better when $m \leq n$, while the Sorted Partition takes over as m exceeds n. Consequently, the $D\&C_n^1$ mechanism is optimised by toggling between these two methods to maximise the captured surplus.



(a) Round-Robin vs Sorted



(b) A closer look

Figure 4.1 Cumulative normalised mean utility under the $D\&C^1$ across different partitioning methods for a fixed profile of $n = 4$ agents and $m = 8$ objects.

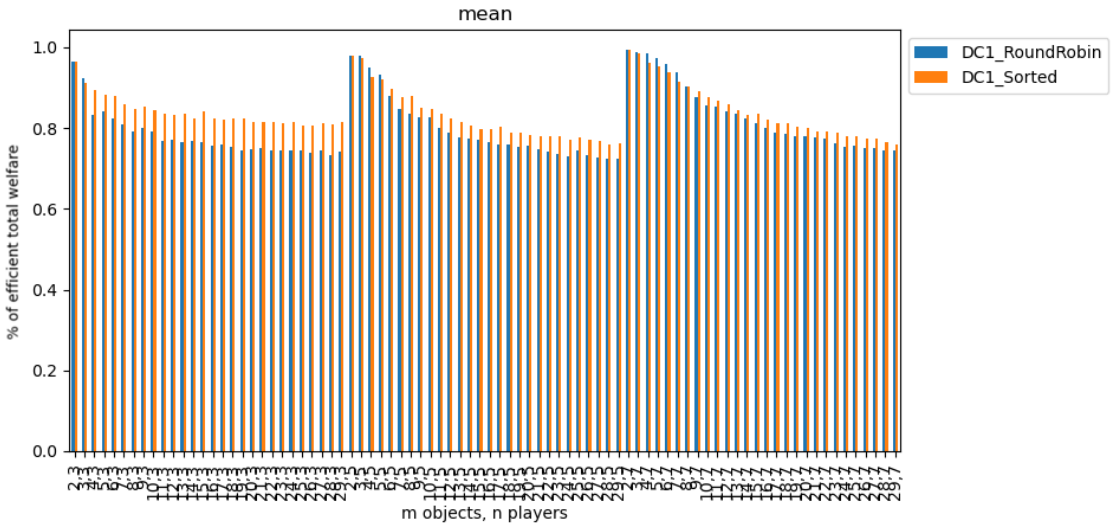


Figure 4.2 $D\&C_n^1$: Round-Robin vs Sorted partition - ordered by player

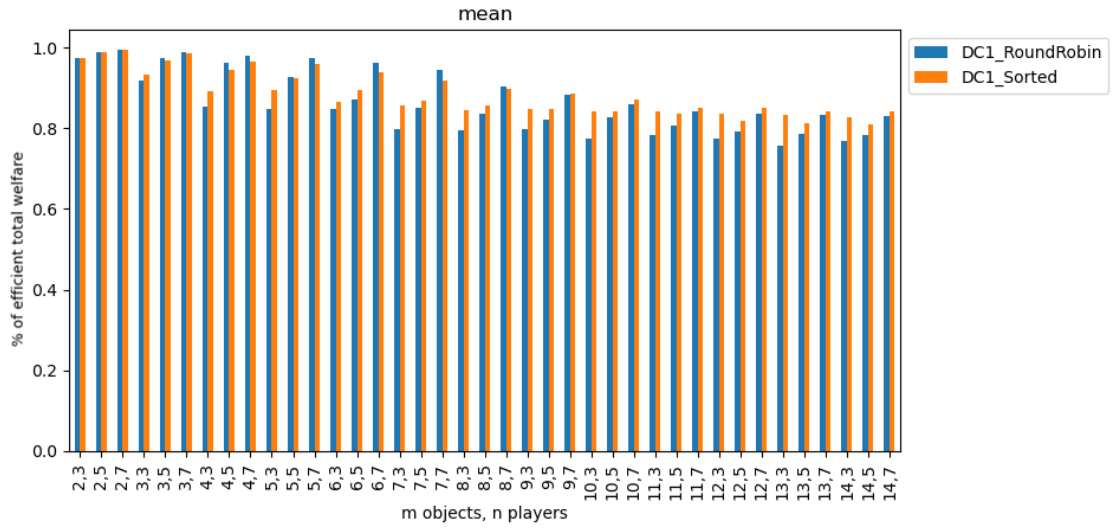


Figure 4.3 $D\&C_n^1$: Round-Robin vs Sorted partition - ordered by object

4.5.7 Main Results for the Additive Case

In the additive utility framework, the total maximal surplus $\mathcal{W}(\vec{u})$ is defined by the sum of maximum valuations for each object $a \in A$, i.e., the total utility after allocating each object to the agent who values it the most.

$$\mathcal{W}(\vec{u}) = \sum_{a \in A} \max_{i \in n} u_i(a)$$

The efficiency of each mechanism (ν) is measured as the percentage of this maximal surplus captured when players follow safe strategies: ν for a mechanism is defined as

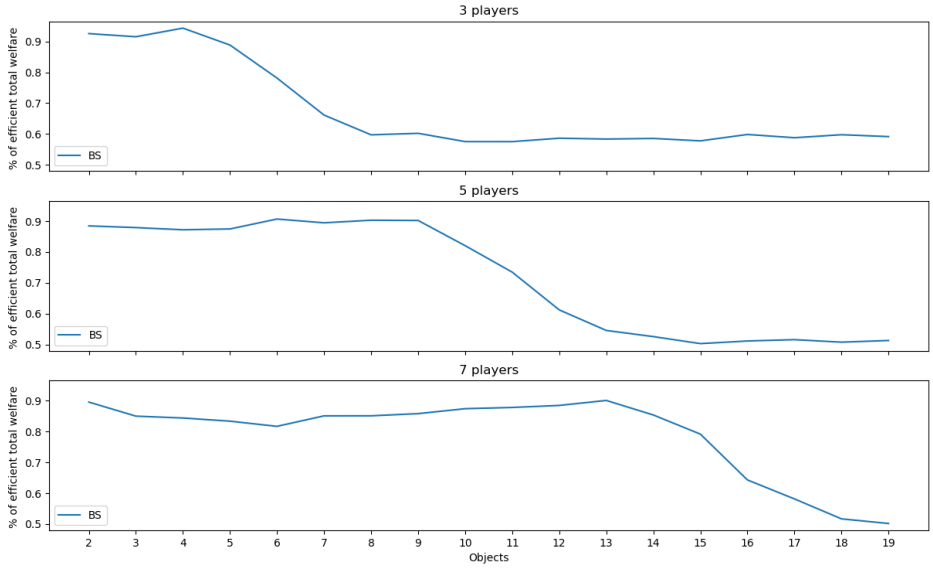


Figure 4.4 Average performance of the $B\&S$ mechanism across varying objects (m) and players (n). Simulations evaluate the performance under safe play across 100 random utility instances for $m \in [1, 100]$ and $n \in \{3, 5, 7\}$.

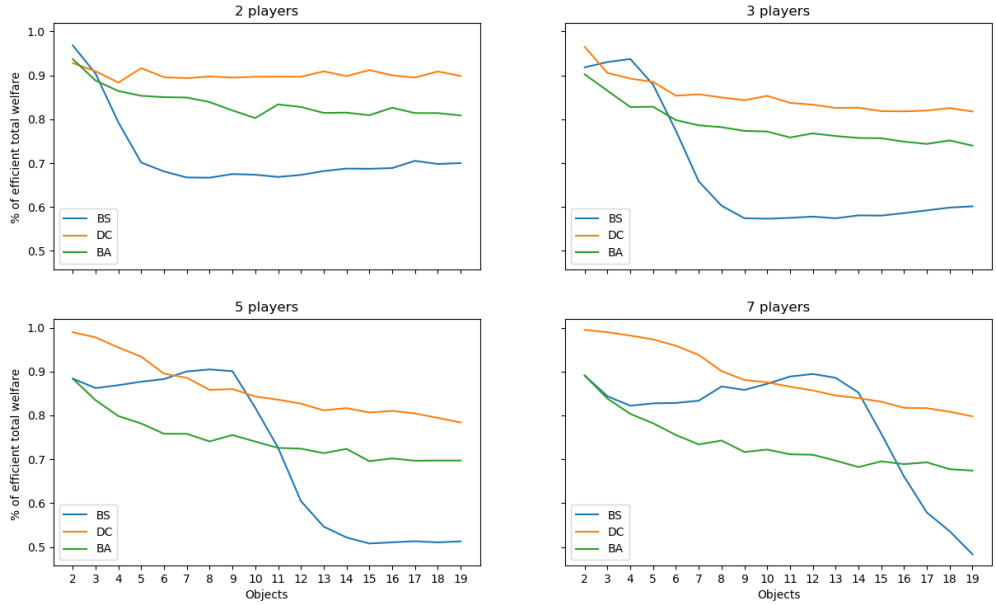


Figure 4.5 Percentage of maximal surplus ($\mathcal{V}$) by the three mechanisms under additive utilities. The results show that while the $B\&S$ mechanism achieves an efficiency peak near $m \approx 2n$, its performance drops sharply and plateaus as the object set expands and $D\&C$ dominantly captures a higher $\mathcal{V}$.

the total value obtained by all agents under independent safe play divided by the maximal efficient surplus $\mathcal{W}(\vec{u})$.

To obtain these results, we create 100 samples of each combination of players and objects ($n = 2, m \in (1, 20)$), and execute all three fair division algorithms for each sample, calculate the average $\mathcal{V}$ (percentage of maximal surplus captured) for each combination of players and objects, as displayed in Figure 4.5.

While the absence of externalities simplifies agent preferences, the results reveal distinct patterns in how mechanisms manage the indivisibility of goods and the distribution of surplus.

The Sweet Spot for *Bid & Sell*: Running our experiments on $B\&S$ algorithm, first we present the mean normalised total values ($\mathcal{V}$) in Figure 4.5. The $B\&S$ mechanism exhibits a non-linear relationship between object quantity and efficiency. While $B\&S$ is designed to extract more information via pricing, its performance in the additive case is subject to significant fluctuations as the set A grows. $B\&S$ reaches its "sweet spot" of maximum surplus capture when $m \approx 2n$. In this zone, the pricing mechanism effectively reveals enough preference data to rival or briefly exceed $D\&C$ outcomes. As the number of objects increases significantly beyond the number of players ($m \gg n$), $B\&S$ efficiency drops sharply and settles near 0.5. This suggests that as the market becomes more complex, the recursive pricing structure becomes less effective at capturing additive surplus compared to partitioning methods. To check if the path of the curve changes for a bigger interval of m, we execute the same program for an increased the range of $m \in (1, 100)$. We see that the mean total value curve plateaus and does not recover (Figure 4.4).

Dominance of *Divide & Choose*: The comparison of all three mechanisms (Figure 4.5) establishes a clear hierarchy for additive environments. Overall, $D\&C$ captures a larger share of the efficient surplus than both BA and $B\&S$. Its ability to partition goods into nearly equal subjective values minimise the need for large cash transfers, leading to higher realised utility for all players. The *Bundled Auction* acts as a consistent floor for efficiency. While it ensures proportionality, its lack of responsiveness (specifically its failure to distribute additive goods) leaves it trailing behind $D\&C$ in almost every scenario where $m > 1$. The additive results underscore that Responsiveness does not always require high informational complexity. While $B\&S$ is the most 'sophisticated' correction to the Proportional Share, the simpler partitioning logic of $D\&C$ is more

effective at capturing additive surplus. This suggests that mechanism selection should be contingent on the expected utility structure of the participants.

4.6 Comparison Under Sub/Superadditive Utilities

In this section, we generate at random some subadditive and superadditive utilities with which we can test the average efficiency of fair division rules. This case is complicated compared to the additive case since the safe play for each player is not straightforward to compute. In our algorithms, we make use of some function approximation techniques to approximate the optimal safe bid for the players.

4.6.1 Generating Players with Sub/Superadditive Utilities

We simulate random players with subadditive utilities at each instance. We achieve this by drawing from a uniform distribution U(1,100), m values for m objects. We also randomly allocate to each player a subadditive function that is either a power function with the power value between $\frac{1}{3}$ to $\frac{99}{100}$ or the function $c \cdot log(1+x)$ where the coefficient c is drawn at random. The random coefficient c is used to adjust the payoffs of players with subadditive utilities so they do not overpoweringly differ from the other players' utilities. The m randomly generated values along with the randomly assigned subadditive function makes up each player's payoff for all possible allocations of objects. In the case of superadditive utilities, we change the above procedure by using at random either a power function with the power value between 1 to 1.5 or the function of a parabola $f(x) = ax^2 + bx$ where $a \in U(0.003, 0.008)$ and $b \in U(0, 1)$.

Methodological Justification and Future Directions. Another direction for our experimental architecture could be the utilisation of structured random preference models, such as the Mallows Model Mallows (1957), or the incorporation of real data collected from actual human fair-division scenarios. In computational social choice and fair allocation, the *Mallows* distribution is often utilised to generate synthetic ordinal preferences centered around a central consensus ranking Amanatidis, Birmpas, Christodoulou, and Markakis (2020); Suksompong and Lang (2025). However, we chose to bypass consensus-driven models like *Mallows* in our core non-additive simulations. As noted by Amanatidis, Birmpas, Christodoulou, and Markakis (2020), when agent profiles are generated under tight correlation parameters, preference

profiles become highly aligned. In a fair allocation setting, forcing all n agents to hold close-to-identical item hierarchies eliminates structural preference asymmetry, which collapses potential gains from trade and forces mechanisms to rely heavily on extensive cash compensations for fairness and surplus-capture constraints. By contrast, pairing independent $U(1, 100)$ draws with distinct, randomly distributed subadditive and superadditive scaling functions preserves preference heterogeneity and offers a rigorous test for fair allocation algorithms.

Nonetheless, evaluating these mechanisms against real-world data remains a compelling avenue for future research. The primary real-world benchmark for this domain is the *Spliddit* dataset, a not-for-profit platform that has collected thousands of user-reported preference profiles for everyday fair division tasks, including rent splitting and indivisible goods division Goldman and Procaccia (2014); Gal, Mash, Procaccia, and Zick (2017). Because Spliddit forces users to distribute a fixed budget of $1,000$ points across items, it captures real-world sentimental and monetary valuations Shah (2017). In future work, these historical, anonymised point vectors could be extracted to initialize our simulation environments. This would allow us to evaluate how *B&S* and *D&C* operate under naturally occurring human utility distributions, which could be highly subjective item combinations that synthetic models cannot easily replicate.

4.6.2 Sub/Superadditive: Maximal Surplus

Maximal surplus in the case of sub/superadditive utilities needs us to evaluate the best combined payoffs among all possible allocations. Since for m objects and n players there are n^m ways to allocate objects, we have a limitation on the upper limit of m and n as 5 and 3 respectively for computational feasibility (full description in Section 4.7). For example, the number of possible allocations among 4 players spikes from 64 to 1024 as we change the number of objects from 3 to 5!

4.6.3 Sub/Superadditive: Bundled Auction

The *Bundled Auction* serves as our benchmark in the non-additive case. In this mechanism, the entire set of objects A is treated as a single indivisible lot. Each agent i submits a single bid β_i representing their valuation of the full bundle. The highest bidder i^* receives all objects and compensates the other $n - 1$ participants by paying each of them $\frac{1}{n}\beta_{i^*}$.

For subadditive (*Frugal*) or superadditive (*Greedy*) utilities, the safe strategy remains truthful bidding, where $\beta_i = u_i(A)$. This ensures that every agent is guaranteed their Proportional Share of $\frac{1}{n}u_i(A)$. As implemented in our simulation, the mechanism evaluates the utility of the total allocation for each player and selects the maximum.

4.6.4 Sub/Superadditive: Bid & Sell

Let us recall the *Bid & Sell* mechanism for two players.

1. Agents bid to become the Seller: this is the lowest bidder with bid x
2. Seller chooses a price $p \in \Delta(x)$
3. Buyer can buy any share S of objects at this price and the Seller takes the unsold objects

To understand how to bid safely, we observe the worst case utility of a player in case his bid is the lowest and wins. The worst case utility as a Seller is

$$\max_{p} \min_{\varnothing \in T \in A} \left(u(T) + p_{A \setminus T} \right)$$

or

$$\max_{p} \min_{\varnothing \in T \in A} \left(u(T) + x - p_T \right)$$

where T is the set of all unsold objects. Meanwhile, the worst case utility as a Buyer is

$$\min_{p} \max_{\varnothing \in S \in A} \left(u(S) - p_S \right)$$

where S is the set of objects purchased. As shown in Bogomolnaia and Moulin (2025), the maxmin function is increasing and minmax function is decreasing. Hence the safe bid is at the intersection of the two functions x^*. To consider all subsets of objects sold by the Seller could be feasible to program depending on the number of objects (2^m possibilities), but to find the maximising/minimising vector of prices p that sum up to the bid x is infeasible. The exact Linear Program has 2^m constraints and is exponential in m (see Section 4.7); we therefore approximate the functions using hill-climbing:

- We discretise the function using evenly spaced nodes on the x-axis to find candidate bids.

- For each candidate bid node$_i$, we use a hill climber function (see Appendix B for the procedure) to find the utility maximising vector of prices p that sums up to node$_i$.
- Finally, we use a small threshold value for the distance between the two functions to find the approximate safe bid x^* that lies around their intersection point (see Figure 4.6).

Having calculated the approximate safe bid, if the player wins with the bid, the next stages of the $B\&S$ mechanism are executed.

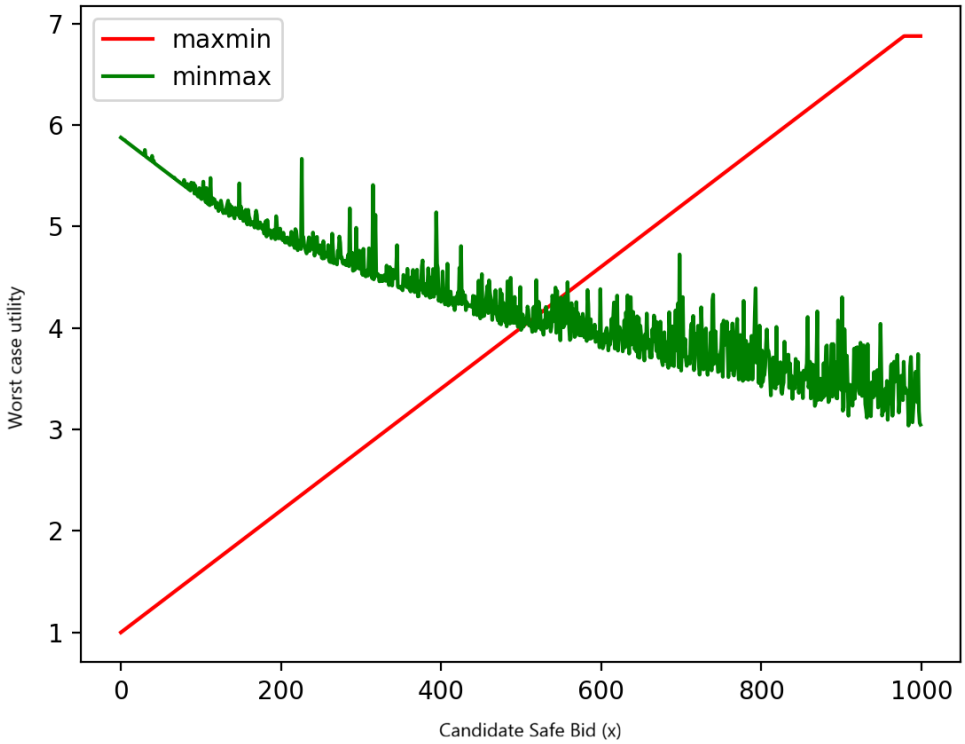


Figure 4.6 $B\&S$ bidding in safe-play: Intersection of worst case as Seller vs worst case as Buyer.

In our non-additive simulations, the logical search space for candidate bids x is bound by $[0, \max(u_i(A))]$. Given that individual item valuations are drawn from $U(1, 100)$ and subsequently scaled via subadditive log functions or superadditive quadratic parabolas $f(x) = ax^2 + bx$, the maximum potential utility for the entire bundle asymptotically approaches a ceiling of $1,000$. Consequently, our safe bid calculation uses a discretised evaluation grid with uniform nodes from 0 to $1,000$. This structure guarantees that the approximate game-theoretic intersection point x^* is captured.

4.6.5 Sub/Superadditive: Divide & Choose

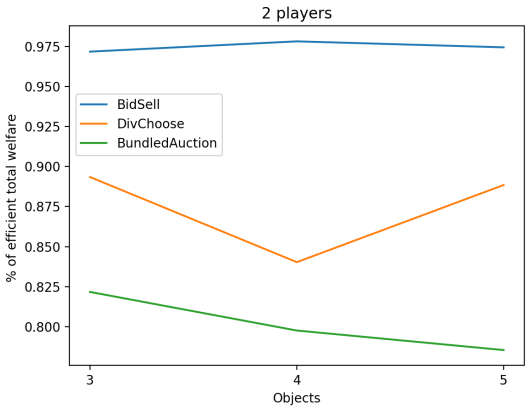
In the *D*&*C* rule, agent i's play is safe iff he bids $\beta_i = \text{Maxmin}_n(u_i) - \text{minMax}_n(u_i)$ in during the bidding round, chooses a partition π_i maximising $u_i(\pi)$ if the bid wins and he becomes the *Divider*, and reports truthfully equalising transfers across the shares of π if he becomes a *Chooser*. To compute the safe bid given one's utilities and the number of players is feasible but becomes costly as the number of partitions and therefore the number of possible allocations grows. For this reason, we limit the number of objects to up to 5. We calculate the minMax, Maxmin utilities and the best partition for each player. The player with the highest safe bid becomes the *Divider* and her best partition proposed is chosen and offered to the *Choosers*.

4.6.6 Main Results for the Sub/Superadditive Case

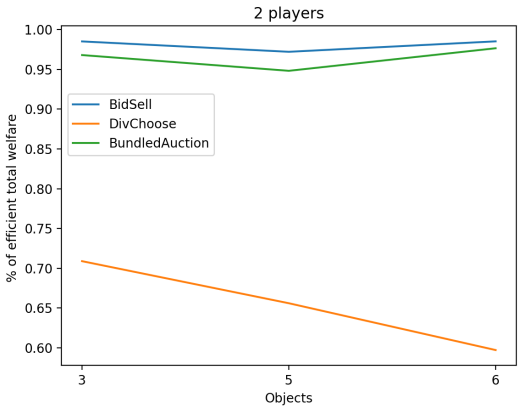
The primary objective of these simulations was to evaluate the hypothesis that mechanism responsiveness (the ability of a rule to adapt to the specific structure of agent utility functions) leads to higher social welfare efficiency. By deploying the *Bundled Auction* (*BA*) as an unresponsive baseline, which forces the allocation of the entire set A regardless of agent preferences, we isolate the efficiency gains achieved by the responsive mechanisms *Bid & Sell* (*B*&*S*) and *Divide & Choose* (*D*&*C*). The results across additive, subadditive (frugal), and superadditive (greedy) scenarios confirm that responsiveness is a critical driver of efficiency, though its relative impact varies significantly depending on the nature of the utility landscape.

The 'Unbundling' Surplus in Subadditive Environments The most dramatic demonstration of responsiveness appears in subadditive environments (Figures 4.7a and 4.7d). Here, agents experience diminishing marginal returns and in extreme cases, value a small subset of goods nearly as highly as the entire bundle.

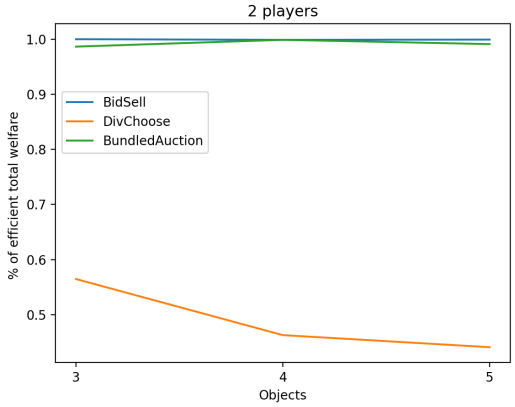
- Failure of the Benchmark: While the *Bundled Auction* ensures Positivity, it is inherently Unresponsive. Because it only considers the valuation of the full set $u(A)$, it fails to distinguish between an easy-to-satisfy *Frugal* agent and a hard-to-satisfy *Greedy* agent, as both typically value the entire set at 1. This results in an identical $\frac{1}{n}$ guarantee for all participants, failing to capture the potential efficiency gains from unbundling items that a *Frugal* agent might find redundant. By forcing a single winner to take all m items, *BA* inherently wastes



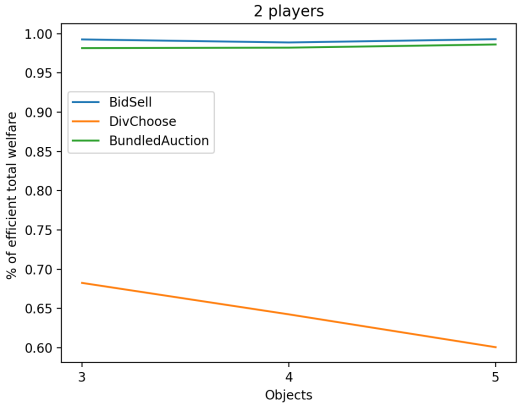
(a) Comparison under sub-additive utilities



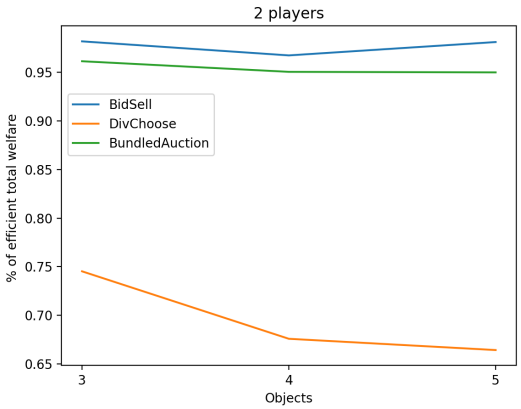
(b) Comparison under super-additive utilities



(c) sub-additive vs super-additive utilities



(d) additive vs sub-additive utilities



(e) additive vs super-additive utilities

Figure 4.7 Comparison of $\mathcal{V}$ across $B\&S$, $D\&C$, and BA mechanisms under heterogeneous utility distributions (additive, subadditive, and superadditive). Each data point represents the average surplus captured under independent safe play, demonstrating the efficiency dominance of $B\&S$ over $D\&C$.

utility. The winner receives redundant items that add little value, while losers who could have derived significant utility from single items receive nothing but cash.

- Dominance of *Bid & Sell*: *B*&*S* achieves the highest efficiency because it offers the finest granularity of choice. In a subadditive landscape, the safe seller strategy involves setting relatively low prices for individual items (since the sum of item values is high relative to the bundle value). This allows buyers to 'cherry-pick' only the items that maximise their net utility, effectively distributing the goods among those who value them most individually. Hence, *B*&*S* successfully captures the 'unbundling' surplus in a subadditive environment.
- *D*&*C* Performance: *D*&*C* also outperforms *BA* by allowing the set of objects to be broken. However, it struggles to match *B*&*S* because creating somewhat equal partitions for subadditive agents is difficult due to the indivisibility of items. *B*&*S*'s use of continuous pricing provides a smoother mechanism for balancing utility than *D*&*C*'s discrete partitioning.

The 'Alignment of Greed' in superadditive environments In stark contrast, superadditive environments (Figures 4.7b and 4.7e) present a scenario where the inefficiency of the baseline mechanism is minimised. Here, greedy agents require the entire bundle (or large parts of it) to realise utility.

- Resilience of the Benchmark: The *Bundled Auction* performs surprisingly well in superadditive cases and falls short in subadditive cases, which is the expected behaviour. Its inherent rigidity (allocating the entire set *A*) aligns perfectly with the rigid demands of superadditive agents. Since everyone wants more rather than less of the whole bundle, *BA* simply ensures it goes to the agent with the highest total valuation, which is a reasonably efficient outcome in this specific context.
- Challenges for Responsive Mechanisms: While *B*&*S* still generally yields the highest returns, its lead over *BA* narrows significantly when the agent pool contains those with superadditive behaviours. The safe strategy for a superadditive seller is to set very high individual prices (summing to their high bundle valuation) to prevent the buyer from cheaply acquiring necessary complements. This high price floor can stifle trade, leading to outcomes similar to the *BA* where the seller simply retains the bundle.

- *D&C* performs worst here, as it is often difficult to create a balanced partition when utility only exists in the larger subsets or the whole set itself, leading to high transfer payments and little actual utility distribution.

Overall Mechanism Ranking Across the broad spectrum of utility types tested (summarised in Figure 4.7), a clear hierarchy of mechanism performance emerges, closely linked to the granularity of choice they offer:

- *Bid & Sell* (Highest Efficiency): By decoupling the roles of buyer and seller and utilising continuous price vectors, *B&S* provides the highest degree of responsiveness. It successfully unbundles items for frugal agents and extracts high value from greedy agents, consistently yielding the highest percentage of maximum possible surplus.
- *Divide & Choose* (Moderate Efficiency): *D&C* offers discrete responsiveness through partitioning. It is superior to bundling when agents desire subsets (subadditive / additive) but fails when preferences are highly complementary (superadditive) because the physical indivisibility of goods prevents the creation of acceptable shares.
- *Bundled Auction* (Lowest vs. Contextual Efficiency): As the unresponsive baseline, *BA* provides a floor for efficiency. It is the worst performer when utility structures are diverse or subadditive. However, it is a robust and simple alternative in highly polarised superadditive markets where its lack of nuance is not a handicap.

The cross-family utility comparisons reinforce the primary conclusion that *B&S* is the superior mechanism for maximising efficiency in the presence of complex and heterogeneous player preferences in this environment, consistently capturing a much higher percentage of the maximal social welfare ($\mathcal{W}(\vec{u})$) compared to *D&C* and *MA*. The comparison against the *Bundled Auction* baseline definitively shows that complex, responsive fair division mechanisms are worth the computational cost in almost all realistic scenarios. The ability of a mechanism like *Bid & Sell* to allow agents to express preferences within the set of goods (rather than just for the set of goods) is the primary engine of economic efficiency in fair division.

4.7 Computational Complexity

We analyse the complexity of each mechanism in terms of n agents and m goods, assuming $O(1)$ utility evaluation. The results directly explain the experimental parameter limits imposed in Sections 4.5 and 4.6.

Bundled Auction. BA collects n bids, selects the maximum, and distributes payments: $O(n)$. The safe bid $\beta_i = u_i(A)$ costs $O(m)$ for additive utilities. Overall: $O(n + m)$.

Divide & Choose. $D\&C$ contains two distinct computational bottlenecks.

MaxMin/MinMax Partition Computation (Non-Additive Case). Evaluating the complete preference space requires searching across all n^m candidate assignments of m goods to n shares, where each validation costs $O(n)$.

- In $D\&C^1$: Every agent i must compute both their MaxMin share and MinMax share to determine their unique safe bid. This requires executing two distinct partition space searches, costing $2 \times O(n^{m+1})$ per agent, yielding an aggregate complexity of $O(n^{m+2})$.
- In $D\&C^2$ (see Bogomolnaia and Moulin (2025)): Agents do not submit bids; instead, they directly propose their most preferred partition. This requires evaluating the MaxMin direction only. Computing this single search space requires $O(n^{m+1})$ steps per agent, which sums to $O(n^{m+2})$ across all n participants. The averaging auction in $D\&C^2$ processes the n submitted candidate partitions by executing the Hungarian algorithm across an $n \times n$ transfer matrix for each proposal, which incurs an operational cost of $O(n \times n^3) = O(n^4)$, which is dominated.

$$O(n^{m+2} + n^4) \longrightarrow O(n^{m+2})$$

Consequently, both variants are asymptotically equivalent at $O(n^{m+2})$, with $D\&C^1$ merely carrying an additional constant factor of 2. This necessitates restricting our non-additive experiments to $m \leq 5$ and $n \leq 3$ (e.g., $3^5 = 243$ evaluations per agent optimisation path). In the strictly additive case, both MaxMin and MinMax collapse to the Proportional Share, making the safe bid 0 and reducing partition selection to a simple sorting operation: $O(m \log m)$.

π-auction assignment. Finding the minimum-slack assignment matrix is a linear assignment problem. While a brute-force enumeration of the $n \times n$ matrix costs $O(n!)$,

our implementation uses the Munkres variant of the Hungarian algorithm, bounding this bottleneck to $O(n^3)$.

Combining these steps:

$$D\&C^1 : O(n^{m+2} + n^3), \qquad D\&C^2 : O(n^{m+2} + n^3).$$

In our experiments, we only implement the $D\&C^1$ version of the mechanism.

Bid & Sell. The safe bid requires solving two linear programs, W_n and L_n, each with m price variables and 2^m constraints (one per subset $T \subseteq A$). Each LP costs $O(\text{poly}(2^m))$. The recursive n-player rule requires $O(n^2)$ such LP solves (Bogomolnaia and Moulin, 2025, Section 8.1), giving:

$$B\&S_n : O\Big(n^2 \cdot \text{poly}(2^m)\Big).$$

This is polynomial in n for fixed m, and exponential in m in general. For *identical goods*, the LP reduces to a one-dimensional search (Lemma 11, ibid.), yielding $O(n^2 m)$ — polynomial in both dimensions. Our implementation approximates the LP by discretising the bid axis with k nodes and using hill-climbing per node.

The safe bid requires solving two linear programs, W_n and L_n. Because agents hold combinatorial valuations, each LP must incorporate a separate constraint for every possible subset combination $T \subseteq A$, generating 2^m constraints. Solving these exact equations costs $O(\text{poly}(2^m))$, where poly$(\cdot)$ represents a polynomial time complexity function with respect to the input size; this makes exact computation polynomial in n for fixed m, but exponential in terms of the objects m. The recursive n-player $B\&S_n$ rule requires $O(n^2)$ such LP iterations (Bogomolnaia and Moulin, 2025, Section 8.1) yielding:

$$B\&S_n : O\Big(n^2 \cdot \text{poly}(2^m)\Big).$$

To circumvent this exponential bottleneck in our empirical simulations, we implement a localised approximation algorithm. The bid axis is discretised using k coordinate nodes. For the maximisation bounds in W_n, a *hill-climbing* search is executed per node to find local utility maxima; symmetrically, for the minimisation in L_n, a *valley-descent* is executed to isolate local utility minima. Letting I denote the maximum number of iterations required for either local search to converge, each step scales linearly with the number of objects m. This heuristic reduces the complexity per LP instance to $O(k \cdot I \cdot m)$, yielding a global simulation runtime of:

$$\text{Approximate } B\&S_n : O(n^2 \cdot k \cdot I \cdot m).$$

When agent utilities are strictly additive, because bundle values collapse to the sum of their individual components, the constraint matrix in the exact theoretical W_n and L_n problems drops from 2^m down to m. This eliminates the exponential dimensionality and allows the exact theoretical mechanism to be solved in polynomial time: $O(n^2 \cdot m)$.

Summary. Table 4.2 collects all computational characteristics. $D\&C$'s binding constraint is on player count n via the partition-space search, whereas exact $B\&S$'s is on object count m due to the $O(2^m)$ constraint matrix. These complementary bottlenecks make $D\&C$ more tractable when n is small and m is large, and $B\&S$ preferable when m is small and n is moderate. Our hill-climbing approximation somewhat removes $B\&S$'s object bottleneck and renders it mildly scalable across larger m.

Table 4.2 Computational complexity summary (n = agents, m = goods, k = discretisation nodes, I = hill-climbing iterations). Individual utility evaluations are assumed to scale at $O(1)$.

Mechanism	**Overall Complexity**	**Binding Constraint**
BA	$O(n + m)$	None
$D\&C^1$ (additive)	$O(m \log m + n^3)$	None (n scalable via Hungarian algorithm)
$D\&C^1$ (non-additive)	$O(n^{m+2} + n^3)$	$m \leq 5,\ n \leq 3$
$B\&S_n$ (Additive Case)	$O(n^2 \cdot m)$	None
$B\&S_n$ (Exact Theoretical)	$O(n^2 \cdot \text{poly}(2^m))$	Exponentially bounded by large m
$B\&S_n$ (Approximate Heuristic)	$O(n^2 \cdot k \cdot I \cdot m)$	Bounded linearly by resolution parameter k

Appendices

Appendix A

Appendix to Chapter 1: An Example

We revisit the Hauk and Hurkens (2002) example to demonstrate how pure duplicates and auxiliary players replicate the effect of a mixed duplicate action. We set the perturbation $\epsilon = 0.01$.
The normal form game $\tilde{B}^{1.1}$ incorporates the original HH payoffs, the pure duplicates L' and C', and the auxiliary Player 3 who plays a version of the Levy matching pennies game against Player 2.

Table A.1 Payoff table: normal form game $\tilde{B}^{1.1}$

P_3 :A	L	C	R	L'	C'
T	5,4,0	-15,0,0	-11,2,0	$5, \frac{21364}{6561}, \frac{4880}{6561}$	$-15, \frac{1600}{6561}, \frac{-14884}{6561}$
M	-23,0,0	-1,8,0	3,1,0	$-23, \frac{-4880}{6561}, \frac{4880}{6561}$	$-1, \frac{54088}{6561}, \frac{-14884}{6561}$
B	-1,2,0	-21,-2,0	1,3,0	$-1, \frac{8242}{6561}, \frac{4880}{6561}$	$-21, \frac{-11522}{6561}, \frac{-14884}{6561}$
Out	0,0,0	0,0,0	0,0,0	$0, \frac{-4880}{6561} \cdot 0.01, \frac{4880}{6561}$	$0, \frac{1600}{6561} \cdot 0.01, \frac{-14884}{6561}$

P_3 :B	L	C	R	L'	C'
T	5,4,0	-15,0,0	-11,2,0	$5, \frac{41128}{6561}, \frac{1600}{6561}$	$-15, \frac{-4880}{6561}, \frac{4880}{6561}$
M	-23,0,0	-1,8,0	3,1,0	$-23, \frac{14884}{6561}, \frac{1600}{6561}$	$-1, \frac{47608}{6561}, \frac{4880}{6561}$
B	-1,2,0	-21,-2,0	1,3,0	$-1, \frac{28006}{6561}, \frac{1600}{6561}$	$-21, \frac{-18002}{6561}, \frac{4880}{6561}$
Out	0,0,0	0,0,0	0,0,0	$0, \frac{14884}{6561} \cdot 0.01, \frac{1600}{6561}$	$0, \frac{-4880}{6561} \cdot 0.01, \frac{4880}{6561}$

Equilibrium Analysis: Let x, y, and z be the strategy profiles for Players 1, 2, and 3 respectively. Player 3's payoffs are non-zero only when Player 2 assigns positive probability to the pure duplicate actions L' and C'. The best reply (x^*, y^*, z^*) calculation for each player:

$$x = (p_1, p_2, p_3, 1 - \sum p)$$

$$y = (q_1, q_2, q_3, q_4, q_5 = 1 - \sum q)$$

$$z = (r, 1 - r)$$

Player 3's payoffs are non-zero only when Player 2 assigns positive probability to the pure duplicate actions L' and C'.

$$U_3(x, y, A) = (\frac{4880}{6561}) \cdot q_4[p_1 + p_2 + p_3 + (1 - \sum p)] + (\frac{-14884}{6561}) \cdot (1 - \sum q)[p_1 + p_2 + p_3 + (1 - \sum p)]$$

Simplified,

$$U_3(x, y, A) = (\frac{4880}{6561}) \cdot q_4 - (\frac{-14884}{6561}) \cdot (1 - \sum q)$$

Similarly,

$$U_3(x, y, B) = (\frac{-1600}{6561}) \cdot q_4 - (\frac{4880}{6561}) \cdot (1 - \sum q)$$

For Player 3 to mix between actions A and B in equilibrium, their expected payoffs must be equal: $U_3(x, y, A) = U_3(x, y, B)$. This indifference condition for Player 3 occurs if and only if Player 2 mixes between the duplicates L' and C' in the following ratio:

$$\frac{q_4}{q_4 + q_5} = \frac{61}{81} = p^* \quad \text{and} \quad \frac{q_5}{q_4 + q_5} = \frac{20}{81} = 1 - p^*$$

where q_4 is the weight on L' and q_5 is the weight on C'. The auxiliary construction ensures that if Player 2 uses either or both of the duplicate strategies L', C', they are disciplined by Player 3 to use the relative weights $\frac{61}{81}$ and $\frac{20}{81}$.

As demonstrated in Hauk and Hurkens (2002), adding a duplicate action with these specific weights and applying the perturbation ϵ eliminates all equilibria in the neighbourhood U' of the "Out" outcome. Since our pure duplicate construction forces this exact behaviour, the "Out" component is successfully eliminated, confirming that the equilibrium is not hyperstable.

Appendix B

Appendix to Chapter 3

B.1 Illustrative n-Player Example: 2 *Greedy* vs. 1 *Frugal*

Before going into the example, here is the recursive definition for B&S. There are $n-1$ rounds of bidding in $B\&S_n(A)$ as long are not all objects in A are purchased in the previous rounds. In each round one agent becomes the Buyer and all others become Sellers; the Buyer leaves after being offered to buy some goods from all the Sellers. The recursive algorithm:

1. each agent i bids x_i. The highest bidder becomes the Buyer.
2. each of the remaining players becomes a Seller, and chooses a price p_j in simplex $\Delta(x_j)$ such that the price of the whole bundle is x_j.
3. the Buyer buys a share S of A (possibly ϕ), where each item in S is bought at the best price p_j offered, provided p_j for the item is not higher than the buyer's utility u_i from the item. The Buyer pays $p_j(S)$ to the Seller and leaves; the rule stops if $S = A$.
4. the remaining agents play $B\&S_{n-1}(A \setminus S)$.

Bid & Sell To illustrate the recursive nature of the *Bid & Sell* ($B\&S_n$) mechanism for three players, we consider a set of three identical goods ($m = 3$) shared between two *Greedy* agents and one *Frugal* agent.
Frugal (F): $u_F(k) = 1$ for any $k \geq 1$ object. She is satisfied with a single item.
Greedy 1 (G_1): $u_{G_1}(3) = 1$; 0 otherwise. He only values the full set.

Greedy 2 (G_2): $u_{G_2}(3) = 1$; 0 otherwise. He also only values the full set.
The proportional share for each would be 1/3.

To determine the individual guarantees for *Frugal* and the two *Greedy* agents in the 3-player *Bid & Sell* ($B\&S_3$) mechanism, we apply the recursive logic where safe play is found at the intersection of an agent's utility as a Seller and a Buyer. If *Frugal* as a Seller sets a uniform price $p_a = x/m$ for every object,

- If the Buyer buys everything ($S = A$): *Frugal* receives the full bid x.
- If the Buyer buys nothing ($S = \emptyset$): *Frugal* keeps all goods, worth 1.
- If the Buyer buys some goods ($S \subset A$): Since at least one good remains, her object utility is still 1, and she also receives the cash $p(S)$. Her utility is $1 + p(S)$, which is > 1.

Therefore, with uniform pricing, *Frugal*'s worst-case utility as a Seller is $\min\{x, 1\}$. As Buyer, *Frugal* only needs to buy the single cheapest item from one of the two Sellers (G_1 and G_2) to get a utility of 1. In the worst case, both Sellers set identical uniform prices based on their bids (which are just below x). She pays $\min\{p_{G_1}, p_{G_2}\} = \frac{x}{m}$. Her utility is $1 - \frac{x}{m}$.

Safe Bid Calculation:

$$x = 1 - \frac{x}{m} \implies x(1 + \frac{1}{m}) = 1 \implies x(\frac{m+1}{m}) = 1$$

$$x^* = \frac{m}{m+1}$$

For $m = 3$, *Frugal*'s guarantee is her worst case utility when playing safe, which is $\frac{m}{m+1} = \mathbf{3/4} = 0.75$.

Greedy agents (u_G) must acquire the entire set to get utility. A *Greedy* Seller is at the mercy of the Buyer. In the worst case, the Buyer buys only one item. The Seller is left with $m - 1$ items (worth 0 to him) and the price of that one item. To protect themselves, the *Greedy* Seller sets a uniform price of $\frac{1}{m}y$ for every item. His utility is $\frac{1}{m}x$. As Buyer, to get any utility, *Greedy* must buy every item $a \in A$. For each item, he pays the minimum of the prices offered by the two Sellers. If both Sellers set uniform prices $\frac{x}{m}$ for all items, *Greedy* pays $\frac{x}{m}$ for each of the m items. Total cost $= m \times \frac{x}{m} = x$. His net utility is: $1 - x$.

Safe Bid Calculation:

$$\frac{1}{m}x = 1 - x \implies x(1 + \frac{1}{m}) = 1$$

$$x^* = \frac{m}{m+1}$$

Final Guarantee Value:

$$\Gamma_3^{BS}(G) = \frac{1}{m} \times \frac{m}{m+1} = \frac{1}{m+1}$$

For $m = 3$, each *Greedy* agent's guarantee is $\frac{1}{m+1} = \mathbf{1/4} = 0.25$.

Divide & Choose In this scenario, we calculate the safe bids and guarantees by evaluating the roles of Divider and Chooser. From the *Greedy* Agents' Perspective (G_1 and G_2), as Divider, a *Greedy* agent wants to ensure that even if they receive the 'worst' share, they are compensated. They propose a partition $\pi = \{A, \emptyset, \emptyset\}$. By attaching balanced transfers, they ensure each of the three shares is worth exactly $\frac{1}{3}$ to them, leading to a MaxMin utility of $\frac{1}{3}$. As Chooser, *Greedy*'s worst case is if the Divider is *Frugal*. She might propose a partition where each share contains one object. Since *Greedy* values any subset $S \subsetneq A$ at 0, all three shares are worth 0 to him. This leads to a MinMax utility of 0. Safe Bid (x_G): The safe bid is the difference between MaxMin and MinMax. Thus, $x_G = \frac{1}{3} - 0 = \frac{1}{3}$. Guarantee: $\Gamma_3^{DC}(G) = \frac{1}{3}(1/3) + \frac{2}{3}(0) = \mathbf{1/9}$.

From the *Frugal* Agent's Perspective (F), as Divider, *Frugal* proposes a partition where each of the three shares contains one object. Since she values any single object at 1, her MaxMin utility is 1. As Chooser, in the worst case, the Divider proposes the bundle partition $\pi = \{A, \emptyset, \emptyset\}$. *Frugal* must pick one share. The worst-case balanced transfers attached to this partition make all shares worth $\frac{1}{3}$ to her. Her MinMax utility is $\frac{1}{3}$. Safe Bid (x_F): $x_F = 1 - \frac{1}{3} = \frac{2}{3}$. Guarantee: $\Gamma_3^{DC}(F) = \frac{1}{3}(1) + \frac{2}{3}(1/3) = \mathbf{5/9}$. The money transfers reported by all 3 agents for the two partitions are given in Tables B.1 and B.2.

Table B.1 Money Transfer Matrix for *Greedy*'s Partition $\pi = \{A, \emptyset, \emptyset\}$

Agent	**Share S_1 (A)**	**Share S_2 ($\emptyset$)**	**Share S_3 ($\emptyset$)**	**Sum**
Frugal	−2/3	+1/3	+1/3	0
Greedy 1	−2/3	+1/3	+1/3	0
Greedy 2	−2/3	+1/3	+1/3	0

Table B.2 Money Transfer Matrix for *Frugal*'s Partition $\pi = \{\{1\}, \{1\}, \{1\}\}$

Agent	**Share S_1 {1}**	**Share S_2 {1}**	**Share S_3 {1}**	**Sum**
Frugal	0	0	0	0
Greedy 1	0	0	0	0
Greedy 2	0	0	0	0

Bundled Auction *Frugal*'s Bid: Since *Frugal* values any non-empty set at 1, her utility for the full bundle is $u_F(A) = 1$. Her safe bid is $\beta_F = 1$. *Greedy*'s Bid: *Greedy* agents only value the full bundle at 1. Their safe bids are $\beta_{G_1} = 1$ and $\beta_{G_2} = 1$. The winner pays $\frac{1}{3}$ to the other agents and receives the entire set of objects A. Each agent is guaranteed their Proportional Share of $\frac{1}{3} \approx 0.33$.

B.2 Illustrative Example: π-Auction

Imagine three agents (X, Y, and Z) and a partition of goods into three shares: $\pi = \{ac, b, \emptyset\}$. Each agent submits balanced transfers to make every share equal to their personal average utility $\Gamma_n^\pi(u_i)$. The values below represent the money each agent needs to be paid (+) or is willing to pay (−) to feel they have received their fair share.

Table B.3 Example of Balanced Transfers in a π-auction ($n = 3$)

Agent	**Share 1**	**Share 2**	**Share 3**	**Sum**
Agent X	**-10**	+2	+8	0
Agent Y	-5	**-3**	+8	0
Agent Z	-2	-1	**+3**	0

Note: Bold values indicate the assignment that minimises total transfers (Total Slack = -10).

Step 1: Choosing the Assignment The auction selects the assignment (one per row/column) that results in the lowest sum. We choose:

- Agent X → Share 1 (−10)
- Agent Y → Share 2 (−3)

- Agent Z $\rightarrow$ Share 3 (+3)
- Total Slack: $(-10) + (-3) + (+3) = \mathbf{-10}$.

Step 2: Distributing the Surplus The total slack of -10 means the agents are willing to pay a net of 10 into the system. This creates a surplus of 10, which is divided equally ($10/3 \approx 3.33$ per person).

Step 3: Final Utility Each agent receives their guaranteed utility plus the rebate: Agent X receives the best share, pays 10, but gets 3.33 back. Agent Z receives the empty share, is paid 3 (as requested), and gets an additional 3.33 rebate.

B.3 Pseudo-Code

Algorithm 3: Agent Utility Generation for Non-Additive Scenarios

Function `initialise_sub`(*objects* m, *players* n)**:**
- $U\_list \leftarrow []$
- $Ufn\_list \leftarrow []$
- $subAddFns \leftarrow$ [*"power0.5"*, *"power2/3"*, *"power1/3"*, *"powerx"*, *"logsubadd"*]
- **for** $i \leftarrow 0$ **to** $n-1$ **do**
  - U_list.append($U(1, 100, m)$; `// uniform random of size` m
    $fn \leftarrow$ `random`($subAddFns$)
  - `// Assign specific subadditive profile and attributes`
  - **if** $fn =$ *"powerx"* **then**
    - $p \leftarrow U(1/3, 99/100)$
    - Ufn_list.append(power(p))
  - **else if** $fn =$ *"logsubadd"* **then**
    - $c \leftarrow U(1, 3)$
    - Ufn_list.append(log(c))
- **return** U_list, Ufn_list

Algorithm 4: Maximal Surplus and *Bundled Auction* Benchmarks

Function `maxSurplus()`**:**
- $allocs \leftarrow$ `getAllAllocations`(m, n)
- **return** $\max_{alloc \in allocs} \sum_{p=0}^{n-1}$ `getUtility`($alloc[p], U_p, Ufn\_p$)

Function `BundledAuction()`**:**
- $total\_set \leftarrow \{0, \ldots, m-1\}$
- $utils \leftarrow$ [`getUtility`($total\_set, U_p, Ufn\_p$) **for** *player* $p \in n$]
- **return** `max`($utils$)

Algorithm 5: Partitioning Strategies for Divide & Choose

```
Function OrderedPartition(u, n):
    tippingPt ← ∑(u)/n
    parts ← [[] for n shares]
    sorted_idx ← argsort(u)
    for i ∈ sorted_idx do
        if ∑ u[parts[curr]] + u[i] < tippingPt then
            parts[curr].append(i)
        else if curr < n − 1 then
            curr ← curr + 1
            parts[curr].append(i)
    return parts
Function RoundRobinPartition(u, n):
    sorted_idx ← argsort(u)
    parts ← [[] for n shares]
    for idx, i in enumerate(sorted_idx) do
        parts[idx (mod n)].append(i)
    return parts
```

Algorithm 6: *Divide & Choose* Procedure ($D\&C_n^1$)

```
Function DC1_additive(U_list, fPartition):
    divider ← random({0 . . . n − 1})
    parts ← fPartition(U_divider, n)
    // Generate balanced money transfers
    money ← [(∑ U_i/n) − getUtilityForParts(parts, U_i) for each player i]
    alloc ← HungarianAlgorithm(money)          // Optimal assignment via
     Hungarian Algorithm
    return ∑_{i=0}^{n−1} utility(parts[alloc[i]], U_i)
```

Algorithm 7: Recursive *Bid & Sell* Logic

```
Function SB_additive_Recursive(U_dict, items, playerUtils):
    if not items then
        return playerUtils
    if length (U_dict) == 1 then
        return playerUtils+∑(U_dict.values())
    buyer, u_buyer ← select player with max total utility
    U_dict.pop(buyer)
    min_price ← min([∑u/n for u ∈ U_dict.values()])
    S ← {j ∈ items | u_buyer[j] ≥ min_price}
    return SB_additive_Recursive(U_dict, items\ S, playerUtils+ u_buyer(S))
```

Algorithm 8: Pricing Approximation via Hill Climbing

```
Function climb_next_step(start, m, x, increment, f, current_max):
    neighbours ← generate nearby Dirichlet-distributed price vectors
    maxPoint ← start
    for neighbour ∈ neighbours do
        newVal ← f(neighbour)
        if newVal > current_max then
            current_max ← newVal; maxPoint ← neighbour
    return current_max, maxPoint
Function hill_climber(m, bid, objective_fn):
    p ← random Dirichlet distribution scaled by bid
    val ← objective_fn(p)
    while better neighbour exists do
        val, p ← climb_next_step(p, m, bid, 0.1, objective_fn, val)
    return val, p
```

Algorithm 9: Hungarian Algorithm for Optimal Assignment

```
Function HungarianAlgorithm(matrix):
    // Identify the assignment that minimises total cost/transfer
    m ← Munkres object initialisation
    indices ← m.compute(matrix)
    winning_permutation ← []
    for row, column in indices do
        winning_permutation.append(column)
    return tuple(winning_permutation)
```

Algorithm 10: Utility and Dictionary Selection Helpers

Function `getUtilityByAllocation`(*alloc, util, fun*)**:**
    $f \leftarrow fun.function$
    arg $\leftarrow fun.attribute$
    `// Calculate utility based on the sum of base values and the function type`
    return $f(\sum_{i \in alloc} util[i], \text{arg})$
Function `MaxFromDictOfLists`(d)**:**
    $max \leftarrow 0$
    $max\_i \leftarrow -1$
    for k ***in*** d **do**
        if $\sum(d[k]) > max$ **then**
            $max \leftarrow \sum(d[k])$
            $max\_i \leftarrow k$
    return $max\_i, d[max\_i]$

Algorithm 11: Generation of All Possible Partitions for n Players

Function `getAllAllocations`(m, n)**:**
    $obj \leftarrow \{0 \dots m-1\}$
    $pset \leftarrow$ `powerSet`(obj)
    `// Create a Cartesian product of powersets for each player`
    $candidate\_space \leftarrow$ `product`($pset, repeat = n$)
    $trimmed\_res \leftarrow []$
    for ***each*** *item* ***in*** *candidate_space* **do**
        merged $\leftarrow$ flatten and sort all indices in *item*
        `// Ensure the allocation is a valid partition of the set A`
        if `length` *(merged)* == m ***and*** *merged* == *obj* **then**
            $trimmed\_res$.`append`($item$)
    return *trimmed_res*

Bibliography

Amanatidis, G., G. Birmpas, G. Christodoulou, and E. Markakis (2020). Fair and efficient allocations under lexicographic preferences. In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 1774–1781.

Audet, C., P. Hansen, B. Jaumard, and G. Savard (2001). Enumeration of all extreme equilibria of bimatrix games. *SIAM Journal on Scientific Computing* 23(1), 323–338.

Avis, D., G. D. Rosenberg, R. Savani, and B. von Stengel (2010). Enumeration of Nash equilibria for two-player games. *Economic Theory* 42(1), 9–37.

Balthasar, A. (2010). Equilibrium tracing in strategic-form games. *Economic Theory* 42(1), 39–54.

Belhaiza, S. (2014). On perfect Nash equilibria of polymatrix games. *Game Theory* 2014.

Bertsimas, D., V. F. Farias, and N. Trichakis (2011). The price of fairness. *Operations Research* 59(1), 17–31.

Bogomolnaia, A. and H. Moulin (2025). Fair division with money and prices: bid & sell versus divide & choose. *Theory and Decision* 99(1), 407–443.

Bouveret, S. and M. Lemaître (2016). Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Autonomous Agents and Multi-Agent Systems* 30(2), 259–290.

Caragiannis, I., C. Kaklamanis, P. Kanellopoulos, and M. Kyropoulou (2009). *The Efficiency of Fair Division*, volume 50.

Chen, X. and X. Deng (2006). Settling the complexity of two-player Nash equilibrium. In: *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, 261–272. IEEE.

Chu, F. and J. Halpern (2001). On the NP-completeness of finding an optimal strategy in games with common payoffs. *International Journal of Game Theory* 30, 99–106.

Conitzer, V. and T. Sandholm (2008). New complexity results about Nash equilibria. *Games and Economic Behavior* 63(2), 621–641.

Cottle, R. W., J.-S. Pang, and R. E. Stone (2009). *The Linear Complementarity Problem*. Society for Industrial and Applied Mathematics.

Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ.

Deligkas, A., J. Fearnley, T. Igwe, and R. Savani (2016*a*). An empirical study of algorithms for solving polymatrix games. In: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*, 945–953.

Deligkas, A., J. Fearnley, T. Igwe, and R. Savani (2016*b*). Polymatrix-generators: A library of polymatrix game generators. https://github.com/polymatrix-games/polymatrix-generators.

Edmonds, J. (1967). Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Standards Sect. B* 71(4), 241–245.

Etessami, K. and M. Yannakakis (2010). On the complexity of Nash equilibria and other fixed points. *SIAM Journal on Computing* 39(6), 2531–2597.

Gal, Y., M. Mash, A. D. Procaccia, and Y. Zick (2017). Which is the fairest (rent division) of them all? In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17)*, 4841–4845.

Garey, M. R. and D. S. Johnson (1979). *Computers and Intractability*. W. H. Freeman, New York.

Gilboa, I. and E. Zemel (1989). Nash equilibria are hard to find. *Mathematics of Operations Research* 14(4), 749–753.

Goldman, J. and A. D. Procaccia (2014). Spliddit: Unleashing fair division into the real world. In: *Proceedings of the 15th ACM Conference on Economics and Computation (EC-14)*, 1001–1002.

Gottlob, G., G. Greco, and F. Scarcello (2005). Pure Nash equilibria: Hard and easy games. *Journal of Artificial Intelligence Research* 24(1), 357–406.

Govindan, S. and R. Wilson (2003). A global Newton method to compute Nash equilibria. *Journal of Economic Theory* 110(1), 65–86.

Govindan, S. and R. Wilson (2005). Essential equilibria. *Proceedings of the National Academy of Sciences* 102(43), 15706–15711.

Grötschel, M., L. Lovász, and A. Schrijver (1988). *Geometric algorithms and combinatorial optimization*. Springer.

Harsanyi, J. and R. Selten (1988). *A General Theory of Equilibrium Selection in Games*. MIT Press Classics. MIT Press.

Hauk, E. and S. Hurkens (2002). On forward induction and evolutionary and strategic stability. *Journal of Economic Theory* 106(1), 66–90.

Hillas, J. (1990). On the Definition of the Strategic Stability of Equilibria. *Econometrica* 58(6), 1365–1390.

Hillas, J. and E. Kohlberg (2002). Foundations of Strategic Equilibrium. In: *Handbook of Game Theory with Economic Applications*, edited by R. J. Aumann and S. Hart, volume 3, 1597–1663. Elsevier.

Howson, J. T., Jr. (1972). Equilibria of polymatrix games. *Management Science* 18(5, Part I), 312–318.

Kohlberg, E. and J.-F. Mertens (1986). On the strategic stability of equilibria. *Econometrica* 54(5), 1003–1037.

Kurokawa, D., A. D. Procaccia, and N. Shah (2018). Leximin allocations in the real world. *ACM Trans. Econ. Comput.* 6(3–4).

Kurokawa, D., A. D. Procaccia, and J. Wang (2018). Fair enough: Guaranteeing approximate maximin shares. *J. ACM* 65(2).

Lemke, C. E. (1965). Bimatrix equilibrium points and mathematical programming. *Management Science* 11(7), 681–689.

Lemke, C. E. and J. T. Howson (1964). Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics* 12(2), 413–423.

Levy, Y. J. (2016). Projections and functions of Nash equilibria. *International Journal of Game Theory* 45(1), 435–459.

Mallows, C. L. (1957). Non-null ranking models. I. *Biometrika* 44(1/2), 114–130.

Myerson, R. B. (1978). Refinements of the Nash equilibrium concept. *International Journal of Game Theory* 7(2), 73–80.

Papadimitriou, C. M. (1994). On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences* 48(3), 498–532.

Savani, R. and B. von Stengel (2006). Hard-to-solve bimatrix games. *Econometrica* 74(2), 397–429.

Selten, R. (1975). Reexamination of the perfectness concept for equilibrium points in extensive games. *International Journal of Game Theory* 4(1), 25–55.

Shah, N. (2017). Spliddit: Fair division of rent, goods, credit, fare, and tasks. *XRDS: Crossroads, The ACM Magazine for Students* 24(1), 42–45.

Simon, S. and K. R. Apt (2015). Social network games. In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15)*, 4240–4246. URL https://www.ijcai.org/Proceedings/15/Papers/593.pdf.

Suksompong, W. and J. Lang (2025). Constrained serial dictatorships can be fair. In: *Proceedings of the 34th International Joint Conference on Artificial Intelligence (IJCAI-25)*.

Tadenuma, K. and W. Thomson (1993). The fair allocation of an indivisible good when monetary compensations are possible. *Mathematical Social Sciences* 25(2), 117–132.

van Damme, E. (1989). Stable equilibria and forward induction. *Journal of Economic Theory* 48(2), 476–496.

van Damme, E. (1991). *Stability and Perfection of Nash Equilibria*. Springer-Verlag, Berlin Heidelberg, 2nd edition.

van den Elzen, A. and D. Talman (1999). An algorithmic approach toward the tracing procedure for bi-matrix games. *Games and Economic Behavior* 28(1), 130–145.

von Stengel, B. (2007). Equilibrium computation for two-player games in strategic and extensive form. In: *Algorithmic Game Theory*, edited by N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, 53–78. Cambridge University Press.

von Stengel, B. (2022). *Game Theory Basics*. Cambridge University Press.

von Stengel, B. (2023). Path-following for bimatrix games using Lemke's algorithm. Course Notes/Draft, accessed at https://github.com/stengel/EquiLearn/blob/main/TeX/using-lemke.pdf.

von Stengel, B. and F. Forges (2008). Extensive-form correlated equilibrium: Definition and computational complexity. *Mathematics of Operations Research* 33(4), 1002–1022.

von Stengel, B., A. van den Elzen, and D. Talman (2002). Computing normal form perfect equilibria for extensive two-person games. *Econometrica* 70(2), 693–715.

Yanovskaya, E. B. (1968). Equilibrium points in polymatrix games (in Russian). *Litovskii Matematicheskii Sbornik (Lithuanian Mathematical Collection)* 8(2), 381–384.