Rational Generating Functions and Integer Programming Games

Matthias Köppe
University of California, Davis, Department of Mathematics, One Shields Avenue, Davis, CA 95616, USA
mkoeppe@math.ucdavis.edu

Christopher Thomas Ryan, Maurice Queyranne
University of British Columbia, Sauder School of Business, 2053 Main Mall, Vancouver, BC, Canada, V6T 1Z2
{chris.ryan, maurice.queyranne}@sauder.ubc.ca

We explore the computational complexity of computing pure Nash equilibria for a new class of strategic games called integer programming games with difference of piecewise linear convex payoffs. Integer programming games are games where players’ action sets are integer points inside of polytopes. Using recent results from the study of short rational generating functions for encoding sets of integer points pioneered by Alexander Barvinok, we present efficient algorithms for enumerating all pure Nash equilibria, and other computations of interest, such as the pure price of anarchy, and pure threat point, when the dimension and number of “convex” linear pieces in the payoff functions are fixed. Sequential games where a leader is followed by competing followers (a Stackelberg–Nash setting) are also considered.

Key words: algorithmic game theory, integer programming, Barvinok’s generating functions, pure Nash equilibria, pure price of anarchy, pure threat point

1. Introduction

In this paper we introduce a new class of strategic games that have appealing properties, and whose set of pure Nash equilibria can be described in a convenient encoding by rational generating functions using techniques pioneered by Barvinok (1994) and expanded by Barvinok and Woods (2003). Generating functions techniques based on Barvinok’s results have been applied to discrete optimization (see, for instance, De Loera et al. (2008), De Loera et al. (2007)), and various other areas of applied and pure mathematics (see De Loera (2005) for a survey). To the authors’ knowledge this is the first application of the theory of rational generating functions to the study of games.

This paper is motivated by open questions regarding the computational complexity of deciding the existence of pure Nash equilibria in strategic games. For a general reference on complexity issues in game theory see Papadimitriou (2007). As opposed to the case of mixed-strategy Nash equilibria which are guaranteed to exist for every game, the general problem of deciding if a strategic game has a pure Nash equilibrium is NP-hard (Gottlob et al. 2005). In view of this difficulty, the problem has been explored under various restrictions on the actions and payoffs of the players; for instance, in graphical games (Alvarez et al. 2005, Gottlob et al. 2005), congestion games (Dunkel and Schulz 2008, Fabrikant et al. 2004) and action graph games (Jiang and Leyton-Brown 2007). This paper continues this tradition, by introducing a class of games that will be shown to have convenient algorithms to decide if instances have pure Nash equilibria, and if they exist, to compute them.

We consider integer programming games, which are simultaneous games where the players’ actions (pure strategies) are lattice points (i.e., integer points) inside polytopes described by systems of linear inequalities. Since the sets of actions are given implicitly by the description of polytopes, they may be of exponential size with respect to the input size. In our setting, each player’s payoffs are
given as difference of piecewise linear convex (DPLC) functions. As an aside, optimization problems involving the difference of convex functions are a well-studied class of nonconvex programs (see for example Horst and Thoai (1999)).

The main result of the paper is that the set of pure Nash equilibria of integer programming games with DPLC payoffs can be encoded as a short rational generating function in polynomial time when the number of players, dimensions of the polytopes that define the action sets and the number of “convex” linear pieces in the payoffs are all fixed. Although these conditions are restrictive, note that each player may have an exponential number of actions. Indeed integer programming games with DPLC payoffs are a subset of a general class of games where deciding if a pure Nash equilibrium exists is \( \Sigma^P_2 \)-complete with a fixed number of players and exponential-sized strategy spaces of Alvarez et al. (2005).

Besides questions of complexity, a short rational generating function encoding is a convenient data structure for answering other questions of interest regarding the structure of pure Nash equilibria and related concepts. For instance, several questions analogous to those explored in Conitzer and Sandholm (2003) regarding mixed strategy Nash equilibria can be answered efficiently in our setting for pure Nash equilibria by using the rational generating function data structure.

We feel the main contributions of the paper are:

- Introducing the use of Barvinok short rational generating functions to the study of strategic games and demonstrating the power of encoding sets of pure equilibria as generating functions.
- Presenting a tractable class of games, integer programming games with DPLC payoffs, for which pure strategy Nash equilibria and related quantities can be computed in polynomial time when certain dimensions are fixed.

Also of note are two ideas used in several places in this paper:

- In order to represent sets of equilibria, or other sets of interest, as rational generating functions we express the set as an overall feasible set in which unwanted elements, expressed as the union of projections of lattice point sets in polytopes, are removed. See for instance the proof of Theorem 5 where the set of pure Nash equilibria is defined in this way. This is a general technique that is adapted to various settings in Sections 4 to 6.
- Some results are easier to show when the actions for each player in the game are extended to include a component that captures the payoff of that player. This extension allows for descriptions of extended strategy profiles and equilibria that are more amenable to generating function techniques and can readily be converted back to normal strategy profiles and equilibria. See for instance Definition 3.

The paper is organized into the following sections. Section 2 introduces integer programming games and discusses an application of this framework to a model of competing firms producing identical indivisible goods. Section 3 reviews the basic theory of Barvinok generating functions and major results that will be used in the remainder of the paper. Section 4 discusses pure Nash equilibria and contains the main contributions of the paper – demonstrating how generating functions can be used to encode sets of pure Nash equilibria. Section 5 details several other applications of generating function constructions to the computation of Pareto optima, the price of anarchy, and pure minmax values. Lastly, Section 6 describes a sequential (Stackelberg–Nash) version of an integer programming game where a leader’s actions affects the description of the polytopes defining the actions sets of a group of followers, who then play a simultaneous game.

2. Integer Programming Games

We begin by introducing the following class of strategic games:

**Definition 1 (Integer Programming Game).** An integer programming game with \( n \) players is a noncooperative game where the \( S_i \) of actions for each player \( i \) is the set of lattice points inside a polytope; that is,

\[
S_i = P_i \cap \mathbb{Z}^{d_i}
\]
where \( P_i = \{ x \in \mathbb{R}^{d_i} : M_i x \leq b_i \} \) is a rational polytope.

Let \( I = \{ 1, \ldots, n \} \) denote the set of players. The set \( S \) of action profiles \( s = (s_1, \ldots, s_n) \) is the Cartesian product of the \( S_i \)'s:

\[
S = \prod_{i=1}^{n} S_i \subseteq \mathbb{Z}^d
\]

where \( d = d_1 + \cdots + d_n \). The payoff functions are integer-valued of the form \( u_i : S \to \mathbb{Z} \) for \( i \in I \).

As noted in the introduction, a distinguishing feature of this class of games is that the action sets \( S_i \) are defined succinctly by linear systems \( M_i x \leq b_i \), even though \( |S_i| \) may be exponential in size with respect to the input size of \( M_i \) and \( b_i \). We use rational generating functions to avoid explicitly enumerating each player’s action set.

**Definition 2 (DPLC payoffs).** An integer-valued payoff function \( u : S \to \mathbb{Z} \) of a game is a **difference of piecewise-linear convex functions** or DPLC function if it can be expressed as:

\[
u(s) = \max_{k \in K} f_k(s) - \max_{l \in L} g_l(s)\]

where the \( f_k \) and \( g_l \) are affine functions with integer coefficients and where \( K \) and \( L \) are finite index sets. We refer to the \( f_k \) as the “convex” pieces of \( u \) and the \( g_l \) as the “concave” pieces of \( u \).

We consider integer programming games where each player \( i \) has a DPLC payoff function

\[
u_i(s) = \max_{k \in K_i} f_{ik}(s) - \max_{l \in L_i} g_{il}(s)
\]

again where the \( f_{ik} \) and \( g_{il} \) are given affine functions with integer coefficients and the sets \( K_i \) and \( L_i \) are given finite index sets.

A first comment regarding this class of games is that they can express any finite game given in normal form. A normal form game is defined by action sets \( A_1, \ldots, A_n \) and payoffs \( \pi_i(a) \) for \( a \in A_1 \times \cdots \times A_n \) and \( i \in I \). We refer to the set \( A = \prod_{i=1}^{n} A_i \) as the set of action profiles. A normal form game is finite if \( A \) is finite. We use the alternate notation \( A_i \) and \( \pi_i \), as opposed to \( S_i \) and \( u_i \), for normal form games to draw a contrast between the fact that in normal form game the actions sets and payoffs for each action profile are given explicitly as part of the input, whereas in integer programming games the action sets \( S_i \) and payoffs \( u_i \) are given implicitly. This contrast between implicit and explicit representations is important when one considers the computational complexity of finding pure Nash equilibria (see [Alvarez et al.](#) (2005) for a technical discussion).

**Proposition 1.** Every finite normal form game is equivalent to an integer programming game with DPLC payoffs.

**Proof.** Let finite normal form game \( G \) be defined by action sets \( A_1, \ldots, A_n \) and payoffs \( \pi_i(a) \) for \( a \in A_i \). Let \( d_i \) equal the number of elements in the action set \( A_i \). Let vector \( x_i = (x_{i,1}, \ldots, x_{i,d_i}) \) denotes a mixed strategy profile for player \( i \). Mixed strategy profiles \( x \) lie in the unit simplex:

\[
P_i = \{ x_i = (x_{i,1}, \ldots, x_{i,d_i}) \in \mathbb{R}^{d_i} : x_i \geq 0, \sum_{j=1}^{d_i} x_{i,j} = 1 \}.
\]

This is a polytope, and the lattice points inside \( P_i \) are exactly its vertices, which correspond to pure strategies (actions) for player \( i \), namely, an action \( a_i \) in \( A_i \) is represented by its characteristic vector \( \chi_i(a_i) \). Thus, we represent the set of pure strategies for player \( i \) as \( S_i = P_i \cap \mathbb{Z}^{d_i} \). As above let \( S = S_1 \times \cdots \times S_n \). This conforms to the framework of integer programming games.

As for the payoffs, our goal is to convert the explicitly given payoffs \( \pi_i \) over the set \( A \) of action profiles into equivalent DPLC payoff functions \( u_i \) over \( S \). Let \( A^+_i = \{ a \in A : \pi_i(a) > 0 \} \) and \( A^-_i = \{ a \in A : \pi_i(a) < 0 \} \). For every \( x = (x_1(a_1), \ldots, x_n(a_n)) \) \( \in S \), define the DPLC payoff

\[
u_i(x) = \max_{a_i \in A^+_i} \pi_i(a) \left( \sum_{i=1}^{n} x_{i,a_i} - n + 1 \right) - \max_{a_i \in A^-_i} \pi_i(a) \left( n - \sum_{i=1}^{n} x_{i,a_i} - 1 \right).
\]
Note that for every $a \in A$ we have $u_i(\chi_1(a_1), \ldots, \chi_n(a_n)) = \pi_i(a)$. Thus the normal form game $G$ is equivalent to the integer programming game with pure strategy sets $S_i$ and DPLC payoffs $u_i$. □

Note that this representation has the same input size as the normal form game itself. Further computational complexity consequences of this proposition are discussed in Remark 2.

Another useful result demonstrates the power of modeling payoffs as DPLC functions. Zalgaller (2000, Theorem 4.2) shows that every continuous piecewise linear function can be represented as the difference of two piecewise linear convex functions. Thus, a DPLC function can be used to describe any continuous piecewise linear payoff function, which in turn can be used to approximate an arbitrary continuous payoff function.

Example 1 (Cournot game with indivisible goods and fixed production costs).
A set of $n$ manufacturers (the players) produce indivisible goods which are then sold in a market. Player $i$ chooses an integer production level $s_{ij}$ for each of its indivisible products $j \in \{1, \ldots, d_i\}$ subject to resource constraints $M_is_i \leq b$, where $s_i = (s_{i1}, \ldots, s_{id_i})$. Thus, player $i$'s action set $S_i$ is the set of integer points in a polytope in $\mathbb{R}^{d_i}$. The payoff to each player consists of revenues and production costs. Under usual assumptions, the revenue to each player $i$ is a concave function of the action profile $s_i$, which can be expressed as a piecewise linear concave function $\min_{l \in R_i} r_{il}(s_i)$. For each player $i$ and product $j$ there may be a fixed production cost $F_{ij}$. The variable production cost is a convex function of the production levels $s_i$, expressed as a piecewise linear convex function $\max_{l \in C_i} c_{il}(s_i)$. The payoff to player $i$ is thus

$$u_i(s) = \sum_{j=1}^{d_i} \max \{ -F_{ij}, -F_{ij}s_{ij} \} - \left( \max_{l \in R_i} (r_{il}(s_i)) + \max_{l \in C_i} c_{il}(s_i) \right)$$

which can be expressed as a DPLC function. As will be discussed further in Remark 3 this is precisely the structure that is analyzed in this paper.

3. Introduction to rational generating functions
We give a very brief overview of the theory of short rational generating functions used in this paper. Barvinok (1994) introduced an algorithm for determining the exact number of lattice points in a rational polytope $P = \{ x \in \mathbb{R}^n : Ax \leq b \}$, that runs in polynomial time for every fixed dimension $n$. This algorithmic breakthrough provided a strengthening of the famous algorithm of Lenstra (1983), which allows to decide whether $P$ contains a lattice point in polynomial time for every fixed dimension.

Barvinok’s method works as follows. Consider the generating function of the lattice point set $P \cap \mathbb{Z}^n$, which is defined as the multivariate Laurent polynomial

$$g(P; \boldsymbol{\xi}) = \sum_{x \in P \cap \mathbb{Z}^n} \xi^x = \sum_{x \in P \cap \mathbb{Z}^n} \xi^{x_1} \cdots \xi^{x_n} \in \mathbb{Z}[[\xi_1^{\pm 1}, \ldots, \xi_n^{\pm 1}]]. \quad (3)$$

This Laurent polynomial, when encoded as a list of monomials, has exponential encoding size. Barvinok’s algorithm computes a different representation of the function $g(P; \boldsymbol{\xi})$ as a sum of basic rational functions in the form

$$g(P; \boldsymbol{\xi}) = \sum_{i \in I} \gamma_i \left( \frac{\xi^{c_i}}{(1 - \xi^{d_{i1}})(1 - \xi^{d_{i2}}) \cdots (1 - \xi^{d_{in}})} \right), \quad (4)$$

where $I$ is a polynomial-size index set and all data are integer. This algorithm runs in polynomial time whenever the dimension $n$ is a fixed constant. A formula of the type (4) is called a short rational generating function.

Note that each of the basic rational functions has poles (the point $\boldsymbol{\xi} = 1$ in particular is a pole of all the basic rational functions), but after summing up only removable singularities remain.
Obtaining the exact number of lattice points of $P$ is easy in this, since clearly $|P \cap \mathbb{Z}^n| = g(P; 1)$. Since (4) is a formula for the same function (except for removable singularities), we also have $|P \cap \mathbb{Z}^n| = \lim_{\varepsilon \to 0} g(P; \xi)$, which can be evaluated in polynomial time by performing a residue calculation with each basic rational function in the sum (4). An important point to note is that this evaluation is possible with arbitrary rational generating functions that correspond to finite lattice point sets. In other words, if we can compute in polynomial time a rational generating function of a finite lattice point set $S \subseteq \mathbb{Z}^n$, we can also compute in polynomial time its cardinality $|S|$. Therefore, we can also decide in polynomial time whether $S \neq \emptyset$.

A first, trivial observation that allows to combine rational generating functions is the following. Let $X \subseteq \mathbb{Z}^n$ and $Y \subseteq \mathbb{Z}^k$ be lattice point sets given by their short rational generating functions $g(X; \xi)$ and $g(Y; \eta)$. Then the direct product (Cartesian product) $X \times Y$ also has a short rational generating function that is simply the product of the rational functions:

$$g(X \times Y; \xi, \eta) = g(X; \xi) \times g(Y; \eta).$$

Barvinok and Woods (2003) developed powerful algorithms to obtain short rational generating functions of more general lattice point sets. The first of these algorithms concerns constant-length Boolean combinations of finite lattice point sets that are already given by rational generating functions.

**Theorem 1 (Boolean Operations Theorem).** (Corollary 3.7 in Barvinok and Woods (2003)) Let $m$ and $\ell$ be fixed integers, and let $\phi: \{0, 1\}^m \to \{0, 1\}$ be any Boolean function such that $\phi(0) = 0$. Then there exists a constant $s = s(\ell, m)$ and a polynomial-time algorithm for the following problem. Given as input, in binary encoding,

- (I$_1$) the dimension $n$ and
- (I$_2$) rational generating functions

$$g(S_p; \xi) = \sum_{i \in I_p} \gamma_{pi} \xi^{c_{pi}} \frac{1 - \xi^{d_{p1}} \ldots 1 - \xi^{d_{pis}}}{(1 - \xi^{d_{p1}}) \ldots (1 - \xi^{d_{pis}})},$$

of $m$ finite sets $S_p \subseteq \mathbb{Z}^n$, represented by the rational numbers $\gamma_{pi}$, integer vectors $c_{pi}$ and $d_{pij}$ for $p = 1, \ldots, m$, $i \in I_p$, $j = 1, \ldots, \ell_m$ such that the numbers $\ell_{mp}$ of terms in the denominators are at most $\ell$,

output, in binary encoding,

- (O$_1$) rational numbers $\gamma_i$, integer vectors $c_i$, $d_{ij}$ for $i \in I$, $j = 1, \ldots, s_i$, where $s_i \leq s$, such that

$$g(S; \xi) = \sum_{i \in I} \gamma_i \frac{\xi^{c_i}}{(1 - \xi^{d_{i1}}) \ldots (1 - \xi^{d_{is_i}})}$$

is a rational generating function of the finite set $S$ that is the Boolean combination of $S_1, \ldots, S_p$ corresponding to the function $\phi$.

Note that the restriction $\phi(0) = 0$ ensures that the set $S$ will be finite. The essential part of the construction of Theorem 1 is the implementation of set intersections, which are based on the Hadamard product Barvinok and Woods (2003, Definition 3.2), which is the bilinear extension of the operation defined on monomials as

$$\alpha x^{\xi} \ast \alpha' x^{\xi'} = \begin{cases} \alpha \alpha' x^{\xi} & \text{if } \xi = \xi', \\ 0 & \text{otherwise}. \end{cases}$$

With this definition, clearly

$$g(S_1 \cap S_2) = g(S_1; \xi) \ast g(S_2; \xi).$$
Another powerful method to define lattice point sets is by integer projections. Let \( S \subseteq \mathbb{Z}^n \) be a finite lattice point set, given by its rational generating function \( g(S; \xi) \). Let \( \psi: \mathbb{Z}^n \to \mathbb{Z}^k \) be a linear function and denote by \( T = \psi(S) \) the image (projection) of \( S \). If the map \( \psi \) is one-to-one (injective) from \( S \), then the generating function \( g(T; \eta) \) of the projection \( T \) can be computed by making a monomial substitution in \( g(S; \xi) \); see Barvinok and Woods (2003, Theorem 2.6). This fact is used in the proof of Corollary 2.

When \( S \) is the set of lattice points in a polytope \( P \), the integer projection method of Barvinok and Woods (2003) can be employed to construct a rational generating function of the projection \( T \).

**Theorem 2 (Projection Theorem).** (Theorem 1.7 in Barvinok and Woods (2003)) Let the dimension \( n \) be a fixed constant. Then there exists a constant \( s = s(n) \) and a polynomial-time algorithm for the following problem. Given as input, in binary encoding,

(I) an inequality description of a rational polytope \( P \subset \mathbb{R}^n \);

(II) a positive integer \( k \); and

(III) a linear map \( \psi: \mathbb{Z}^n \to \mathbb{Z}^k \) given by an integral matrix;

output, in binary encoding,

(1) rational numbers \( \gamma_i \), integer vectors \( c_i, d_{ij} \) for \( i \in I, j = 1, \ldots, s_i \), where \( s_i \leq s \), such that

\[
g(T; \xi) = \sum_{i \in I} \gamma_i \frac{s_i^{c_i}}{(1 - s_i^{d_{i1}})(1 - s_i^{d_{i2}})\ldots(1 - s_i^{d_{is_i}})}
\]

is a rational generating function of the set \( T = \psi(P \cap \mathbb{Z}^n) \).

Once a rational generating function of a set \( S \) has been computed, various pieces of information can be extracted from it. We have already mentioned that it is possible to compute the cardinality of \( S \) in addition to that, we can explicitly enumerate all elements of \( S \). Since the cardinality of \( S \) can be exponential in the encoding length of the input, we use output-sensitive complexity analysis, i.e., to measure the complexity of the enumeration algorithm in terms of both the input and the output. The strongest notion of an output-sensitive polynomial-time enumeration algorithm is that of a polynomial-space polynomial-delay enumeration algorithm. Such an algorithm only uses space that is polynomial in the encoding length of the input data. In addition, the time spent between outputting two items, and before outputting the first item and after outputting the last item, is bounded by a polynomial in the encoding length of the input data. The following result is a version of Theorem 7 of De Loera et al. (2007).

**Theorem 3 (Enumeration Theorem).** Let the dimension \( k \) and the maximum number \( \ell \) of binomials in the denominator be fixed. Then there exists a polynomial-space polynomial-delay enumeration algorithm for the following enumeration problem. Given as input, in binary encoding,

(I) a number \( M \in \mathbb{Z}_+ \);

(II) rational numbers \( \gamma_i \), integer vectors \( c_i, d_{ij} \) for \( i \in I, j = 1, \ldots, \ell_i \), where \( \ell_i \leq \ell \) such that

\[
\sum_{i \in I} \gamma_i \frac{s_i^{c_i}}{(1 - s_i^{d_{i1}})(1 - s_i^{d_{i2}})\ldots(1 - s_i^{d_{i\ell_i}})}
\]

is a rational generating function of a set \( V \subseteq \mathbb{Z}^k \) of lattice points with \( V \subseteq [-M, M]^k \);

(III) an integer \( p \) with \( 1 \leq p \leq k \);

output, in binary encoding,

(1) all points in the projection of \( V \) onto the last \( p \) components,

\[
W = \{ w \in \mathbb{Z}^p : \exists t \in \mathbb{Z}^{k-p} \text{ such that } (t, w) \in V \},
\]

in lexicographic order.
In addition, binary search can be used to optimize a linear function over a lattice point set encoded as a rational generating function.

**Theorem 4 (Linear Optimization Theorem).** Let the dimension \( k \) and the maximum number \( \ell \) of binomials in the denominator be fixed. Then there exists a polynomial-time algorithm for the following problem. Given as input, in binary encoding:

1. \((I_1)\) and \((I_2)\) as in Theorem 3
2. a vector \( \mathbf{f} \in \mathbb{Z}^k \),

output, in binary encoding,

1. \((O_1)\) an optimal solution \( \mathbf{v}^* \in \mathbb{Z}^k \) of the optimization problem \( \max \{ \langle \mathbf{f}, \mathbf{v} \rangle : \mathbf{v} \in V \} \).

We will use all the above results in the following constructions.

### 4. Calculating Pure Nash Equilibria in Integer Programming Games

Consider an integer programming game with DPLC payoffs as defined in Section 2. Our goal is to encode the Nash equilibria of such a game as a short rational generating function. The most general setting we provide an efficient algorithm for such an encoding is when the number of players and the dimension of their action sets are fixed and each player’s DPLC payoff function has the form:

\[
u_i(\mathbf{s}) = \max_{k \in K_i} f_{ik}(\mathbf{s}) - \max_{l \in L_i} g_{il}(\mathbf{s})
\]

where we now assume that the size of \( K_i \) is a fixed. Since \( S \) is bounded we assume without loss of generality that \( u_i(\mathbf{s}) \geq 0 \) for all \( i \in I \) and \( \mathbf{s} \in S \). The analysis proceeds with two fundamental insights. First, when there are a fixed number of “convex” pieces, i.e., when \( |K_i| \) is fixed, each player’s payoff is piecewise linear concave within the region where an \( f_{ik} \)’s remains maximum. The second insight is that when payoffs are piecewise linear concave the hypograph of the payoff function is then a polyhedral set, encodable as a short rational generating function.

First, a simple result towards partitioning the action profile space into regions according to the values of the linear pieces of the payoffs. We assume that \( K_i \) is a totally ordered set.

**Lemma 1.** For each player \( i \), the set of all action profiles can be expressed as a disjoint union

\[
S = \bigcup_{k \in K_i} S_{ik}
\]

where

\[
S_{ik} = \left\{ \mathbf{s} \in S : \begin{array}{ll}
f_{ik}(\mathbf{s}) \geq f_{ij}(\mathbf{s}), & j > k \\
f_{ik}(\mathbf{s}) > f_{ij}(\mathbf{s}), & j < k 
\end{array} \right\}.
\]

**Proof.** We first show that \( S = \bigcup_{k \in K_i} S_{ik} \) and later establish it is a disjoint union. Clearly \( \bigcup_{k \in K_i} S_{ik} \subseteq S \). It remains to show the reverse inclusion. Let \( \mathbf{s} \in S \) and define \( J(\mathbf{s}) = \{ j \in K_i : f_{ik}(\mathbf{s}) = \max_{k \in K_i} f_{ik}(\mathbf{s}) \} \) and \( j(\mathbf{s}) = \min J(\mathbf{s}) \), then \( \mathbf{s} \in S_{ij}(\mathbf{s}) \).

To show that the union is in fact disjoint suppose by contradiction that there exists an \( \mathbf{s} \) in \( S_{ik} \cap S_{ik'} \) where \( k > k' \). If \( k > k' \) then since \( \mathbf{s} \in S_{ik} \) this implies \( f_{ik}(\mathbf{s}) \geq f_{ik'}(\mathbf{s}) \). However, since \( \mathbf{s} \in S_{ik'} \) this yields that \( f_{ik'}(\mathbf{s}) > f_{ik}(\mathbf{s}) \), a contradiction. The result follows.

Note that we could equivalently write \( S_{ik} \) as follows,

\[
S_{ik} = \left\{ \mathbf{s} \in S : \begin{array}{ll}
f_{ik}(\mathbf{s}) \geq f_{ij}(\mathbf{s}), & j > k \\
f_{ik}(\mathbf{s}) \geq f_{ij}(\mathbf{s}) + 1, & j < k
\end{array} \right\}.
\]

We can do this because the action profiles in \( S \) are integer vectors and the data defining the functions \( f_{ij} \) are integral. The same holds true for all the strict inequalities in this paper, and thus
the alternative descriptions using a strict inequality or a weak inequality with one unit added to the larger side of the inequality are used interchangeably throughout.

The next step is to refine the partition of $S$ to account for all players simultaneously. To do so we introduce the following convenient notation. Let $K = \prod_{i=1}^{n} K_i$ be the set of all vectors $k = (k_1, \ldots, k_n)$ of indices where $k_i \in K_i$ for $i \in I$. Using this notation, denote by $S_k$ the intersection $S_{1k_1} \cap \cdots \cap S_{nk_n}$. Employing this notation we state the following simple corollary of the previous lemma:

**Corollary 1.** The set of action profiles can be partitioned as follows:

$$S = \bigcup_{k \in K} S_k$$

Thus each action profile $s$ lies in a unique subset $S_k$ of $S$ where it is known that the payoff for each player $i$ is

$$u_i(s) = f_{ik_i}(s) + \min_{l \in L_i} g_{il}(s)$$

and hence a piecewise linear concave function of $s$. To take advantage of this concave structure in the payoff we propose an extension of the game.

**Definition 3 (Extended Game).** Given an integer programming game with DPLC payoffs let $\hat{s} = (s; y) = (s_1, \ldots, s_n; y_1, \ldots, y_n)$ denote an extended action profile which includes variables $y_i \in \mathbb{Z}$ which keep track of payoff values. The set $\hat{S}$ of extended action profiles is the set

$$\hat{S} = \bigcup_{k \in K} \hat{S}_k$$

where

$$\hat{S}_k = \{ \hat{s} = (s, y) : s \in S_k \text{ and } 0 \leq y_i \leq f_{ik_i}(s) - g_{il}(s) \text{ for all } l \in L_i \text{ and } i \in I \}.$$ 

By Corollary 1 it is easy to see that (6) is a disjoint union.

We define the extended utility function $\hat{u}_i$ for each player $i$ as

$$\hat{u}_i(\hat{s}_i) = y_i$$

which is a linear function on $\hat{S}$. We call the tuple $\hat{G} = (\hat{S}, \hat{u}_1, \ldots, \hat{u}_n)$ the extended game of the original game $G = (S, u_1, \ldots, u_n)$.

At this point there are three important observations regarding the definition of extended games. Note that since $S$ is a bounded by a polytope and the constraints $0 \leq y_i \leq f_{ik_i}(s) - g_{il}(s)$ for all $l \in L_i$ and $i \in I$ bound the values of $y_i$ it follows that each $\hat{S}_k$, and thus $\hat{S}$ is bounded. This fact will be important when encoding $\hat{S}$ by a rational generating function and applying the Linear Optimization Theorem, which we have occasion to do in what follows. Note also that $\hat{u}_i(\hat{s}_i)$ is a linear function over $\hat{S}$ and thus more amenable to encoding by generating functions than the piecewise linear payoffs of the original game.

We also remark that the extended game $\hat{G}$ is not a simultaneous-move game since the players’ choices are not independent and some choices of $\hat{s}$ may lead to infeasibility. Similar issues are explored in Bhattacharjee et al. (2000); however, we do not treat the extended game $\hat{G}$ as a game unto itself, but simply as a mathematical construct to study the equilibria of the original game $G$, and we thus ignore these considerations.

A key step is to establish a correspondence between the pure Nash equilibria to those of the original game. As will be seen, this correspondence relies on the fact that the descriptions of the action profile sets involve disjoint unions.
In analyzing games we often consider deviations from the vector \( \mathbf{k} = (k_1, \ldots, k_n) \) in which the \( i \)-th component \( k_i \) is replaced with \( k'_i \in K_i \). For convenience, let \( \mathbf{k}_{-i} \) denote the vector of the remaining (unchanged) indexes; therefore,

\[
(\mathbf{k}_{-i}, k'_i) = (k_1, \ldots, k_{i-1}, k'_i, k_{i+1}, \ldots, k_n).
\]

We follow a similar convention in terms of action profiles. Namely, given an action profile \( \mathbf{s} = (s_1, \ldots, s_n) \), if player \( i \) deviates to action \( s'_i \) then we let \( \mathbf{s}_{-i} \) denote the (unchanged) actions of the remaining players, so

\[
(\mathbf{s}_{-i}, s'_i) = (s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_n).
\]

The equilibrium concept used in the extended game is defined as follows:

**Definition 4 (Extended pure Nash equilibrium).** Let \( \hat{G} = (\hat{S}, \hat{u}_1, \ldots, \hat{u}_n) \) be an extended game. An extended pure Nash equilibrium \( \hat{s} = (s; y) \) is an extended action profile where if \( \hat{s} \in \hat{S}_k \) then there does not exist \( \hat{s}' = (s'; y') \) such that \( (\hat{s}_{-i}, \hat{s}'_i) \in \hat{S}_{(k_{-i}, k'_i)} \) and \( \hat{u}_i(\hat{s}_{-i}, \hat{s}'_i) > \hat{u}_i(\hat{s}) \).

**Lemma 2.** Consider the game \( G = (S, u_1, \ldots, u_n) \) and its extended game \( \hat{G} = (\hat{S}, \hat{u}_1, \ldots, \hat{u}_n) \) as defined above.

(i) An extended pure Nash equilibrium of \( \hat{G} \) must be of the form

\[
\hat{s} = (s; u(s)) = (s_1, \ldots, s_n; u_1(s), \ldots, u_n(s)).
\]

(ii) There is a bijection between the set \( N \) of pure Nash equilibria of the original game and the set \( \hat{N} \) of extended pure Nash equilibria of the extended game.

**Proof.** (i) Let \( \hat{s} = (s; y) \in \hat{S} \). By the disjoint union \( \hat{S} \) there exists a unique \( k \) such that \( \hat{s} \in \hat{S}_k \). It follows that \( s \in S_k \). For all \( i \in N \) we have \( s_i \in S_{ik_i} \) and \( y_i \leq f_{ik_i}(s) - \max_{l \in L_i} g_{il}(s) = u_i(s) \). Thus, \( \hat{u}_i(\hat{s}) = y_i \leq u_i(s) = \hat{u}_i(s_1, u_1(s); \ldots; s_n, u_n(s)) \). Hence, whenever \( y_i < u_i(s) \) it is profitable for player \( i \) to deviate to the extended action \( (s_i; u_i(s)) \). Therefore, an extended pure Nash equilibrium must have the form \( (7) \). (ii) Consider the mapping \( \varphi : N \longrightarrow \hat{N} \) defined by \( s \longmapsto (s; u(s)) \). We claim \( \varphi \) is a well-defined bijection. First we show that \( \varphi \) is well-defined, that is \( \varphi(s) \in \hat{N} \) for all \( s \in N \). Clearly \( \varphi(s) \) is in \( \hat{S} \). Let \( s \in S_k \) for some \( k \). Now consider a feasible deviating action for player \( i \), \( \hat{s}'_i = (s'_i, y'_i) \), where \( (\hat{s}_{-i}, \hat{s}'_i) \in \hat{S}_{(k_{-i}, k'_i)} \) for some \( k'_i \in K_i \). In other words, player \( i \) deviates from choosing \( s_i \in S_{ik_i} \) to choosing \( s'_i \in S_{ik'_i} \) and changing \( y_i \) to \( y'_i \). We have

\[
\hat{u}_i(\hat{s}_{-i}, \hat{s}'_i) = y'_i \leq u_i(s_{-i}, s'_i) \leq u_i(s) = \hat{u}_i(s)
\]

where the second inequality holds since \( s \) is a Nash equilibrium for the game \( G \) and the final equality follows from (i). It follows that \( \hat{s} \) is an extended Nash equilibrium for the game \( \hat{G} \) and mapping \( \varphi \) is well-defined.

It is clear that \( \varphi \) is injective. As for surjectivity, by part (i) if follows that every Nash equilibrium in \( \hat{N} \) has the form \( (s; u(s)) \) for some \( s \in S \). It just remains to show that all such equilibria arise with \( s \in N \). Suppose the contrary, that is \( \hat{s} = (s; u(s)) \in \hat{N} \) but \( s \notin N \). Then there must be a profitable deviation \( s' \in S \) for some player \( i \) in the original game \( G \); \( u_i(s_{-i}, s'_i) > u_i(s) \). This implies that there is a profitable deviation \( \hat{s}'_i = (s'_i; u_1(s_{-i}, s'_i)) \) in the extended game since

\[
\hat{u}_i(\hat{s}) = u_i(s) < u_i(s_{-i}, s'_i) = \hat{u}_i(\hat{s}_{-i}, \hat{s}'_i).
\]

With this bijection, we can now state the main result of the paper:

**Theorem 5.** Consider an integer programming game with DPLC payoffs given by the following input in binary encoding:
(I$_1$) the number $n$ of players, and a bound $B \in \mathbb{N}$;
(I$_2$) for each $i \in I = \{1, \ldots, n\}$, the dimension $d_i$ and an inequality description $(M_i, b_i)$ of a rational polytope $P_i = \{x \in \mathbb{R}^{d_i} : M_i x \leq b_i\} \subseteq [-B, B]^{d_i}$;
(I$_3$) for each $i \in I$, nonnegative integers $|K_i|$ and $|L_i|$, and for all integers $k, l$ such that $1 \leq k \leq |K_i|$ and $1 \leq j \leq |L_i|$, integer vectors $\alpha_{ik} \in \mathbb{Z}^{d_i}$, $\gamma_{il} \in \mathbb{Z}^{d_i}$ (where $d = d_1 + \cdots + d_n$) and integers $\beta_{ik}, \delta_{il}$ defining the affine functions $f_{ik} : S \rightarrow \mathbb{Z}$ and $g_{il} : S \rightarrow \mathbb{Z}$ by $f_{ik}(s) = \alpha_{ik} \cdot s + \beta_{ik}$ and $g_{il}(s) = \gamma_{il} \cdot s + \delta_{il}$ for all $s \in S = \prod_{i=1}^{n} (P_i \cap \mathbb{Z}^{d_i})$.

The set $\hat{N}$ of extended pure Nash equilibria of the extended game $\hat{G} = (\hat{S}, \hat{u}_1, \ldots, \hat{u}_n)$ has a short rational generating function encoding, which can be computed in polynomial time when the total dimension $d$ and the sizes $|K_i|$ are fixed for all $i \in I$.

**Proof.** We express the set of extended Nash equilibria as follows:

$$\hat{N} = \hat{S} \setminus \bigcup_{i=1}^{n} D_i$$

where

$$D_i = \bigcup_{k \in K_i} \bigcup_{k' \in K_i} \operatorname{proj}_k\left\{ (\hat{s}, \hat{s}') : \hat{s} \in \hat{S}_k, (\hat{s}_{-i}, \hat{s}'_{-i}) \in \hat{S}_{(k_i, k'_i)}, \hat{u}_i(\hat{s}_{-i}, \hat{s}_i') \geq \hat{u}_i(\hat{s}) + 1 \right\} \quad (8)$$

Note that some of the projected sets may be empty.

The set $D_i$ is the set of action profiles where player $i$ has a profitable deviation. The description of $D_i$ in (8) is a union over profitable deviations from one set in the partition of $\hat{S}$ to another. This description of $\hat{N}$ is easy to verify using the definition of extended pure Nash equilibria.

We now establish that $\hat{N}$ can be encoded as a short rational generating function. First we claim $\hat{S}$ admits such an encoding. Consider the description of $\hat{S}$ given in (6). The sets $F_{ik}$ are sets of lattice points inside rational polytopes and thus encodable as short rational generating functions. This in turn implies that $\hat{S}_k$ admits such an encoding since it in turn is the set of lattice points inside a polytope. By the Boolean Operations Theorem, it follows that $\hat{S}$ can be encoded as a short rational generating function in polynomial time, since there is a constant number of sets $\hat{S}_k$ under the assumption that the sets $I$ and $K_i$ are of fixed size.

Note in addition that the sets to be projected in (8) are again sets of lattice points inside of rational polytopes, by observing that the extended payoffs functions are linear. By the Projection Theorem it follows that each set in the union can be encoded as a short rational generating function. Using again the Boolean Operations Theorem we conclude that each $D_i$, and thus $\hat{N}$, admit short rational generating function encodings which can be computed in polynomial time when the sizes of the sets $K_i$ are fixed for all $i \in I$. We remark that the outer union in (8) (indexed by $k \in K$) is a disjoint union; thus its rational generating function can be computed by adding the rational generating functions of the parts, rather than using the construction of the Boolean Operations Theorem.

**Corollary 2.** Consider an integer programming game with DPLC payoffs given by the same input as in Theorem 5. The set $\hat{N}$ of pure Nash equilibria has a short rational generating function encoding which can be computed in polynomial time when the total dimension $d$ and the sizes $|K_i|$ are fixed for all $i \in I$.

**Proof.** By the previous theorem, we can encode the set of *extended* pure Nash equilibria, $\hat{N}$, in polynomial time. Using the bijective map $\varphi$ given in the proof of Lemma 2, we can use an appropriate monomial substitution in the rational generating function description of $\hat{N}$ to yield a short rational generating function encoding for $\hat{N}$. 

□
The true power of the previous lemma lies in the fact that having a short rational encoding of a set allows for efficient counting, optimizing and enumerating procedures as discussed in Section 3. Using these techniques we can answer numerous questions of interest on the existence, uniqueness and structure of equilibria.

**Corollary 3.** Consider an integer programming game with DPLC payoffs given by the same input as in Theorem 5. There is a polynomial time algorithm to compute the number of pure Nash equilibria, when the total dimension $d$ and the sizes $|K_i|$ are fixed for all $i \in I$. In addition, under the same assumptions there is a polynomial time algorithm to find a sample pure strategy Nash equilibrium, when at least one exists.

**Proof.** Given the short rational generating function encoding of $N$ we calculate $|N|$ in polynomial time by counting methods discussed near the beginning of Section 3. If an equilibrium exists, we can output one by running the polynomial-delay enumeration algorithm on $N$ described in the Enumeration Theorem and terminating just after one equilibrium has been generated. This can be done in polynomial time. □

Note that the algorithm runs in time polynomial in the total number $\sum_{i \in I} |L_i|$ of “concave” pieces in the payoff functions. This corollary can be used to answer the question of whether a unique pure Nash equilibrium exists – simply check whether $|N| = 1$. The following result is also immediate by the Enumeration Theorem.

**Corollary 4.** Consider an integer programming game with DPLC payoffs given by the same input as in Theorem 5. There is a polynomial-space polynomial-delay algorithm to enumerate all the pure Nash equilibria of the game when the total dimension $d$ and the sizes $|K_i|$ are fixed for all $i \in I$.

Reflecting on the content and context to the results of this section, we make the following useful remarks.

**Remark 1.** Considering the number of elements of the game we need to fix in Corollary 3 – fixed number of players, fixed dimension of the polytopes, fixed sizes of the $K_i$ – one might ask if there is an alternate method to generating functions that might yield similar results. The key observation is that the action sets are described implicitly as lattice points in polytopes given by linear inequalities, and thus the number of actions for each player may be exponential in the input size. Thus, simple enumeration of all the action profiles in $S$ is a computationally intractable approach to the problem.

**Remark 2.** It was shown in Proposition 1 that every normal form game can be expressed as an integer programming game with DPLC payoffs. Note, however, that the dimensions of the action spaces are equal to number of corresponding actions in the normal form game. Indeed, using the notation of the proof of Proposition 1 we have $S_i \in \mathbb{Z}^{A_i}$. From a complexity point of view this representation is unsatisfactory. In Corollary 3 we require the dimension of the action spaces to be fixed, and thus we can only handle a fixed number of actions in the underlying normal form game. Normal form games with a fixed number of players and fixed number of actions are computationally uninteresting.

**Remark 3.** The Cournot game in Example 1 fits the assumptions of Corollary 3. We assume that the number $n$ of players (manufacturers) is “small”, i.e., fixed, in order for each manufacturer to have appreciable market power. We also assume that the total number $d$ of products is small. The sets $K_i$ have cardinality $O(2^d)$, which is fixed, and thus the decomposition of $S$ in Corollary 3 is comprised of a constant number of subsets. Since the algorithm in Corollary 3 scales polynomially with the sizes of the sets $L_i$, we can afford a variable number of “concave” pieces in the description of the payoff functions. These “concave” pieces are used to represent general concave revenue functions and the convex parts of the cost functions, when restricted to integer points.
5. Related computations
In addition to counting and enumerating pure Nash equilibria, generating function techniques can be used to derive efficient algorithms for related computations for integer programming games. Several of these are discussed in the following subsections.

First, note that the encoding of the set of Nash equilibria as a short rational generating function, being a compact representation, is useful for learning about the specific structure of a game’s equilibria. For instance, a simple calculation suffices for deciding the existence of a pure Nash equilibrium where player \(i\) plays a given action \(s_i\). Indeed, simply find the short rational generating function encoding of

\[ N^{(s_i)} = N \cap \{ s \in S : s_i = \bar{s}_i \}. \]

in polynomial time under the same assumptions in the previous section.

Now onto some more sophisticated calculations.

5.1. Pareto optimality
Consider the question of finding Pareto optimal pure Nash equilibria, if any exist, in an integer programming game with DPLC payoffs. To tackle this, we start by encoding the set of Pareto optimal action profiles of the game.

**Theorem 6.** Consider an integer programming game with DPLC payoffs given by the same input as in Theorem 5. The set \(PO\) of Pareto optimal action profiles has a short rational generating function encoding, which can be computed in polynomial time when the total dimension \(d\) and the sizes \(|K_i|\) are fixed for all \(i \in I\).

**Proof.** The proof is similar to those in the proof of Lemma 2, Theorem 5 and Corollary 2.

(i) Define \(\hat{PO}\) as the set of Pareto optimal points in the extended game and find a generating function encoding for \(\hat{PO}\).

(ii) Derive a bijection between \(PO\) and \(\hat{PO}\).

(iii) Use the generating function of \(\hat{PO}\) and the bijection to obtain the generating function of \(PO\).

For part (i) consider the following decomposition of \(\hat{PO}\):

\[
\hat{PO} = \{ \hat{s} \in \hat{S} : \exists \hat{s}' \in \hat{S} \text{ such that } (\hat{u}_j(\hat{s}') \geq \hat{u}_j(\hat{s}) \text{ for all } j \in I) \text{ and } (\hat{u}_i(\hat{s}') > \hat{u}_i(\hat{s}) \text{ for some } i \in I) \}
\]

where

\[
PD_i = \bigcup_{k,k' \in K} \text{proj}_k \left\{ (\hat{s},\hat{s}') : \hat{s} \in \hat{S}_k, \hat{s}' \in \hat{S}_{k'}, \hat{u}_j(\hat{s}') \geq \hat{u}_j(\hat{s}) \text{ for all } j \neq i, \hat{u}_i(\hat{s}') \geq \hat{u}_i(\hat{s}) + 1 \right\}
\]

is the set of Pareto dominated points due to a better alternative for player \(i\). By analogous arguments as in the proof of Theorem 5 we can encode \(\hat{PO}\) as a short rational generating function in polynomial time.

As for (ii) the argument is nearly identical to that in the proof of Lemma 2 and is therefore omitted. Finally, (iii) uses the same idea as found in the proof Corollary 2.

Using the Boolean Operations Theorem we obtain a short rational generating function encoding of the set \(N \cap PO\) of all Pareto optimal pure Nash equilibria of the original game:

**Corollary 5.** Consider an integer programming game with DPLC payoffs given by the same input as in Theorem 5. The set of Pareto optimal pure Nash equilibria has a short rational generating function encoding, which can be computed in polynomial time when the total dimension \(d\) and the sizes \(|K_i|\) are fixed for all \(i \in I\).

As in Section 4 we can use this generating function encoding to count and enumerate Pareto optimal equilibria.
5.2. Pure prices of anarchy and stability

The pure price of anarchy of a game measures the negative effects of competition on social welfare. The social welfare of an action profile is the corresponding total payoff of all players. The pure price of anarchy is the ratio of the maximum social welfare where the agents act together to maximize their total payoff to the worst social welfare that arises from a pure Nash equilibrium. The pure price of stability is the ratio of maximum social welfare to the best social welfare that arises from a pure Nash equilibrium. The pure price of anarchy has been studied in various network games, see Dunkel and Schulz (2008) and the references therein for recent results on weighted congestion games. Using rational generating function techniques we can calculate the pure price of anarchy and stability efficiently.

Theorem 7. Consider an integer programming game with DPLC payoffs given by the same input as in Theorem 5. There exist algorithms to compute the pure price of anarchy and the pure price of stability that run in polynomial time when the total dimension $d$ and the sizes $|K_i|$ are fixed for all $i \in I$.

Proof. Let $w^*$ denote the maximum social welfare attainable under cooperation of the players; that is,

$$w^* = \max \left\{ \sum_{i=1}^{n} u_i(s) : s \in S \right\}. \quad (9)$$

Again we work with the extended game. The first step in calculating $w^*$ is to note that

$$w^* = \max \left\{ \sum_{i=1}^{n} y_i : (s;y) \in \hat{S} \right\}. \quad (10)$$

The equivalence of $(9)$ and $(10)$ is verified by first noting that for all $(s;y) \in \hat{S}$, $y_i \leq u_i(s) = \sum_i \hat{u}_i(s;u(s))$. Therefore, if $(s^*,y^*)$ is an optimal solution to the right-hand side of $(10)$ then we must have $y_i^* = u_i(s)$ for all $i \in N$. This implies $s$ is an optimal solution the right-hand side of $(9)$.

To find $w^*$ we optimize the linear function $\sum_i y_i$ over $\hat{S}$. Every $(s;y) \in \hat{S}$ satisfies $s \in [-B,B]^d$ and

$$y_i \leq u_i(s) \leq B \left( \max_{k \in K_i} (||\alpha_{ik}||_1 + ||\beta_{ik}||) + \max_{l \in L_i} (||\gamma_{il}||_1 + ||\delta_{il}||) \right).$$

That is, $(s;y) \in [-M,M]^{d+n}$ for some polynomially sized integer $M$. Since in Section 4 we found a short rational generating function encoding of $\hat{S}$, we apply the Linear Optimization Theorem to calculate $w^*$ in polynomial time.

Let $\bar{w}$ denote the worst social welfare attained by a pure Nash equilibrium; that is,

$$\bar{w} = \min \left\{ \sum_{i=1}^{n} u_i(s) : s \in N \right\}.$$

To calculate $\bar{w}$ we note

$$\bar{w} = \min \left\{ \sum_{i=1}^{n} y_i : (s;y) \in \hat{N} \right\},$$

which follows from Lemma 2(i). Using the short generating function encoding of $\hat{N} \subseteq \hat{S}$ found in Section 4 we again apply the Linear Optimization Theorem to calculate $\bar{w}$ in polynomial time. Thus, we obtain the price of anarchy $w^*/\bar{w}$ in polynomial time.

The method for calculating the pure price of stability is similar. \qed
5.3. Pure Threat Point

The pure minmax value to player $i$ in a game is defined as:

$$\min_{s_{-i} \in S_{-i}} \max_{s_i \in S_i} u_i(s_i, s_{-i}).$$

(11)

Although mixed strategies are usually considered in calculating the (mixed) minmax values, here we restrict attention to pure strategies. The vector of mixed minmax values is known as the (mixed) threat point, which has drawn recent attention in the study of repeated games and explorations of computational implications of the Folk Theorem (see Borgs et al. (2007)). Analogously, we define the pure threat point as the vector of pure minmax values.

It was recently shown that, in various restrictive settings, the problem of calculating the (mixed) threat point is NP-hard. For instance, it can be shown that computing the (mixed) threat point of a three player game with binary payoffs ($\{0,1\}$) is NP-hard to approximate (see Borgs et al. (2007) Theorem 1 for a precise statement and proof.) Despite this negative result we show that pure threat points can be computed efficiently in our setting.

**Theorem 8.** Consider an integer programming game with DPLC payoffs given by the same input as in Theorem 5. There exists a polynomial time algorithm to compute the pure threat point when the total dimension $d$ and the sizes $|K_i|$ are fixed for all $i \in I$.

**Proof.** We begin by demonstrating how to calculate the minmax value for player $i$ in polynomial time for each player $i$. Observe that an optimal value to the following bilevel optimization problem is the pure minmax value of player $i$:

$$\min_{s_i, s_{-i}} \left\{ u_i(s_i, s_{-i}) : s_{-i} \in S_{-i}, s_i \in \arg \max_{s_i' \in S_i} u_i(s_i', s_{-i}) \right\}.$$  

(12)

This bilevel optimization problem (see Colson et al. (2007)) has essentially two players: a lower level player, or follower, who is player $i$, and an upper level player, or leader, who represents all the other players cooperating to “punish” $i$. Let

$$G_i = \left\{ s \in S : s_i \in \arg \max_{s_i' \in S_i} \{ u_i(s_i', s_{-i}) \} \right\}$$

denote the set of bilevel feasible solutions to (12). Note that (12) is equivalent to $\min \{ u_i(s) : s \in G_i \}$.

As before we turn our attention to the extended game. We define the analogous set $\hat{G}_i$:

$$\hat{G}_i = \left\{ \hat{s} \in \hat{S} : \hat{s}_i \in \arg \max_{\hat{s}_i' \in \hat{S}_i} \{ \hat{u}_i(\hat{s}_{-i}, \hat{s}_i) : (\hat{s}_{-i}, \hat{s}_i') \in \hat{S}_i \} \right\}.$$ 

Observe that if $\hat{s} = (s, y)$ then $y_i = u_i(s)$. The set $\hat{G}_i$ can be expressed as $\hat{G}_i = \hat{S} \setminus D_i$, where $D_i$ is defined as in (8). This follows since the optimization problem facing player $i$ is the same problem as when determining extended Nash equilibria is a single player game. Thus by a direct application of Theorem 5 we can encode $\hat{G}_i$ as a short rational generating function. Note that $\hat{G}_i \subseteq \hat{S} \subseteq [-M, M)^{d+n}$ where $M$ is as defined in the proof of Theorem 7. By applying the Linear Optimization Theorem find the optimal value of $\min y_i : (s, y) \in \hat{G}_i = \min \{ u_i(s) : s \in G_i \}$, in polynomial time under the stated assumptions. The pure threat point can thus be calculated in polynomial time by finding the minmax value for each player. □
6. Stackelberg–Nash equilibria

We now turn to applying these techniques to a sequential setting. In a Stackelberg–Nash game, Player 0 (the leader) chooses an action, described by a vector \( s_0 \in S_0 \). The remaining players \( i \in I = \{1, \ldots, n\} \) (the followers) then simultaneously choose their actions \( s_i \in S_i(s_0) \). Each player \( i \in I_0 = \{0,1,\ldots,n\} \) then collects a payoff \( u_i(s_0,s) \) where \( s = (s_1, \ldots, s_n) \).

We assume \( S_0 = P_0 \cap \mathbb{Z}^{d_0} \) where \( P_0 \) is a rational polytope. For each \( s_0 \in S_0 \) the followers play an integer programming game with DPLC payoffs. The action of each follower \( i \in I \) is described by a vector \( s_i \in \mathbb{Z}^{d_i} \) from the set \( S_i(s_0) = P_i(s_0) \cap \mathbb{Z}^{d_i} \). We assume \( P_i(s_0) \) is the rational polytope

\[
P_i(s_0) = \{ x \in \mathbb{R}^{d_i} : M_i \mathbf{x} \leq \pi_i(s_0) \}
\]

where \( \pi_i(s_0) \) is an integer valued affine function. Let \( d = d_1 + \cdots + d_n \) and \( d^+ = d_0 + d \).

Regarding payoffs, we assume each follower has a DPLC payoff \( u_i(s) \), independent of the leader’s choice \( s_0 \) and given by

\[
u_i(s) = \max_{k \in K_i} f_{ik}(s) - \max_{l \in L_i} g_{id}(s).
\]

The leader’s payoffs are defined as the DPLC function

\[
u_0(s_0, s) = \max_{k \in K_0} f_{0k}(s_0, s) - \max_{l \in L_0} g_{0d}(s_0, s).
\]

We assume all \( K_0 \) and \( L_i \) are finite index sets and all \( f_{ik} \) and \( g_{id} \) are integer valued affine functions.

Observe that given \( s_0 \in S_0 \) we have a setup identical to that of Section 4, where the set of action profiles for the followers is \( S(s_0) = \prod_{i=1}^n S_i(s_0) \subseteq \mathbb{Z}^d \).

We are interested in computing an optimal action for the leader while guaranteeing there exists a pure Nash equilibrium between the followers; see Remark 5 below for justification of our interest in pure Nash equilibria. Let \( N(s_0) \) denote the set of pure Nash equilibria between the followers when the leader has chosen action \( s_0 \in S_0 \). As in Section 4, a pure Nash equilibrium in \( N(s_0) \) is an action profile \( s \in S(s_0) \) such that for every \( i \in I \) there does not exist a deviation \( s'_i \in S_i(s_0) \) such that \( u_i(s_{-i}, s'_i) > u_i(s) \).

The leader faces the following optimization problem:

\[
\max_{s_0, s} \{ u_0(s_0, s) : s_0 \in S_0 \text{ and } s \in N(s_0) \}.
\]

Let \( N^+ \) denote the set of all Stackelberg–Nash equilibria, i.e., optimal solutions \( (s_0; s) \) to the optimization problem (15).

Remark 4. Note that this formulation implicitly assumes that the leader, after choosing \( s_0 \), can choose a pure Nash equilibrium \( s \in N(s_0) \) in order to maximize her payoff. This is a generalization to the case of competing followers of the “optimistic assumption” common in the multilevel optimization literature. The simplest illustration of the assumption is in the bilevel setting where the leader has the ability to choose among alternate optima to the follower’s problem (see Colson et al. (2007)). Here we assume more generally that the leader can choose among the alternate pure Nash equilibria between the followers.

Remark 5. The focus solely on pure strategies may need some motivation. Some choice of \( s_0 \in S_0 \) may give rise to no pure Nash equilibria in the followers’ game, leaving only mixed Nash equilibria. We assume that the leader will avoid such an \( s_0 \), even if it gave rise to higher expected payoffs. Consider this as an extreme form of risk aversion, where any equilibrium in pure strategies is preferred by the leader so as to avoid any uncertainty in payoffs. By similar reasoning, we also assume the leader will not be interested in the mixed equilibria when pure equilibria exist. Extending the optimistic assumption discussed in Remark 4, we assume the leader can compel the followers to reach a pure equilibrium whenever it exists.
Theorem 9. Consider a Stackelberg–Nash game with DPLC payoffs defined by the following input, given in binary encoding:

(I1) the number \( n \) of followers, and a bound \( B \in \mathbb{N} \);

(I2) the dimension \( d_0 \) and an inequality description \( (M_0, b_0) \) of a rational polytope \( P_0 = \{ x \in \mathbb{R}^{d_0} : M_0 x \leq b_0 \} \subseteq [-B, B]^{d_0} \) defining the leader’s feasible set \( S_0 = P_0 \cap \mathbb{Z}^{d_0} \);

(I3) for each \( i \) \( \in \{ 1, \ldots, n \} \), the dimension \( d_i \), number \( m_i \) of constraints, integer \( m_i \times d_i \) matrix \( M_i \), integer \( d_0 \times m_i \) matrix \( \Phi_i \) and integer vector \( \psi_i \in \mathbb{Z}^{m_i} \) defining the affine function \( \pi_i : S_0 \to \mathbb{Z}^{m_i} \) by \( \pi_i(s_0) = \Phi_i s_0 + \psi_i \), and defining the follower \( i \)'s parameterized polytope \( P_i(s_0) = \{ x \in \mathbb{R}^{d_i} : M_i x \leq \pi_i(s_0) \} \);

(I4) for each \( i \), nonnegative integers \( |K_i| \) and \( |L_i| \), and for all integers \( k, l \) such that \( 1 \leq k \leq |K_i| \) and \( 1 \leq j \leq |L_i| \), integer vectors \( \alpha_{ik} \in \mathbb{Z}^{d_i} \), \( \gamma_{il} \in \mathbb{Z}^{d_i} \) (where \( d = d_1 + \cdots + d_n \)) and integers \( \beta_{ik}, \delta_{ik} \) defining the affine functions \( f_{ik} : \mathbb{Z}^{d} \to \mathbb{Z} \) and \( g_{il} : \mathbb{Z}^{d} \to \mathbb{Z} \) by \( f_{ik}(s) = \alpha_{ik} \cdot s + \beta_{ik} \) and \( g_{il}(s) = \gamma_{il} \cdot s + \delta_{il} \) for all \( s \in \mathbb{Z}^{d} \).

Then there exists a polynomial-time algorithm to compute the leader’s optimum payoff and a short rational generating function encoding of the set \( N^+ \) of all Stackelberg–Nash equilibria when the total dimension \( d^+ \) and the sizes \( |K_1|, |K_2|, \ldots, |K_n| \) are fixed.

Proof. We mimic the development leading up to Theorem 5 in Section 4 by defining an extended proof.

Let \( \hat{S} = (S_0, s, y) \) where \( y_i \leq u_i(s) \) for all \( i \). As before denote \( \hat{s} = (s, y) \). Let

\[
S_{ik} = \{(s_0, s) : s_0 \in S_0, s \in S(s_0), f_{ik}(s) \geq f_{ij}(s) \text{ for } j > k, f_{ik}(s) > f_{ij}(s) \text{ for } j < k \}
\]

and thus construct the disjoint union

\[
\hat{S} = \bigcup_{k \in K} \hat{S}_k
\]

where

\[
\hat{S}_k = \{(s_0, s, y) : s_0 \in S_0, s \in S_k \text{ and } 0 \leq y_i \leq f_{ik}(s) - g_{il}(s) \text{ for all } l \in L_i \text{ and } i \in I \}
\]

denoting \( S_k = S_{1k_1} \cap \cdots \cap S_{nk_n} \). Note that \( \hat{S}_k \) is a lattice point set in a polytope, and thus we can encode \( \hat{S} \) by a short rational generating function.

Let \( \hat{N} \) denote the set of extended action profiles \( (s_0, \hat{s}) \) such that \( s_0 \in S_0 \) is any feasible leader action and \( \hat{s} \) is a pure Nash equilibrium in the followers extended game when the leader has chosen \( s_0 \). Now express \( \hat{N} \) as

\[
\hat{N} = \hat{S} \setminus \bigcup_{i=1}^n D_i
\]

where

\[
D_i = \bigcup_{k \in K} \bigcup_{k' \in K_i} \text{proj}_{s_0, \hat{s}} \left\{ (s_0, \hat{s}, \hat{s}') : (s_0, \hat{s}, \hat{s}') \in \hat{S}_k, (s_0, \hat{s}, \hat{s}') \in \hat{S}_{(k' \setminus k_i, i)}, \hat{u}_i(\hat{s}, \hat{s}_i) \geq \hat{u}_i(\hat{s} + 1) \right\}.
\]

Since all sets to be projected are lattice point sets inside polytopes, we can apply reasoning as in the proof of Theorem 5 and encode \( \hat{N} \) by a short rational generating functions under the stated assumptions. We establish a bijection between \( \hat{N} \) and the set \( \hat{N} = \{(s_0, s) : s_0 \in S_0, s \in N(s_0) \} \)
of feasible solutions to (15). We claim the function \( \varphi : N \rightarrow \hat{N} \) defined by \((s_0, s) \mapsto (s_0, s; u(s))\) is a well-defined bijection. The details are similar to that in the proof of Lemma 2 and are thus omitted. Via this bijection we derive a short rational generating function encoding of \( N \) by monomial substitution.

Now, let \( N^{(v)} \) denote the set of feasible solutions to (15) that guarantee the leader a payoff of at least \( v \):

\[
N^{(v)} = N \cap \left( \left\{ (s_0, s) : u_0(s_0, s) \geq v \right\} \right)
= N \cap \left( \bigcup_{k \in K_0} \left\{ (s_0, s) : f_{0k}(s_0, s) - g_{0l}(s_0, s) \geq v \; \forall l \in L_0 \right\} \right).
\]

Note this is an intersection and union of polytopal lattice point sets. Since the size of \( K_0 \) is fixed we can apply the Boolean Operations Theorem to encode \( N^{(v)} \) by a short rational generating function. Now, using binary search for \( v \) between \(-B^+\) and \( B^+\) where

\[
B^+ = B \left( \max_{k \in K_0} (||\alpha_{0k}||_1 + ||\beta_{0k}||_1) + \max_{l \in L_0} (||\gamma_0||_1 + ||\delta_0||_1) \right),
\]

and the counting algorithm to test for non-emptiness of \( N^{(v)} \), we can find an optimal payoff \( v^+ \) to the leader. The set \( N^+ \) is therefore equal to \( N^{(v^+)} \). \( \square \)

As in the previous section, a short rational generating function encoding of the set \( N^+ \) of Stackelberg–Nash equilibria leads to results analogous to Corollary 3 and Corollary 4. Thus, we can derive efficient procedures to decide on the existence of and to enumerate Stackelberg–Nash equilibria.

### 7. Conclusions and directions for further research

In this paper we introduced classes of games and proposed algorithms for studying their pure strategy Nash equilibria using rational generating functions. The simplicity by which they can be used to compute important information on the structure of these games, demonstrates the power of generating functions as an analytical and computational tool for game theory.

There is considerable scope to explore applications of integer programming games to other situations. In addition, the use of generating function techniques is a novel approach to game theory and can be applied to various types of games; e.g., threat point computations for repeated integer programming games; games with other payoff functions; and computations related to other solution concepts.

### Acknowledgements

The research of the last two authors was supported in part by a Discovery grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada to the third author. The second author is also supported in part by the Shelby L. Brumelle Memorial Scholarship.

### References

Alvarez, C., J. Gabarro, M. Serna. 2005. Pure Nash equilibria in a game with large number of actions. *Mathematical Foundations of Computer Science – 2005, Lecture Notes in Computer Science*, vol. 3618/2005. Springer, Berlin, Heidelberg, 95–106.

Barvinok, A. 1994. A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. *Mathematics of Operations Research* 19(4) 769–779.

Barvinok, A., K. M. Woods. 2003. Short rational generating functions for lattice point problems. *Journal of the American Mathematical Society* 16 957–979.
Bhattacharjee, R., F. Thuijsman, O. J. Vrieze. 2000. Polytope games. *Journal of Optimization Theory and Applications* **105**(3) 567–588.

Borgs, C., J. Chayes, N. Immorlica, A. T. Kalai, V. Mirrokni, C. Papadimitriou. 2007. Myth of the Folk Theorem. To appear in STOC 2008.

Colson, B., P. Marcotte, G. Savard. 2007. An overview of bilevel optimization. *Annals of Operations Research* **153**(1) 235–256.

Conitzer, V., T. Sandholm. 2003. Complexity results about Nash equilibria. *International Joint Conference on Artificial Intelligence*. 1–18.

De Loera, J. A. 2005. The many aspects of counting lattice points in polytopes. *Mathematische Semesterberichte* **52**(2) 175–195.

De Loera, J. A., R. Hemmecke, M. Köppe. 2007. Pareto optima of multicriteria integer linear programs. To appear INFORMS Journal on Computing, 2008.

De Loera, J. A., R. Hemmecke, M. Köppe, R. Weismantel. 2008. FPTAS for optimizing polynomials over the mixed-integer points of polytopes in fixed dimension. *Mathematical Programming, Series A* **118** 273–290. doi:10.1007/s10107-007-0175-8.

Dunkel, J., A. Schulz. 2008. On the complexity of pure-strategy Nash equilibria in congestion and local-effect games. *Mathematics of Operations Research* (to appear).

Fabrikant, A., C. Papadimitriou, K. Talwar. 2004. The complexity of pure Nash equilibria. *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing* 604–612.

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

Horst, R., N. V. Thoai. 1999. DC programming: Overview. *Journal of Optimization Theory and Applications* **103**(1) 1–43.

Jiang, A. X., K. Leyton-Brown. 2007. Computing pure Nash equilibria in symmetric action graph games. *American Association of Artificial Intelligence (AAAI)*. 79–85.

Lenstra, Jr., H. W. 1983. Integer programming with a fixed number of variables. *Mathematics of Operations Research* **8**(4) 538–548.

Papadimitriou, C. 2007. The complexity of finding Nash equilibria. N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani, eds., *Algorithmic Game Theory*, chap. 2. Cambridge University Press, 29–52.

Zalgaller, V. A. 2000. Representations of functions of several variables by difference of convex functions. *Journal of Mathematical Sciences* **100**(3) 2209–2227.