Factoring Games to Isolate Strategic Interactions

George B. Davis, Michael Benisch,
Kathleen M. Carley, and Norman M. Sadeh

January 2007
CMU-ISRI-06-121R[a]

[a]This report is a revised version of CMU-ISRI-06-121 and supercedes that report.

Institute for Software Research International
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work supported by National Science Foundation ITR grant 0205435, IGERT grant 9972762, as well as Office of Naval Research grant N00014-02-1-0973. Additional support at Carnegie Mellon University was provided by the Center for Computational Analysis of Social and Organizational Systems, and the e-Supply Chain Management Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the National Science Foundation, the Office of Naval Research, or the U.S. government.

Keywords: Strategically Aware Technology, Strategic Network Formation, Computational Game Theory

Abstract

Game theoretic reasoning about multi-agent systems has been made more tractable by algorithms that exploit various types of independence in agents' utilities. However, previous work has assumed that a game's precise independence structure is given in advance. This paper addresses the problem of finding independence structure in a general form game when it is not known ahead of time, or of finding an approximation when full independence does not exist. We give an expected polynomial time algorithm to divide a bounded-payout normal form game into factor games that isolate independent strategic interactions. We also show that the best *approximate* factoring can be found in polynomial time for a specific interaction that is not fully independent. Once known, factors aide computation of game theoretic solution concepts, including Nash equilibria (or ϵ-equilibria for approximate factors), in some cases guaranteeing polynomial complexity where previous bounds were exponential.

1 Introduction

In a system of self-interested agents (i.e. a game), an agent can use game theoretic solution concepts to reason about the dynamics of its environment. However, it is often too computationally expensive to apply a standard solution concept (such as Nash equilibrium) to a standard problem representation (such as the normal form). One response to this has been to identify structures in player utility functions that make solving games easier, and to represent games in compact forms which specify those structures.

The prototypical compact form is the extensive form (cf. [12]), which encodes sequential actions in games. Another example is the graphical model [8], which compacts games where player actions affect utilities only of selected "neighbors". Local effect games [11] and action graph games [3] capture symmetries between players (shared actions) and conditional interactions (where actions affect only the outcomes of players of certain other actions). In addition to helping agents select strategies, more compact representations of complex games allow modelers to reason about larger social systems, and help mechanism designers to design scalable systems with guaranteed properties.

In this paper, we take a new perspective on compact forms by considering an agent presented with a game where useful structure may exist, but is unknown. We model this by searching a normal form game for a specific structure whereby independent strategic interactions are embedded in a larger game. Our structure, once identified, can greatly accelerate computation of solution concepts in some games – and, as we will show, has the advantage that it can be searched for efficiently and even approximated for interactions that are not completely independent.

As an illustration of our structure, we describe a stylized model of airline competition where revenue is derived only from direct flights, and split between local competitors. Airlines in this scenario can choose whether to service a particular route without considering other routes, and still play optimally. To formalize this independence structure we introduce in Section 2 an algebra over NFGs which allows games with independent interactions to be reinterpreted as the *product* of independent *factor* games. As we show in Section 3, factoring a game under our algebra produces a more compact representation (in some cases exponentially so), and the factors of a game support a "divide and conquer" approach to speed the finding of Nash equilibria.

Factors in our representation correspond to connected components in a previous graphical representation for games, the Multi-Agent Influence Diagrams (MAIDs) of Koller and Milch [10]. MAIDs also capture structure in probabilistic events and directed independence (where one decision is independent of another but not vice-versa), but are more complex and have not been investigated from our perspective of search and approximation. Using our algebraic representation and restricting our model to complete independence enables our two key contributions.

Most previous compact forms can naively represent any game (e.g. a fully connected graphical game or action graph), but it is generally assumed that a *compact* game representation is known prior to computation. We address the problem of finding a best compact representation programatically by giving an algorithm in Section 4 whereby an agent or analyst presented with an arbitrary

game in its normal form can find all of its factors. By leveraging factoring techniques from computational algebra, our algorithm runs in expected time polynomial in the game size when the largest difference in the integer forms of any one player's utilities is bounded[1], even as the number of players and actions grows.

We take the Normal Form Game (NFG) as the object of our algorithms and analysis because its exhaustive enumeration of each player's utility from every combination of pure strategies most fundamentally captures the notion of a game with unknown structure. The normal form has worst-case representation length exponential in the number of players, making it unlikely that large multi-agent systems will represented this way. However, the normal form remains an important theoretical tool because results and techniques for the normal form can frequently be translated to other representations. There is also the potential for direct application of structural search on games learned empirically in their normal form, such as those described in [17].

Our factoring algorithm contributes to another body of research on preprocessing games to aide future computation and provide better worst-case bounds. Other examples include iterated dominance [9], generalized eliminability [6], and CURB set preprocessing [2]. Games reduced by these techniques typically omit information unnecessary for Nash equilibrium search. A factoring, by contrast, contains all information in the original game. As a preprocessor, our factoring algorithm gives a new class of games for which Nash equilibria can be computed in polynomial time (those with a constant maximum factor size). Cases with tighter bounds on Nash equilibrium search are of interest as the problem of finding Nash equilibria in general has recently been shown to be $\mathcal{PPAD}$-Complete [5], even in two player games with binary payouts [1] (and the fastest known algorithms are worst case exponential in the size of a game, c.f. [14]).

To our knowledge, none of the compact forms and only one of the preprocessors [7][2] have approximate counterparts for games lacking the precise structure they require. From our structural search perspective, it may be unrealistic to assume that an exact structure exists in the multi-agent system being modeled. To address this, we define in Section 5 an approximation on our algebras, ϵ-factoring, for games with *nearly* independent strategic interactions. As an example, we extend our direct flight game to include limited revenue for indirect flights. We show that ϵ-factors can be used to compute an ϵ-Nash equilibrium of their approximated game. Cartwright [4] has shown that agents with human-like social behaviors will reach such an equilibrium, and solution concepts with relaxation parameters have been more generally advocated for modeling agent behavior in realistic settings [16].

We give a polynomial time algorithm for finding the ϵ-factoring that isolates a specific interaction with minimum ϵ. This allows an agent, as input, to propose a model proposing decisions for players to make independently. The ϵ of the returned factoring captures the stability of this

[1]Since we convert rational payouts to non-negative integers by linear transformation, this can be viewed as a bound on the precision of player utilities.

[2]Conitzer and Sandholm describe a preprocessor for games where actions can be ordered according to their resulting utilities. They provide an approximate version for games in which some actions break the ordering by a bounded difference in utilities.

model, in terms of incentives for all players to consider the full game instead. Additionally, the returned ϵ-factoring incorporates the best possible approximation of strategic interactions between decisions mistreated as independent.

Motivating Example

To motivate the structure we will discuss in this paper, we present the Direct Flight Game (DFG). Players in a DFG represent competing airlines that must simultaneously choose subsets of direct routes between cities to service. Parameters of a DFG specify, for each route, the cost of servicing the route and the total revenue it will generate. Any player servicing a route pays its cost, but competitors servicing the same route split its revenue. An example of a DFG with two routes is given graphically in Figure 1.

Figure 1: Graphical example of a two-route DFG.

Its full normal form representation for two players is presented in Figure 2 (top). Rows and columns represent pure strategies of each player and the tuple at their intersection lists the row and column utilities respectively. Note that when deciding whether or not to service a route, a player needs to consider only whether or not he or any of his opponents will service that particular route — all decisions regarding other routes do not affect the utility generated by this one. In this sense, the full DFG can be considered a collection of independent games, one per route, as shown in Figure 2 (bottom). Note that if we were to add an additional route, we could simply append an additional factor game, while the full NFG would quadruple in size.

Explained in this way, the normal form representation of the DFG registers as contrived. However, this is not immediately apparent from the normal form without the given context.

2 An Algebra of Games

In this section we define a game *product* operator, $\otimes$, and present associated vocabulary regarding the algebra it induces on games. The goal is to re-interpret a single game as the product of smaller games that can be played independently while capturing all outcomes and strategic interactions in the product game. We formally consider an n-player NFG, G^A, as consisting of the following.

- $\boldsymbol{S_i^A}$, the set of pure strategies associated with each player $\boldsymbol{i}$. (We denote pure strategies with $\boldsymbol{s}$ and mixtures as $\boldsymbol{m}$).
- $\boldsymbol{u_i^A : P \to \mathbb{Q}}$, player $\boldsymbol{i}$'s utility function, where $\boldsymbol{P}$ is a profile designating a strategy for each player, and the rational value returned is player $\boldsymbol{i}$'s utility under $\boldsymbol{P}$. If $\boldsymbol{P}$ includes mixed strategies, $\boldsymbol{u_i^A(P)}$ denotes player $\boldsymbol{i}$'s *expected* utility.

	Neither	LA-CH	CH-NY	Both
Neither	0,0	0,2	0,4	0,6
LA-CH	2,0	-1,-1	2,4	-1,3
CH-NY	4,0	4,2	-1,-1	-1,1
Both	6,0	3,-1	1,-1	-2,-2

	¬LA-CH	LA-CH
¬LA-CH	0,0	0,2
LA-CH	2,0	-1,-1

	¬CH-NY	CH-NY
¬CH-NY	0,0	0,4
CH-NY	4,0	-1,-1

Figure 2: *Top:* Two-player normal form representation of example DFG from Figure 1. *Bottom:* Independent factor games corresponding to each route.

Definition 1. *The* **product** *of two n-player normal form games G^A and G^B, denoted $G^A \otimes G^B$, is a new game G^C with the following properties.*

- *Player i's pure strategy set S_i^C equals $S_i^A \times S_i^B$. E.g., there is exactly one strategy $s_{ab} \in S_i^C$ corresponding to every pair $s_a \in S_i^A$, $s_b \in S_i^B$.*
- *Given* pure *strategy profile P^C for G^C, where profiles P^A and P^B contain corresponding pure strategies in each factor game, $\forall_i u_i^C(P^C) = u_i^A(P^A) + u_i^B(P^B)$.*

Definition 2. *Let G^C be an n-player NFG. Two n-player NFGs, G^A and G^B, are* **factors** *of G^C if $G^A \otimes G^B = G^C$.*

The bottom segment of Figure 2 gives factor games whose product is our example DFG. Each factor game corresponds to a particular route, and pure strategies in the factor games correspond to servicing or not servicing that route. Each outcome in the full product game is the element-wise sum of a pair of outcome tuples, one from each factor game. For example, the outcome in the full NFG when the column player services only LA-CH and the row player services only CH-NY is, $(4, 2) = (0, 2) + (4, 0)$.

We can see that the ordering of the factors does not matter in the construction of the full game. We can also add a route to our example DFG by taking the product of the full game with a game corresponding to only the new route. This illustrates the following properties of the game product operation.

Proposition 1. *$\otimes$ is* **commutative** *($G^A \otimes G^B = G^B \otimes G^A$) and* **associative** *($(G^A \otimes G^B) \otimes G^C = G^A \otimes (G^B \otimes G^C)$)*[3].

We define game product's identity set, I_n, as the set of n-player games with a single action for each player and some constant c_i as player i's utility from the single outcome. Taking the product $G \otimes I_n$ merely shifts payouts in G, which does not affect the relative attractiveness of any actions under any contingency. For simplicity, we will discuss only normalized games with each player's

[3]Omitted proofs are available in the appendix.

smallest utility equal to 0, and use I_n to refer to the identity element giving each player utility $c_i = 0$.

Definition 3. *An n-player game G is* **irreducible** *with respect to game product if I_n and G are its only factors.*

Definition 4. *A* **factoring** *for G^A is a collection of games F such that $\prod^{\otimes}_{G^k \in F} G^k = G^A$. If all games in F are irreducible, then F is a* **total factoring** *of G^A.*

The following vocabulary is used throughout this paper to relate objects associated with a product game to those associated with its factors.

If $G^C = G^A \otimes G^B$, then a pure strategy in G^C is called a ***product strategy*** and can be denoted $s_{ab} \in S^C_i$ if it corresponds to ***factor strategies*** $s_a \in S^A_i$ and $s_b \in S^B_i$.

We denote a mixed strategy of player i in the product game m^C_i. The ***projection strategy*** of m^C_i onto G^A, denoted $\vec{m}^A_i$, is the mixed strategy in the factor game with weight on each pure strategy equal to the sum of weights in m^C_i on its products. For example, if m^C_r is a profile for the row player of our DFG example with weights $[.1, .2, .3, 4]$, then its projections are $\vec{m}^A_r = [.4, .6]$ and $\vec{m}^B_r = [.3, .7]$.

There is a special class of mixtures where the probability on each product strategy is the product of the probabilities on its factor strategies in its projections. We call such a mixture the ***product mixture*** of its projections. For example, the product of projections $\vec{m}^A_r$ and $\vec{m}^B_r$ above is a new row mixture for G^C, $[.4 \times .3, .6 \times .3, .4 \times .7, .6 \times .7] = [.12, .18, .28, .42]$. Note that some mixtures, including m^C_r defined above, cannot be described as the product of any mixtures in the factor games.

This notation extends to profiles as well; the ***projection profile*** of P^C onto G^A is denoted $\vec{P}^A$, and contains projections of each player's strategy. Similarly, P^C is the ***product profile*** of ***factor profiles*** P^A and P^B if each player's strategy in P^C is the product of that player's strategies in P^A and P^B.

3 Reasoning with Factors

One benefit of our algebra is that game factors are games in the traditional sense, and all solution concepts and algorithms for the normal form can be applied to them. To demonstrate that strategic invariants of a product game are captured by its factor games, we show how a variety of solution concepts can be transferred directly from factor games to their product. Specifically, we show that the dominated and rationalizable strategies, and Nash equilibrium profiles in factor games describe the only corresponding entities in their product.

First we consider the relationship between strategies that are dominated in factor and product games. Dominance plays an important role in guarantees of truthful behavior for many designed mechanisms. Strategy s_i for player i is strictly dominated if there is another strategy s'_i which

is always preferable; i.e., there exists a strategy s_i' such that for all profiles P_{-i} over opponent strategies $u_i(P_{-i} \cup \{s_i\}) < u_i(P_{-i} \cup \{s_i'\})$. (For the notion of weak dominance the inequality need only be strict for at least one opponent profile.)

Theorem 1. *A product strategy is* **dominated** *iff at least one of its factor strategies is dominated.*

We now provide a similar result for the concept of rationalizability. A strategy for player i is rationalizable if it is i's *best response* against some rationalizable opponent strategy profile.

Theorem 2. *A product strategy is* **rationalizable** *iff all of its factor strategies are rationalizable in their respective factor games.*

Finally, we consider the most important point valued solution concept, the Nash equilibrium. A strategy profile is in Nash equilibrium if no agent can benefit from unilaterally deviating. The following results show that Nash equilibria in factor games describe the only equilibria in their product games.

Lemma 1. *If a product profile is in* **Nash equilibrium***, then its projection onto any factor game is in Nash equilibrium.*

Sketch. Assume P^C is a Nash equilibrium profile in the product game, and its projection onto G^A, $\vec{P}^A$, is not in equilibrium. Then there must be some player i for which there exists a mixed strategy m_i' that is a better response than his projected strategy $m_i \in \vec{P}^A$. We let Δ_i^A be a change vector describing how to change m_i into m_i'. Using Δ_i^A we can build a similar change vector for i's mixture in the product game, Δ_i^C, which improves i's utility, violating the assumption that the original profile was in equilibrium. □

Theorem 3. *The product of a set of factor profiles is in Nash equilibrium in a product game iff each factor profile is in Nash equilibrium in its corresponding factor game.*

Sketch. A product profile in equilibrium must be the product of equilibria based on Lemma 1. The product of equilibrium profiles is an equilibrium in the product game because the best response structure is preserved by the product operator (i.e. the product of the best responses in each factor game must be a best response to their product profile). □

Note that Theorem 3 leaves open the possibility that there exist equilibrium profiles in a product game that cannot be described as the product of *any* set of factor profiles. (For example, in a game with all zero utilities, any profile is in equilibrium.) However, the following proposition confirms that products of factor equilibria still represent any such profiles.

Proposition 2. *If a Nash equilibrium profile in a product game cannot be represented as a product of factor profiles, then the profile is a* weak *Nash equilibrium (at least one player is indifferent about playing his equilibrium strategy). Furthermore, there exists an equilibrium profile which* is *the product of equilibria profiles in factor games, such that each player is indifferent between their strategies in the original and product profiles.*

Sketch. It can be shown that the product of any equilibria's projections is an equilibrium profile consisting of strategies to which players are indifferent. □

To summarize the results regarding Nash equilibria, we have shown that all Nash equilibrium profiles in a product game are either strict equilibria which are the product of equilibrium profiles in factor games, or else they are part of a collection of weak equilibria whose projections are Nash equilibria. In the latter case, the product of the projections constitutes a representative of the collection. Therefore we can describe all Nash equilibria in a product game, by generating profiles that are the products of every combination of equilibria in its factor games.

The number of Nash equilibria in a product game can potentially be infinite, but we can produce a single Nash equilibrium simply by finding one in each factor. We therefore derive the following two corollaries. We denote the complexity of finding a Nash equilibrium in a game with representation length x as $\texttt{NE}(\texttt{x})$.

Corollary 1. *A Nash equilibrium of a game with representation length n and a known factoring can be found in $O(n + \texttt{NE}(n'))$ time, where n' is the representation length of the largest factor.*

Corollary 2. *For any game with representation length n and constant maximum irreducible factor size, its totally factored representation (i) has length $O(\log(n))$, and (ii) allows a Nash equilibrium to computed in time $O(n)$.*

4 Factoring Games

Consider a scenario where factorable interactions are not immediately apparent, but the corresponding game is factorable. For example, the NFG given in Figure 2 could be an airline's empirical estimation of how profit is distributed under different competitive scenarios, without the airline knowing that profits from routes are independent. In this section we give the outline (abbreviated due to space) of an algorithm for finding factors of an arbitrary NFG in time polynomial in the size of the game and the largest difference in any one player's utilities. The two primary aspects of game factoring are i.) identifying the redundancy in utilities that allows them to be expressed as summations and ii.) maintaining the strategic interactions of the original game in the factors. Our algorithm involves two phases accomplishing these tasks.

4.1 Factoring Strategy Sets

In the first phase we describe each player's impact on all outcome utilities as the sum of the impacts of several distinct decisions, rather than a single choice between pure strategies. We identify for a player i the strategy sets $S_i^1, \ldots S_i^k$ with the *potential* to become i's factor strategy sets in a total factoring. We call these "potential" factor strategy sets because in this phase we consider one

player at a time and ignore interactions between players that could prevent decisions from being reasoned about independently.

We first represent each player i's strategy set S_i in the original game as a polynomial f_i encoding the utilities of all outcomes in which each strategy is played. Each term of f_i encodes the impact of a single strategy s as the product of placeholder variables, labeled $x_{(P_{-i},j)}$, corresponding to each profile of opponent pure strategies P_{-i} and player j. The exponent on each x in a term equals the utility for player j when s is played against P_{-i}. Utilities may be rational or negative, so we multiply them by a constant λ and offset them by C in order to make them valid polynomial exponents (i.e. non-negative integers). The coefficients of the polynomial are in the non-negative integers $\mathbb{N}$ (coefficients greater than one occur only in the case of redundant strategies).

$$f_i = \sum_{s \in S_i} \prod_{P_{-i}} \prod_{j} x_{(P_{-i},j)}^{\lambda u_j(P_{-i} \cup \{s\}) + C} \tag{1}$$

For example, in the DFG described in Figure 2 we represent the row player's strategy set as the following polynomial. Note that we have given arbitrary but consistent indices to each x. Exponents on variables with odd indices are payouts for the row player; even ones are for the column player.

$$\begin{aligned} f^V = x_1^2 x_2^2 x_3^2 x_4^4 x_5^2 x_6^6 x_7^2 x_8^8 \quad &+ \quad x_1^4 x_2^2 x_3^1 x_4^1 x_5^4 x_6^6 x_7^1 x_8^5 \quad + \\ x_1^6 x_2^2 x_3^6 x_4^4 x_5^1 x_6^1 x_7^1 x_8^3 \quad &+ \quad x_1^8 x_2^2 x_3^5 x_4^1 x_5^3 x_6^1 x_7^0 x_8^0 \end{aligned}$$

Observe that the product of two polynomials constructed in this way is a new polynomial where the exponents on each term are the sums of exponents from a pair of terms in the factor polynomials. In this way the product of two such polynomials represents a strategy set with outcome utilities as defined by our game product operator. Conversely, factoring any f_i into the product of lower degree polynomials is equivalent to finding i's potential factor strategy sets, $S_i^1, \ldots S_i^k$.

4.2 Preserving Strategic Invariants

Each potential factor strategy set can be thought of as a separate decision made by a player regarding how to impact the game. However, these decisions may not all be made in isolation — some of them *strategically interact*[4], and must be reasoned about together as part of the same factor game. Intuitively, a potential factor strategy set S_i^A strategically interacts with S_j^B if player i's choice from S_i^A affects the relative impact on some player of j's choices from S_j^B.

[4]Note that a directed version of the same concept is given for MAIDs as *strategic relevance*.

Definition 5. *Observe that we can describe any pure strategy profile P in the product game as the product of choices from each potential factor strategy set for all players. Given $s \in S_i^A$ and $s' \in S_j^B$, let $P_{(s,s')}$ denote the profile identical to P except for i's choice of s and j's choice of s'. S_i^A* **strategically interacts** *with S_j^B if there exists profile P, potential factor strategies $s_a, s'_a \in S_i^A$ and $s_b, s'_b \in S_i^B$, and player k such that k's difference in utility in the product game when j switches from s_b to s'_b is not the same for both of i's choices, s_a and s'_a,*

$$u_k(P_{(s_a,s_b)}) - u_k(P_{(s_a,s'_b)}) \neq u_k(P_{(s'_a,s_b)}) - u_k(P_{(s'_a,s'_b)})$$

To find groups of strategy sets that *can* be isolated, we compute the transitive closure of the strategic interaction relation. Groups of potential factor strategy sets under this closure are minimal collections such that no members from different groups interact. Each such group corresponds to one factor game in a total factoring of the original.

To generate a factor game from a group, we first construct a factor strategy set for each player by taking the Cartesian product of that player's potential factor strategy sets in the group. (If a group has no sets for a player, we give that player one dummy strategy with no impact on the game). We then calculate utilities of the factor game by holding decisions outside that factor constant and examining the differences in utility granted to each player when different combinations of strategies within the factor are selected.

4.3 Complexity of Game Factoring

To factor the multivariate polynomial over $\mathbb{N}$ corresponding to each player's strategy set we adapt a method from [18]. We first use fast algorithms to factor the polynomial over the integers $\mathbb{Z}$, which might result in factors with negative coefficients. Then we try all ways of multiplying factors to find a factoring with all positive coefficients. The fastest algorithms for factoring a sparse-representation multivariate polynomial over $\mathbb{Z}$ (e.g. [15]) have worst case expected complexity polynomial in representation size when degree is bounded. In our representation, this bound on degree corresponds to a bound on the maximum difference in the integer form of any one player's payouts. Since the size of the polynomial we construct is the same as the size of the game, and the number of factors is bounded by the degree, we can both factor the polynomial and multiply all combinations of factors in expected polynomial time.

Interestingly, the semi-ring of polynomials over $\mathbb{N}$ is not a *unique factorization domain* (UFD), meaning that there are polynomials over $\mathbb{N}$ with multiple unique total factorizations (c.f. [18]) and consequently strategy sets that can be factored multiple ways. This result can be extended to apply more widely to our algebra, showing that there exist games with multiple total factorings. Fortunately, the method above for finding a positive factoring of a polynomial can enumerate *all* such factorings with the same worst-case polynomial time complexity. Thus, we can, with no greater difficulty, enumerate all potential factorings for each player's strategies.

Given multiple ways to factor the strategy sets of each player, we may have to construct a different total factoring of the game for every combination. However, in a game with bounded

payouts there is a constant bound on the number of ways to factor one of our polynomials, and therefore a constant bound on the number of total factorings. Additionally, we can compute the transitive closure of strategic interaction in polynomial time, giving us the following proposition.

Proposition 3. *Our algorithm finds all total factorings of an NFG with bounded payouts and representation length n in expected time worst-case polynomial in n.*

In practice, we can factor a 2-player DFG with 10 routes (2^{10} strategies per player) in about 1 minute on a 3Ghz Pentium 4 with 1GB of RAM.

5 Approximate Game Factoring

There are many reasons an agent might mistreat decisions that strategically interact as independent. For example, due to an incorrect model, or computational resources too limited to reason about the decisions simultaneously. In this section, we use our algebra to examine at what cost a decision that is only *nearly* independent can be made in isolation. We define an ϵ-factoring of a game as a collection of factors whose product is nearly (its outcomes are within ϵ of) the game.

Definition 6. *The collection of games, F, with product G^B, is an ϵ-**factoring** of G^A if every pure strategy in G^A corresponds to a distinct pure strategy in G^B such that for any pure strategy profile P^A in G^A, its corresponding profile P^B in G^B, and each player i, $\mid u_i^A(P^A) - u_i^B(P^B) \mid \leq \epsilon$.*

We can extend our results regarding Nash equilibria to ϵ-factoring by considering the notion of an ϵ-Nash equilibrium. A strategy profile is said to be in ϵ-Nash equilibrium if no agent can benefit by more than ϵ from unilaterally deviating. It has been shown that agents with sufficient behaviors of imitation and innovation will learn to play such an equilibrium [4].

Theorem 4. *The product of Nash equilibrium profiles in an ϵ-factoring of a game is an ϵ-Nash equilibrium of the game.*

To motivate this application of our algebra, we present the Indirect Flight Game (IDFG), which extends the DFG to provide bounded revenue from indirect (multi-hop) flights. An additional parameter specifies revenue to be split between carriers servicing some specific indirect paths. The NFG in Figure 3 presents an IDFG variation of our original DFG example, where an additional revenue of 2 is available to providers of indirect service along the path NY-CH-LA.

The IDFG example can no longer be factored along each direct route because players receive the additional indirect revenue by a combination of decisions regarding the two. However, since the game is very similar to the DFG, it is still *nearly* factorable. In fact, the original factoring given for the DFG in Figure 2 (bottom) is an ϵ-factoring of the IDFG with $\epsilon = 2$. Interestingly, another ϵ-factoring of the example IDFG, shown in Figure 4, achieves $\epsilon = \frac{1}{2}$.

The naive ϵ-factoring of our IDFG example simply ignores the additional utility from indirect demand. The ϵ-factoring in Figure 4 incorporates in its utilities an implicit estimation of the impact

	Neither	LA-CH	CH-NY	Both
Neither	0,0	0,2	0,4	0,8
LA-CH	2,0	-1,-1	2,4	-1,5
CH-NY	4,0	4,2	-1,-1	-1,3
Both	8,0	5,-1	3,-1	-1,-1

Figure 3: IDFG example with revenue of 2 available for indirect service between NY-LA via CH.

	¬LA-CH	LA-CH
¬LA-CH	$-\frac{1}{4}, -\frac{1}{4}$	$-\frac{1}{4}, 2\frac{3}{4}$
LA-CH	$2\frac{3}{4}, -\frac{1}{4}$	$-\frac{1}{4}, -\frac{1}{4}$

	¬CH-NY	CH-NY
¬CH-NY	$-\frac{1}{4}, -\frac{1}{4}$	$-\frac{1}{4}, 4\frac{3}{4}$
CH-NY	$4\frac{3}{4}, -\frac{1}{4}$	$-\frac{1}{4}, -\frac{1}{4}$

Figure 4: Minimal ϵ-factoring of IDFG example in Figure 3.

of strategic interactions across factors. This is illustrated by comparing the two factorings in terms of the difference in utility between servicing and not servicing a route. In the lower ϵ-factoring, this difference has been increased in anticipation of the bonus for servicing both routes. The best such estimation that isolates a particular strategic interaction is the ϵ-factoring along that interaction with minimal ϵ. It can be found by a semi-naive algorithm that, given the strategy sets (but not utility functions) of two ϵ-factors, assigns optimal outcome utilities. (The factoring in Figure 4 was computed in this way.)

Our algorithm is based on the fact that two outcome utilities (those with largest positive and negative errors) determine the ϵ in any minimal ϵ-factoring. For each potential such pair, it solves a linear program (LP) to assign factor utilities that minimize ϵ when that pair defines it. From the solutions to all LPs it selects the one yielding minimum ϵ. More precisely, for each player i and pair of distinct pure strategy profiles in the product game, P and P', we construct the following linear program.

$$\min_{u_i^A(\cdot), u_i^B(\cdot)} \quad u_i^A(\vec{P}^A) + u_i^B(\vec{P}^B) - u_i^C(P)$$

subject to,

1. $u_i^A(\vec{P}^A) + u_i^B(\vec{P}^B) - u_i^C(P) \quad = \quad -\left(u_i^A(\vec{P'}^A) + u_i^B(\vec{P'}^B) - u_i^C(P')\right)$
2. $\left(\forall_{P^A, P^B}\right) \quad u_i^A(P^A) + u_i^B(P^B) - u_i^C(P^A \otimes P^B) \quad \leq \quad u_i^A(\vec{P}^A) + u_i^B(\vec{P}^B) - u_i^C(P)$
3. $\left(\forall_{P^A, P^B}\right) \quad u_i^A(P^A) + u_i^B(P^B) - u_i^C(P^A \otimes P^B) \quad \geq \quad u_i^A(\vec{P'}^A) + u_i^B(\vec{P'}^B) - u_i^C(P')$

The LP optimizes $u_i^A(\cdot)$ and $u_i^B(\cdot)$, player utilities for every profile in each approximate factor game. The objective is to minimize ϵ, defined as the amount i's utility in the product of the

approximate factors exceeds i's utility when P is played in the original game. The first constraint ensures that the maximum positive error when P is played is equal to the maximum negative error when P' is played. This is essential to having minimum ϵ in this iteration, since error from utilities under P and P' are assumed to define ϵ from below and above respectively. The second set of constraints ensures that, for each pair of approximate factor utilities, their sum underestimates the corresponding utility in the original game by at most ϵ. The third set symmetrically assures that they do not overestimate by more than ϵ.

The approximate factoring with each player's utility assigned this way has ϵ equal to the maximum ϵ of the solution for any one player.

Proposition 4. *Our algorithm finds an ϵ-factoring that isolates a specific decision with minimal ϵ in time polynomial in the representation of the game.*

The ϵ in the returned factoring can be considered the cost — in terms of model accuracy, or the stability of equilibria it is used to compute — of reasoning about that interaction separately from the rest of the original game. An agent or mechanism designer can use this technique to propose a factoring as an approximate compact form and compute the incentive for any player to consider the actual game instead.

6 Conclusions and Future Research

We presented an algebra that represents some complex games as the product of factor games with independent strategic interactions. A game factoring can be considered a compact form and, for a variety of solution concepts, reasoning about factors suffices to analyze their product (but is potentially much faster). We outlined an algorithm for finding all irreducible factors of arbitrary games with bounded payouts, with expected time polynomial in their size. Our algorithm is built on fast computational algebra methods for factoring sparse-representation polynomials and computing transitive closures. We tested the runtime of our algorithm on a specific class of games; additional experiments are needed to assess runtime on irreducible games and games with differently shaped factors.

To characterize strategic interactions that are *nearly* independent, we defined an approximation on our algebra, ϵ-factoring. We showed that ϵ-equilibria of a game can be found by finding Nash equilibria of its ϵ-factors. The existence of an approximate reduced form begs the question, how good is a particular approximate representation? We showed that calculating the minimal ϵ-factoring with a particular structure can answer this in terms of the incentive for any player to consider the full game instead. We outlined a polynomial time algorithm for finding the minimal ϵ-factoring that isolates a specific interaction. The resulting approximation incorporates a best-possible estimation of ignored strategic interactions.

The most immediate application of our algorithms is in factoring and approximately factoring games with unknown structure, such as those learned empirically. For games with known structure

our algebraic representation and theoretical results can potentially be used in their analysis.

We considered computing an ϵ-factoring only when a factor structure was given as input. An algorithm without this parameter, which searches for ϵ-factorings that compact a game while maintaining low ϵ, could allow fully automatic detection of tractable models with approximate guarantees. This can be done naively by enumerating all possible factorings and applying our linear programming technique. However, this would be impractical in most cases, suggesting the need for a better search strategy.

Search and approximate search algorithms for game factoring also serve as preprocessors for other types of analysis, and this approach could be applied to other compact forms as well. It is also worth exploring which structural concepts underlying various compact forms are non-exclusive and could be used in conjunction. For example, both our direct flight game and its factors can be represented as congestion games [13].

Our game product operator is defined in terms of summed factor utilities. A natural extension to our algebra would be to consider other operations on factor utilities, such as multiplication, max, or arbitrary functions. Another area for future work involves examining algorithms with worst-case complexity independent of degree for factoring sparse integer polynomials over $\mathbb{N}$, since this forms the inner loop of our game factoring algorithm. This problem has been studied for polynomials on $\mathbb{Z}$ (and is believed to be co-$\mathcal{NP}$-Hard), but not with the additional constraints imposed on polynomial form by our game theoretic setting. Games with additional special properties (including other compact form structures) may be even easier to factor.

References

[1] T. Abbott, D. Kane, and P. Valiant. On the complexity of two-player win-lose games. In *Proceedings of FOCS'05*, 2005.

[2] M. Benisch, G. B. Davis, and T. Sandholm. Algorithms for rationalizability and curb sets. In *Proceedings of AAAI'06*, 2006.

[3] N. Bhat and K. Leyton-Brown. Computing nash equilibria of action-graph games. In *Proceedings of UAI'04*, 2004.

[4] E. Cartwright. Learning to play approximate nash equilibria in games with many players. In *FEEM, Working Paper No. 85.04*, 2004.

[5] X. Chen and X. Deng. Settling the complexity of 2-player Nash-equilibrium. Technical Report TR05-140, Electronic Colloquium on Computational Complexity, 2005.

[6] V. Conitzer and T. Sandholm. A generalized strategy eliminability criterion and computational methods for applying it. In *Proceedings of AAAI'05*, 2005.

[7] V. Conitzer and T. Sandholm. A technique for reducing normal form games to compute a nash equilibrium. In *Proceedings of AAMAS'06*, 2006.

[8] M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. In *Proceedings of UAI'01*, 2001.

[9] D. E. Knuth, C. H. Papadimitriou, and J. N. Tsitsiklis. A note on strategy elimination in bimatrix games. *Operations Research Letters*, 7(3):103–107, 1988.

[10] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1):181–221, 2003.

[11] K. Leyton-Brown and M. Tennenholtz. Local-effect games. In *Proceedings of IJCAI'03*, 2003.

[12] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1999.

[13] R. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[14] R. Savani and B. von Stengel. Exponentially many steps for finding a nash equilibrium in a bimatrix game. *FOCS'04*, pages 258–267, 2004.

[15] J. von zur Gathen and E. Kaltofen. Factoring sparse multivariate polynomials. *J. Comput. Syst. Sci.*, 31(2):265–287, 1985.

[16] Y. Vorobeychik and M. P. Wellman. Mechanism design based on beliefs about responsive play (position paper). In *Proceedings of EC'06 Workshop on Alternative Solution Concepts for Mechanism Design*, 2006.

[17] M. P. Wellman, D. M. Reeves, K. M. Lochner, and R. Suri. Searching for walverine. In *Proceedings of IJCAI'05 Workshop on Trading Agent Design and Analysis*, 2005.

[18] L. Zhi and Z. Liu. P-irreducibility of binding polynomials. *Computers and Mathematics with Applications*, 38:1–10, 1999.

7 Appendix

Proposition 1. $\otimes$ *is* **commutative** *(*$G^A \otimes G^B = G^B \otimes G^A$*) and* **associative** *(*$(G^A \otimes G^B) \otimes G^C = G^A \otimes (G^B \otimes G^C)$*).*

Proof. This follows trivially from the fact that addition has these properties, and is the underlying function of the $\otimes$ operator. □

Theorem 1. *A product strategy is* **dominated** *iff at least one of its factor strategies is dominated.*

Proof. For clarity, the following proof considers only strict dominance and a factoring of only two factors. The result can be trivially chained to account for any number of factors and extended to weak dominance. Let s_{ab} be a strategy in game G^C with factor strategies s_a and s_b in factor games G^A and G^B respectively. If s_a is dominated by some strategy $s_{a'}$ in G^A then s_{ab} must be dominated by $s_{a'b}$ since the utilities of s_{ab} and $s_{a'b}$ differ only in their contributions from outcomes in G^A. The same reasoning applies to the case where s_b is dominated by some $s_{b'}$. This proves the forward implication of the theorem that s_{ab} is dominated if s_a or s_b is dominated.

Now we will prove the backwards implication, namely that if the strategy s_{ab} is dominated in G^C then s_a or s_b must be dominated in its respective game. Assume that neither s_a or s_b is dominated in its respective game, but s_{ab} is dominated in G^C. The fact that neither factor strategy is dominated means that there exists some profile of opponent pure strategies, P^A, for which s_a is better than some other strategy in G^A, $s_{a'}$, and likewise for s_b against P^B compared to $s_{b'}$. Based on this reasoning, the strategy s_{ab} must be better than the strategy $s_{a'b'}$ in G^C against the opponent profile P^C which is the product of profiles P^A and P^B. Thus s_{ab} is not dominated in G^C which contradicts our assumption. □

Theorem 2. *A product strategy is* **rationalizable** *iff all of its factor strategies are rationalizable in their respective factor games.*

Proof. We will provide reasoning that applies to games with two factors, which can be trivially chained to account for any number of factors. Consider a strategy s_{ab} for player i in a game G^C which is the product strategy of s_a in G^A and s_b in G^B. Assume that both s_a and s_b are rationalizable in both of their respective games, then there exist rationalizable profiles in G^A and G^B for the players other than i, P^A_{-i} and P^B_{-i}, such that s_a and s_b are the best responses of i against P^A_{-i} and P^B_{-i} respectively. It must be the case that s_{ab} is also a best response to the product of P^A_{-i} and P^B_{-i} in G^C, which ensures that s_{ab} is also rationalizable. This argument can easily be extended to all of the other players in the game, which proves the forward implication of the theorem.

To prove the backward implication of the theorem we assume that a strategy s_{ab} is rationalizable, but (wlog) s_a, its factor strategy from game G^A, is not. This implies that there exists a rationalizable strategy profile P^C to which s_{ab} is a best response, but by our assumption s_a cannot be the best response to the projection of P^C onto G^A, $\vec{P}^A$. Thus there is some other strategy $s_{a'}$ that provides more utility than s_a against $\vec{P}^A$, and consequently $s_{a'b}$ must provide more utility against P^C than s_{ab} (since s_{ab} and $s_{a'b}$ differ only in their contribution from the G^A factor game). This leads to a contradiction with the fact that s_{ab} was a best response to P^C (which followed directly from our assumption that s_{ab} was rationalizable). □

Lemma 1. *If a product profile is in* **Nash equilibrium***, then its projection onto any factor game is in Nash equilibrium.*

Proof. Assume the converse, that P^C is a Nash equilibrium profile in the product game, and its projection onto G^A, $\vec{P}^A$, is not in equilibrium. Then there must be some player i for which there exists a mixed strategy m_i' that is a better response than his projected strategy $m_i \in \vec{P}^A$. We let Δ_i^A be a change vector describing how to change m_i into m_i'. Using Δ_i^A we build a similar change vector for the full game, Δ_i^C, that improves i's utility in the product game, violating the assumption that the original profile was in equilibrium. We use Δ_i^A to improve i's utility in G^C by adjusting profile P^C in the following way.

$$p_{ab} \in P_i^C = p_{ab} + \frac{\Delta^A p_{ab}}{\sum_{b'} p_{ab'}}$$

This adjustment to P^C guarantees that its projection onto G^B has not changed at all, and player i has received the additional benefit from Δ^A which contradicts our assumption that P^C was in equilibrium. □

Theorem 3. *The product of a set of factor profiles is in Nash equilibrium in a product game iff each factor profile is in Nash equilibrium in its corresponding factor game.*

Proof. A product profile in equilibrium must be the product of equilibria based on Lemma 1. To prove that the product of equilibrium profiles is an equilibrium in the product game we will consider the case of a game with two factor profiles, P^A and P^B (this reasoning can be trivially chained to account for more than two factors).

By the definition of a Nash equilibrium we have the following equations.

$$\left(\forall i, s_a' \in S_i^A\right) \quad \sum_{s_a} p_{s_a} u_i(s_a, P_{-i}^A) \quad \geq \quad u_i(s_a', P_{-i}^A) \tag{2}$$

A symmetric equation holds for the equilibrium in B. By summing the two equations, we have

$$\left(\forall i, s_a' \in S_i^A, s_b' \in S_i^B\right) \quad \sum_{s_a} p_{s_a} u_i(s_a, P_{-i}^A) \quad +$$

$$\sum_{s_b} p_{s_b} u_i(s_b, P_{-i}^B) \quad \geq \quad u_i(s_a', P_{-i}^A) + u_i(s_b', P_{-i}^B) \tag{3}$$

Since each profile must be a valid probability distribution, and consequently $\sum_{p_{s_a}} = \sum_{p_{s_b}} = 1$, all of the inequalities above can be re-written in the following way.

$$\sum_{s_a} \sum_{s_b} p_{s_a} p_{s_b} \left(u_i(s_a, P_{-i}^A) + u_i(s_b, P_{-i}^B)\right) \quad \geq$$

$$u_i(s_a', P_{-i}^A) + u_i(s_b', P_{-i}^B) \tag{4}$$

This proves that the product projection of P^A and P^B, $P^C = \{p_{s_a}p_{s_b} \mid p_{s_a} \in P^A, p_{s_b} \in P^B\}$, is a Nash equilibrium profile.

□

Proposition 2. *If a Nash equilibrium profile in a product game cannot be represented as a product of factor profiles, then the profile is a* weak *Nash equilibrium (at least one player is indifferent about playing his equilibrium strategy). Furthermore, there exists an equilibrium profile which* is *the product of equilibria profiles in factor games, such that each player is indifferent between their strategies in the original and product profiles.*

Proof. Let profile P^C be a Nash equilibrium in product game G^C that is not a product of factor profiles. Our proposition states that there exists a mixed strategy m'_i for each player i such that i.) $u_i^C(P_{-i}^C \cup \{m'_i\}) = u_i^C(P^C)$ and ii.) each m'_i is the product of factor mixtures which comprise equilibrium profiles in their corresponding factor games.

Lemma 1 requires that even if a Nash equilibrium cannot be written as the product of factor profiles, its projections must nonetheless be Nash equilibria in their corresponding factor games. Let each m'_i called for by our theorem be the product of the mixtures belonging to player i in the projections of P^C. This satisfies the second condition of our Proposition, the first condition must be true since both m_i and m'_i are best responses to the rest of the profile. □

Corollary 1. *Given factors of a game, G, a Nash equilibrium of G can be found in $O(n + \texttt{NE}(n'))$ time, where n is the representation length of G and n' is the representation length of its largest factor.*

Proof. This follows trivially from the fact that we can construct an equilibrium by taking the product of equilibrium profiles in the factor games of G. □

Corollary 2. *For any game with representation length n and constant maximum irreducible factor size, its totally factored representation (i) has length $O(\log(n))$, and (ii) allows a Nash equilibrium to computed in time $O(n)$.*

Proof. If the maximum factor size is m then there can be at most $\log_m(n)$ irreducible factors. Computing Nash equilibria of $\log_m(n)$ constant size games requires linear time. □

Proposition 3. *Our algorithm finds all total factorings of an NFG with bounded payouts and representation length n in expected time worst-case polynomial in n.*

Proof. Our algorithm for factoring games is correct (finds only total factorings) since factoring the polynomial representation ensures strategies can be represented as the sum of their factors and strategic interactions are isolated in their respective factors. Our algorithm is complete since we find all possible total factorings, and has this time complexity as described in the main text. □

Theorem 4. *The product of Nash equilibrium profiles in an ϵ-factoring of a game is an ϵ-Nash equilibrium of the game.*

Proof. Let $P^1, \ldots P^k$ be profiles in Nash equilibrium in ϵ-factors of game G. We know from Theorem 3 that the product of $P^1, \ldots P^k$, P', must be a Nash equilibrium in their product game G', and from the definition of an ϵ-factoring that the utility of every pure strategy profile in G' is within ϵ of its corresponding profile in G. Trivially, this result extends to mixed strategy profiles. By playing P' in G' no agent can deviate and improve utility at all, thus the most any player could possibly improve over playing P' in G is due to a difference between the two games, which is bounded by ϵ. □

Proposition 4. *Our algorithm finds an ϵ-factoring that isolates a specific decision with minimal ϵ in time polynomial in the representation of the game.*

Proof. The factoring has minimal ϵ due to the construction of the algorithm (all pairs of outcomes are checked for each player and there are n^2 pairs of outcomes). Polynomial complexity follows from the fact that the number and size of each of the linear programs is polynomial ($O(n)$ variables and constraints), and linear programs can be solved in polynomial time. □