Exponentially Many Steps for Finding a Nash Equilibrium in a Bimatrix Game

Rahul Savani and Bernhard von Stengel
Department of Mathematics, London School of Economics,
Houghton St, London WC2A 2AE, United Kingdom
rahul@maths.lse.ac.uk, stengel@maths.lse.ac.uk

Abstract

The Lemke–Howson algorithm is the classical algorithm for the problem NASH of finding one Nash equilibrium of a bimatrix game. It provides a constructive and elementary proof of existence of an equilibrium, by a typical "directed parity argument", which puts NASH into the complexity class PPAD. This paper presents a class of bimatrix games for which the Lemke–Howson algorithm takes, even in the best case, exponential time in the dimension d *of the game, requiring* $\Omega((\theta^{3/4})^d)$ *many steps, where* θ *is the Golden Ratio. The "parity argument" for NASH is thus explicitly shown to be inefficient. The games are constructed using pairs of dual cyclic polytopes with* $2d$ *suitably labeled facets in* d*-space.*

1. Introduction

Game theory is the formal study of conflict and cooperation. In computer science, game theory has attracted recent interest for economic aspects of the internet, such as electronic commerce, selfish routing in networks [29], and algorithmic mechanism design [24]. In complexity theory, game-theoretic ideas are basic for proving lower bounds for randomized algorithms [39] and in the competitive analysis of online algorithms [2].

A *bimatrix game* is a two-player game in strategic form, a basic model in non-cooperative game theory. The strategic form is specified by a finite set of "pure" strategies for each player, and a (for simplicity of input, integer) payoff for each player for each *strategy profile* (which is a tuple of strategies, one for each player). The game is played by each player independently and simultaneously choosing one strategy, whereupon the players receive their respective payoffs. A player is allowed to randomize according to a probability distribution on his pure strategy set, which defines a *mixed strategy* for that player. Players are then interested in maximizing their expected payoffs. A *Nash equilibrium* is a profile of (possibly mixed) strategies such that no player can gain by unilaterally choosing a different strategy, where the other strategies in the profile are kept fixed. Every game has at least one equilibrium in mixed strategies [23]. For two players, the game is specified by two $m \times n$ integer matrices A and B, where the m rows are the pure strategies i of player 1 and the n columns the pure strategies j of player 2, with resulting matrix entries a_{ij} and b_{ij} as payoffs to player 1 and 2, respectively. This is called a bimatrix game (A, B).

A classic open problem is the complexity of the problem NASH of finding one Nash equilibrium of a bimatrix game. For the special case of zero-sum games, which are bimatrix games $(A, -A)$, this problem can be solved by linear programming, but in general is not known to be polynomial. Together with factoring, NASH has been called "the most important concrete open question on the boundary of P today" [27]. A standard method for finding one Nash equilibrium of a bimatrix game is the *Lemke–Howson* (LH) algorithm [15]. In this paper, we present a class of games where this algorithm is exponential. The LH algorithm is a pivoting method related to the simplex algorithm for linear programming [5], which also has worst-case exponential behavior [14]. Our games show that even the *best-case* behavior of the LH algorithm can be exponential, for *any* choice of its free parameter (the first variable "to enter the basis"). To our knowledge, these are the first examples of this kind. Finding a Nash equilibrium in subexponential time must therefore go beyond this classic pivoting approach.

The Nash equilibria of a bimatrix game are solutions to a linear complementarity problem (LCP) [4]. In [22][10], LCPs are constructed where for certain pivoting methods *one* path is exponentially long, in analogy to [14]. In our case, *all* LH paths are exponentially long. Moreover, the examples in [22][10] do not arise from games.

In: *Proc. 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2004)*, 258–267.

Our games are hard to solve for the LH algorithm for any choice of its free parameter. Another approach to finding an equilibrium is to guess its *support* (the set of pure strategies that have positive probability), solve the linear equations that equalize the expected payoffs for the pure strategies in the support, and check that no other strategy has higher payoff (see Lemma 1 below). In the games presented here, the support of the unique equilibrium is the set of all strategies, which is easily guessed. However, our games are a first step towards "challenge instances" for the problem NASH. In [30], the present construction is extended to $d \times 2d$ games. These games have many equilibria, where player 1 always uses all d rows, but where the supports for player 2 form an exponentially small fraction of all $\binom{2d}{d}$ sets of columns of size d. In these games, support guessing takes exponential expected time. Moreover, all LH paths are exponentially long.

NASH belongs to the complexity class TFNP of total function problems in NP [26, p. 229]: With the bimatrix game as input, the required output are the mixed strategy probabilities (as the decision problem whether an equilibrium exists is trivial), where the equilibrium property is verified in polynomial time. The class TFNP does not have complete problems unless NP = co-NP [20].

More specifically, NASH belongs to the subclass of TFNP call PPAD, of problems with a parity argument for directed graphs [25, p. 516]. The parity argument states that a directed graph (defined implicitly), where the indegree and outdegree of every node is at most one, consists of cycles and directed paths, so that there are as many starting points as endpoints of these paths. An instance of a problem in PPAD is specified by a polynomial-time algorithm for finding at least one starting point, and for finding the neighbor of a point in the graph or else declaring it as an endpoint. The possible endpoints (of which at least one exists) are the allowed function values. The LH algorithm is a special case. It uses a trivial *artificial equilibrium* as starting point, has a freely chosen starting edge as a parameter, and then uses a unique "complementary" pivoting rule for determining the next "basic solution". It thereby traces the vertices of a certain polytope and ends at an equilibrium. The edges of the graph are directed (so the direction of the path can be determined even without knowing the past history) by a geometric orientation [32]; see also Figure 1 below. The parity argument may be inefficient if the paths are not of polynomial length. Our paper shows explicitly that this inefficiency may occur for NASH, by giving games that produce exponentially long LH paths.

A related question is if NASH may be complete for the class PPAD. This seems to require encoding an arbitrary polynomial-time Turing machine computation into complementary pivoting steps for polynomial-sized payoff matrices, which looks difficult.

Unlike the set of solutions to a linear program, or equivalently [5, p. 290] of equilibria of a zero-sum game, the set of Nash equilibria of a general bimatrix game is not convex. Thus, NASH cannot be approached in an obvious way by interior point methods. Moreover, the set of all Nash equilibria is computationally "hard" in the sense that various associated decision problems are NP-complete, e.g. if the game has only one Nash equilibrium, or one with a certain support size, or with a player's payoff above a given bound [8][3]. Therefore, any method for finding one Nash equilibrium, e.g. by divide-and-conquer or incrementally (which is not obvious at any rate) must be weaker than characterizing the set of all equilibria in the end, if such a method is to be polynomial (unless P = NP).

Lemke's algorithm [16] is closely related to the LH algorithm. Its extra flexibility given by the choice of numerical values in an auxiliary vector, rather than just of finitely many starting edges as in the LH algorithm, deserves further study; see also [19].

Equilibrium enumeration methods (see [36][38][11][1] and the survey in [34]) can be modified to terminate once the first equilibrium is found, and would have to be tested on our games as well. These methods are designed to produce all rather than just one equilibrium, which cannot even be polynomial in the *output* size (unless P = NP), since deciding if a game has only one Nash equilibrium is NP-complete ([8], see above). There is no a priori reason to assume that these methods are good for finding just one equilibrium. Similarly, general algorithms for finding equilibria in games with any number of players are not likely to be fast. These include path-following algorithms [7], which typically specialize to pivoting in the two-player case (for a generalization of LH to more than two players see [28][37]), and algorithms for approximating fixed points [18][25].

Finding an ε*-approximate* equilibrium (see e.g. [17]) is a different problem than NASH. In our games, all points on the LH path fulfill the equilibrium condition except for one pure strategy, and it is possible that the payoff "error" for that strategy goes below ε after a small number of steps; we have not investigated this. Similarly, a pivoting method implemented in floating-point arithmetic may quickly and erroneously produce an "equilibrium" due to rounding errors. Numerical problems arise since our payoffs are derived from the moment curve, which leads to ill-conditioned matrices. These may possibly be avoided by using points on the trigonometric moment curve [40, p. 75f] [9, p. 67], as mentioned in Appendix A of [31]. Working implementations of the LH algorithm use exact integer arithmetic [35].

Our work is most closely related to that of Morris [21], who used dual cyclic polytopes to produce exponentially long paths, called "Lemke" paths in [21], for a related method. As we will explain in Section 2, this can be interpreted as the LH method for finding a *symmetric* equilibrium of a symmetric bimatrix game. However, these games have additional non-symmetric equilibria that are found very quickly by the general LH algorithm, and are therefore not useful for our purpose. Morris showed that Lemke paths cannot be used to address the Hirsch conjecture [13]. This famous conjecture states a tight linear bound on the shortest path between any two vertices of a polytope, for which the best known bounds are not even polynomial [12]. A *polynomial* pivoting algorithm for NASH (or even for finding a symmetric Nash equilibrium of a symmetric game, using the symmetrization in (2) below), applied to zero-sum games, would answer that question as well.

Section 3 describes our construction. The LH paths are defined purely combinatorially in terms of the supports of, and best responses to, the mixed strategies that it traces. These correspond to known bit patterns that encode the vertices of dual cyclic polytopes, which are one of the few classes of polytopes whose face structure is known in arbitrary dimension. Linear recurrences for the various path lengths give rise to their exponential growth. The longest path lengths are given by every third Fibonacci number, growing with $\theta^{3d/2}$ for a $d \times d$ game, where θ is the Golden Ratio. Shorter path lengths are obtained by certain sums of these, the shortest length being $\Omega(\theta^{3d/4})$. Proof details and examples of our LH paths are given in [31].

2. Games, polytopes, and the Lemke–Howson algorithm

Given a bimatrix game (A, B) with $m \times n$ payoff matrices A and B, a mixed strategy for player 1 is a vector x in $\mathbb{R}^m$ with nonnegative components that sum to one, and a mixed strategy for player 2 is a similar vector y in $\mathbb{R}^n$. All vectors are column vectors; the row vector corresponding to x is written as the transpose $x^\top$. A *best response* to y is a mixed strategy x of player 1 that maximizes his expected payoff $x^\top Ay$, and a best response to x is a mixed strategy y of player 2 that maximizes her expected payoff $x^\top By$. A Nash equilibrium is a pair of mutual best responses, that is, a mixed strategy pair $(\overline{x}, \overline{y})$ so that $\overline{x}^\top A\overline{y} \geq x^\top A\overline{y}$ and $\overline{x}^\top B\overline{y} \geq \overline{x}^\top By$ for all other mixed strategies x and y. Best responses are characterized by the following combinatorial condition, which we state only for a mixed strategy x of player 1.

Lemma 1 [23] *Let x and y be mixed strategies of player 1 and 2, respectively. Then x is a best response to y if and only if all strategies in the support of x are pure best responses to y.*

A game (A, B) is *symmetric* if $A = B^\top$, so it does not change when the players change roles. The game of "chicken" with $A = B^\top = \begin{pmatrix} 2 & 2 \\ 4 & 1 \end{pmatrix}$ is an example. Its equilibria, in terms of probability vectors, are the bottom left pure strategy pair $\big((0,1)^\top, (1,0)^\top\big)$ with payoffs $4, 2$ to players $1, 2$, the top right pure strategy pair $\big((1,0)^\top, (0,1)^\top\big)$ with payoffs $2, 4$, and the mixed strategy pair $\big((1/3, 2/3)^\top, (1/3, 2/3)^\top\big)$ with payoffs $2, 2$. The mixed strategy equilibrium is the only symmetric equilibrium. Its probabilities are uniquely determined by the condition that the pure strategies in the support of the opponent's strategy must both be best responses (by Lemma 1) and hence have equal expected payoff.

In a mixed equilibrium, the probabilities are uniquely given by the pair of supports if the corresponding submatrices have full rank; the support sizes are then equal. This holds if the game is *nondegenerate*, defined by the property that the number of pure best reponses to any mixed strategy never exceeds the size of its support (see [34] for a detailed discussion). Even in a degenerate bimatrix game, any Nash equilibrium is a convex combination of extreme equilibria [38][11], which are also determined by linear equalities. The LH algorithm can be extended to degenerate games by standard lexicographic perturbation techniques [15][34]. All games considered here are nondegenerate.

By Lemma 1, an equilibrium is given if any pure strategy of a player is either a best response (to the opponent) or is played with probability zero (by the player himself). This can be captured by polytopes [40][9] whose facets represent pure strategies, either as best responses or having probability zero. We explain first the simpler case of symmetric equilibria of a symmetric game with $d \times d$ payoff matrix C to player 1, say. Let

$$S = \{ z \in \mathbb{R}^d \mid z \geq \mathbf{0}, \ \ Cz \leq \mathbf{1} \} \qquad (1)$$

where **0** and **1** denote vectors with all entries 0 and 1, respectively, and inequalities holding for all components. We can assume that C *is nonnegative and has no zero column* by adding a constant to all payoffs, which does not change the best response structure, so that the polyhedron S is bounded and thus a polytope. We assume there are no redundant inequalities in $Cz \leq \mathbf{1}$, which would correspond to dominated strategies [34]. Then the game is nondegenerate if and only if the polytope S is *simple*, that is, every vertex lies on exactly d facets of the polytope [40][9]. A facet is obtained by making one of the inequalities defining the polytope *binding*, that is, by converting it into an equality.

Lemma 2 [36] *A mixed strategy pair* (x, y) *is a symmetric Nash equilibrium of the game* $(C, C^\top)$ *if and only if* $x = y = u \cdot z$ *and* $z \in S$ *in* (1), $z \neq \mathbf{0}$, $u = 1/\sum_i z_i$, *and* $z^\top(\mathbf{1} - Cz) = 0$, *where* z *must be a vertex of* S *by nondegeneracy.*

In the game of chicken above, $z = (1/6, 1/3)^\top$ gives the symmetric equilibrium. The vector z has to be rescaled to become a mixed strategy x. The equilibrium payoff u, normalized to 1 in $Cz \leq \mathbf{1}$, is the scaling factor. The converse mapping from x to z defines a projective transformation of a polyhedron representing the upper envelope of expected payoffs to the polytope S [34].

The conditions in Lemma 2 define an LCP [4], usually stated as: find z so that

$$z \geq \mathbf{0}, \qquad q + Mz \geq \mathbf{0}, \qquad z^\top(q + Mz) = 0,$$

here with data $M = -C$, $q = \mathbf{1}$. This LCP has a trivial solution $z = \mathbf{0}$, which is not a Nash equilibrium. However, $z = \mathbf{0}$ is an *artificial equilibrium* which is the starting point of the LH algorithm (in our first version for symmetric games, giving what [21] calls "Lemke paths").

It is useful to *label* the facets of S [32]. For each pure strategy i, the facets defined by $z_i = 0$ and by $(Cz)_i = 1$ both get label i. Every vertex has the label of the facets it lies on. The complementarity condition $z^\top(\mathbf{1} - Cz) = 0$ then means that z is *completely labeled* (has all labels i), since then each i is either not played or a best response, as required in equilibrium. Since S is simple, such a completely labeled vertex has each label exactly once.

The LH algorithm is started from the completely labeled vertex $z = \mathbf{0}$ by choosing one label k that is *dropped*, meaning that label k is no longer required. This is the only free choice of the algorithm, which from then on proceeds in a unique manner. By leaving the facet with label k, a unique edge is traversed whose endpoint is another vertex, which lies on a new facet. The label, say j, of that facet is said to be *picked up*. If this is the missing label k, the algorithm terminates at a completely labeled vertex. Otherwise, j is clearly *duplicate*, and the next edge is (uniquely) chosen by leaving the facet that so far had label j, and the process repeated. The LH method generates a sequence of k-*almost complementary* edges and vertices (having all labels except possibly k, where k occurs only at the starting point and endpoint). The resulting path cannot repeat a vertex as this would offer a second way to proceed when that vertex is first encountered, which is not the case (since S is simple). Hence, it terminates at a Nash equilibrium.

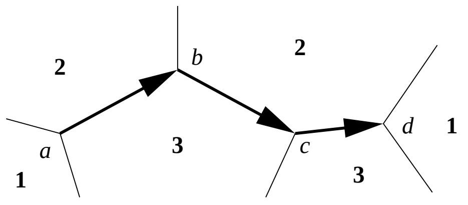

Figure 1

This is illustrated in Figure 1 for dimension 3. Point *a* is completely labeled, being adjacent to facets with labels $1, 2, 3$. Dropping label 1, it proceeds to point *b* picking up label 2, now duplicate. The next point is *c* with duplicate label 3, and finally *d* where the missing label 1 is picked up, which terminates the path.

As in the simplex algorithm [5], edge traversal is implemented algebraically by *pivoting* with variables entering and leaving a basis, the nonbasic variables representing the facets. The only difference is the rule for choosing the next entering variable, which in linear programming is done so as to improve the objective function. Here, it is the *complementary pivoting* rule where the nonbasic variable with duplicate label enters the basis.

Furthermore, the path is directed, giving a "directed parity argument" [25] which puts the problem in the class PPAD, rather than just in PPA. In Figure 1, the starting point *a* has an *orientation*, with the labels $1, 2, 3$ in clockwise order. When label 1 is dropped, the remaining labels keep their orientation (in one dimension less) relative to the edges of the path. In Figure 1, label 2 is always to the left and label 3 always to the right of the edge. At the endpoint of the path, the missing label is picked up at the other end of the edge, so that the orientation of that vertex is opposite to that of the starting vertex of the path; in Figure 1, point *d* has labels $1, 2, 3$ in anticlockwise order. This generalizes to higher dimensions, where orientation is defined as the sign of a certain determinant. The endpoints of any LH path have opposite orientation, which leads to an *index theory* of equilibria [32][7]. Knowing the orientation of the artificial equilibrium, the orientation of any k-almost complementary edge can be determined directly, which gives the PPAD property.

For nonsymmetric bimatrix games (A, B), or even for finding nonsymmetric equilibria of symmetric games as in the game of "chicken" above, the LH algorithm is applied as follows, which is its standard form. Let

$$z = \begin{pmatrix} x \\ y \end{pmatrix}, \qquad C = \begin{pmatrix} 0 & A \\ B^\top & 0 \end{pmatrix}. \tag{2}$$

The polytope S of dimension $d = m + n$ in (1) is then the *product* $P \times Q$ of the polytopes

$$\begin{aligned} P &= \{ x \in \mathbb{R}^m \mid x \geq \mathbf{0},\ \ B^\top x \leq \mathbf{1} \}, \\ Q &= \{ y \in \mathbb{R}^n \mid Ay \leq \mathbf{1},\ \ y \geq \mathbf{0} \}. \end{aligned} \tag{3}$$

Any Nash equilibrium (x, y) of (A, B) is again given by $z^\top(\mathbf{1} - Cz) = 0$, which is equivalent to $x^\top(\mathbf{1} - Ay) = 0$ and $y^\top(\mathbf{1} - B^\top x) = 0$. These conditions state that x is a best response to y and vice versa, where x and y have to be normalized to represent mixed strategies. The only difference to Lemma 2 is that this normalization has to be done separately for x and y, rather than for the entire vector z. It is easy to see that in equilibrium $x = \mathbf{0}$ if and only if $y = \mathbf{0}$, where $(\mathbf{0}, \mathbf{0})$ is the artificial equilibrium.

The LH algorithm is then applied as before, where a label corresponds either to a strategy i of player 1 or a strategy j of player 2. These have to be distinct, so it is convenient to number the n strategies of player 2 as $m + 1$, $\ldots, m+n$, as suggested in [32]. A duplicate label then represents a pure strategy that has both probability zero and is a best response. If this is a strategy i of player 1, for example, it determines the facet $x_i = 0$ in P or $(Ay)_i = 1$ in Q, corresponding to the respective ith inequality in (3).

The LH path on the edges of $S = P \times Q$ is a subgraph of the *product graph* of the edge graphs of P and Q. This means that edges are alternately traversed in P and Q, keeping the vertex in the other polytope fixed. A duplicate label picked up in Q is dropped in P and vice versa. This is the standard view of the LH algorithm; for further details see [34].

3. Lemke–Howson on labeled dual cyclic polytopes

We construct square games with $m = n = d$ strategies for each player. Similar to [33], these are derived from *dual cyclic polytopes* [40][9] in dimension d with $2d$ facets. A standard way of obtaining a cyclic polytope P' in dimension d with $2d$ vertices is to take the convex hull of $2d$ points $\mu(t_i)$ on the *moment curve* $\mu\colon t \mapsto (t, t^2, \ldots, t^d)^\top$ for $1 \leq i \leq 2d$. However, the polytopes in (3) are defined by inequalities and not as convex hulls of points. In the dual of a polytope, its vertices are re-interpreted as normal vectors of facets. The polytope P' is first translated so that it has the origin $\mathbf{0}$ in its interior, for example by subtracting the arithmetic mean $\overline{\mu}$ of the points $\mu(t_i)$ from each such point. The resulting vectors $c_i = \mu(t_i) - \overline{\mu}$ then define the dual cyclic polytope

$$P'' = \{ z \in \mathbb{R}^d \mid c_i^\top z \leq 1,\ 1 \leq i \leq 2d \}.$$

A vertex u of such a polytope is characterized by a bitstring $u_1 u_2 \cdots u_{2d}$ of length $2d$, with the kth bit u_k indicating whether u is on the kth facet ($u_k = 1$) or not ($u_k = 0$). The polytope is simple, so exactly d bits are 1, the other d bits are 0. Assume that $t_1 < t_2 < \cdots < t_{2d}$ when defining the kth facet of P'' by making the inequality $(\mu(t_k) - \overline{\mu})^\top z \leq 1$ binding. Then the vertices of P'' are characterized by the 0-1 strings fulfilling the *Gale evenness* condition [6]: A bitstring represents a vertex if and only if any substring of the form $01 \cdots 10$ has even length, so 0110, 011110, etc., is allowed, but not 010, 01110, and so on. A maximal substring of 1's is called a *run*. We only consider *even* dimension d, where the allowed odd runs of 1's at both ends of the string can be glued together to form an even run, which shows the cyclic symmetry of the Gale evenness condition. Let $G(d)$ be the set of these Gale evenness bitstrings of length $2d$ with d ones.

Both P and Q in (3) will be dual cyclic polytopes with a special order of their inequalities corresponding to the facet labels. A suitable affine transformation [33, p. 560] gives P from P'', and Q in a similar manner, so that the first d inequalities (for the pure strategies of player 1) in P have the form $x \geq \mathbf{0}$, and the second d inequalities (for the pure strategies of player 2) in Q are $y \geq \mathbf{0}$. The remaining d inequalities $B^\top x \leq \mathbf{1}$ in P and $Ay \leq \mathbf{1}$ in Q then determine the game (A, B). The game data is of polynomial size in d.

The equilibrium condition and the LH algorithm depend on which facets a vertex belongs to, as encoded in the

Gale evenness bitstrings in $G(d)$, and on the facet labels. These are defined by permutations l and l' of $1, \ldots, 2d$ for P and Q, respectively. For a vertex u of P, which we identify with its bitstring in $G(d)$, its labels are given by $l(k)$ where $u_k = 1$, and the labels of a vertex v of Q are $l'(k)$ where $v_k = 1$, for $1 \le k \le 2d$. The kth facet of P (corresponding to the kth position in a bitstring) has label $l(k) = k$, so l is simply the identity permutation. The kth facet of Q has label $l'(k)$. The permutation l' has the fixed points $l'(1) = 1$ and $l'(d) = d$, and otherwise exchanges adjacent numbers, as follows:

$$l'(k) = \begin{cases} k, & k = 1, d, \\ k + (-1)^k, & 2 \le k \le d-1, \\ k - (-1)^k, & d+1 \le k \le 2d. \end{cases} \qquad (4)$$

The artificial equilibrium e_0 is a vertex pair (u, v) so that u is labeled with $1, \ldots, d$ and v with $d+1, \ldots, 2d$. In terms of bitstrings, $u = 1^d 0^d$ (which are d ones followed by d zeros) and $v = 0^d 1^d$, which both fulfill Gale evenness, and have the indicated labels under l and l', respectively, so that

$$e_0 = (1^d 0^d, 0^d 1^d) \in G(d) \times G(d). \qquad (5)$$

A similar Nash equilibrium e_1 is readily identified, which has full support.

Lemma 3 *Let* $e_1 = (0^d 1^d, 1^d 0^d)$. *This is the only Nash equilibrium of the game.*

Proof. Let (u, v) be a completely labeled vertex pair, and suppose that $u_d = 1$. If $u_{d+1} = 1$, then $v_d = 0$ and $v_{d+2} = 0$ (via complementarity, since $l'(d+1) = d+2$) so $v_{d+1} = 0$ by Gale evenness, and thus $u_{d+2} = 1$. Continuing in that way, all 1's to the right of the dth bit u_d of u (which is 1) have to come in pairs. Similarly, if $u_{d-1} = 1$, then $v_{d-2} = 0$ by complementarity, which with $v_d = 0$ implies $v_{d-1} = 0$. This means that the 1's to the left of u_d come in pairs if there is a zero to the left of them. In the latter case, the run of 1's containing u_d has odd length, so it must include u_{2d}, but then is too long. Hence, the only possibility where $u_d = 1$ is when $(u, v) = e_0$. Similarly, $u_d = 0$ implies $(u, v) = e_1$. $\square$

Hence, all LH paths, for any dropped label, lead from e_0 to e_1. Denote by $\pi(d, k)$ the path when label k is dropped in dimension d, regarded as a sequence (u^0, v^0) $(u^1, v^1) \cdots (u^L, v^L)$ of vertex pairs in $P \times Q$, that is, in $G(d) \times G(d)$, and let $L(d, k) = L$ be the length of that path.

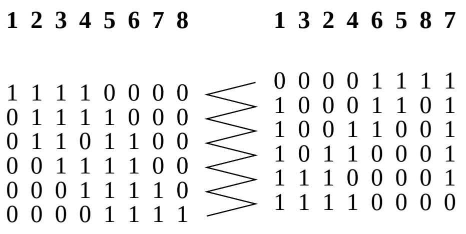

Figure 2

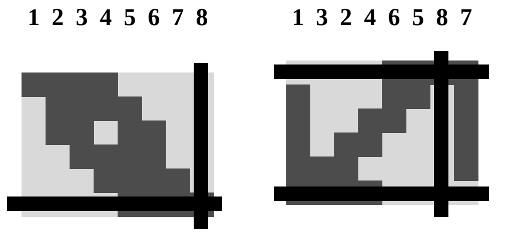

Figure 3

As an example, Figure 2 shows $\pi(4, 8)$, with P on the left and Q on the right. The numbers at the top are the labels $l(k)$ and $l'(k)$ for $k = 1, \ldots, 8$. The starting point e_0 is the vertex pair $e_0 = (u^0, v^0) = (11110000, 00001111)$. We first drop label 8 in Q, so the bit v_7^0 (since $l'(7) = 8$) changes from 1 to 0, which by Gale evenness gives the bit string 10001101 as the new vertex v^1 in Q. In Figure 2, u^0 is connected to both v^0 and v^1 by a sloped line, since $u^0 = u^1$. These sloped lines (forming the middle zigzag path) indicate the vertex pairs on the LH path, which we use since in each step only one vertex changes but the other stays fixed. In v^1, label 1 has been picked up, which is now duplicate and dropped from u^1 in P, giving the next vertex $u^2 = 01111000$. The new duplicate label is 5 and in the next step dropped in Q, giving vertex 10011001. In that manner, the path proceeds until it ends at e_1.

We will show that all paths can be expressed in terms of the two special paths $\pi(d, 1)$ and $\pi(d, 2d)$. These have certain symmetries.

Lemma 4 *Let* $L = L(d, 1)$ *and let* (u^i, v^i) *be the* i*th vertex pair of the path* $\pi(d, 1)$. *Then for* $0 \le i \le L$, $(u^i, v^i) = (v^{L-i}, u^{L-i})$.

Proof. The particular names of the labels do not matter, so we can re-name them for both P and Q with the permutation l' in (4), the kth facet in P getting label $l'(l(k))$, which is $l'(k)$, and in Q label $l'(l'(k))$, which is $l(k)$. But then P and Q switch roles, e_0 is exchanged with e_1, label 1 stays the same, and the path backwards corresponds to $\pi(d, 1)$ itself as claimed. $\square$

For the path $\pi(d,2d)$, we disregard the first vertex pair and the last two vertex pairs. The remaining path, which will call $B(d)$, is point-symmetric in each polytope, by reversing the bitstrings while ignoring the zero bit for the missing label. Figure 3 shows this for $d = 4$ where the disregarded rows and columns are struck out.

Lemma 5 *Let* $L = L(d,2d)$ *and let* (u^i, v^i) *be the* i*th vertex pair of the path* $\pi(d,2d)$ *for* $0 \le i \le L = L(d,2d)$. *Then for* $1 \le i \le L-2$,

$$\begin{aligned} u^i_k &= u^{L-1-i}_{2d-k} \qquad (1 \le k \le 2d-1), && (6)\\ v^i_1 &= v^{L-1-i}_{2d} = 1, && (7)\\ v^i_k &= v^{L-1-i}_{2d-k} \qquad (2 \le k \le 2d-2), && (8)\\ u^i_{2d} &= v^i_{2d-1} = 0. && (9)\end{aligned}$$

In (u^1, v^1), *the duplicate label is* 1, *which is then dropped in* P, *and never picked up again.*

Proof. An example for the following arguments is provided by Figure 2. Equation (9) holds because the label $2d$ is missing for all $i = 1, \ldots, L-2$. After one step on $\pi(d,2d)$, the vertex pair

$$(u^1, v^1) = (1^d0^d, 10^{d-1}1^{d-2}01) \qquad (10)$$

is reached. The duplicate label is 1, which has been picked up in Q, and will next be dropped in P from u^1. The last vertex pair of $\pi(d,2d)$ is $(u^L, v^L) = e^d_1$. This is reached by picking up label $2d$ in P. The previous vertex pair is therefore $(u^{L-1}, v^{L-1}) = (0^{d-1}1^d0, 1^d0^d)$, where label d is duplicate. The vertex pair (u^{L-2}, v^{L-2}) is therefore

$$(u^{L-2}, v^{L-2}) = (0^{d-1}1^d0, 1^{d-1}0^d1), \qquad (11)$$

with duplicate label $2d-1$. This vertex pair is reached from (u^{L-3}, v^{L-3}) by picking up this label $2d-1$ in P. This describes the starting vertex pair (u^1, v^1) and ending vertex pair (u^{L-2}, v^{L-2}) of the path $B(d)$.

Equations (6), (7), and (8) are then shown by induction; for details see [31]. □

Two vertices of $G(n)$ are connected by an edge if and only if the corresponding bitstrings differ only by two substrings which are 1^k0 for one bitstring and 01^k for the other (where k is even), using the cyclic symmetry of the Gale evenness bitstrings if necessary. For example, the vertices v^0 and v^1 in Figure 2 are 00001111 and 10001101, where the substrings are those in positions $7, 8, 1$. These use the cyclic symmetry since the substrings in question involve both position $2d$ and position 1. We say that such an edge *wraps around*. If the mentioned substrings 1^k0 and 01^k are contiguous substrings of positions 1 through $2d$, the edge does *not* wrap around, like, for example, the edge connecting vertices $u^1 = 11110000$ and $u^2 = 01111000$ in Figure 2.

Lemma 6 *No edge of* $\pi(d,1)$ *wraps around in either polytope: If the edge connects* (u,v) *to* (u',v), *then the edge connecting* u *and* u' *in* P *does not wrap around, and if the edge of* $\pi(d,1)$ *connects* (u,v) *to* (u,v'), *then the edge connecting* v *and* v' *in* Q *does not wrap around.*

Proof. The first edge of $\pi(d,1)$ joining e_0 to $(01^d0^{d-1}, 0^d1^d)$ does not wrap around, and neither does the last edge joining $(0^d1^d, 01^d0^{d-1})$ to e_1. In all other edges, position 1 is zero in both polytopes, so none of these edges wraps around either. □

For any two paths π and π' on $G(n) \times G(n)$, we denote by $\pi + \pi'$ the path obtained by joining with an edge the last vertex pair of π to the first pair of π', assuming this edge exists. The length of the new path is the sum of the lengths of π and π' plus one; the number of its *vertex pairs* is simply the respective sum.

The following central theorem describes how paths $\pi(d,1)$ and $\pi(d,2d)$ are composed of other such paths, possibly from lower dimension.

Theorem 7 *Consider* $A(d) = \pi(d,1)$ *and* $B(d) = (u^1, v^1) \cdots (u^{L-2}, v^{L-2})$ *where* (u^i, v^i) *is the* i*th vertex pair of* $\pi(d,2d)$, $0 \le i \le L = L(d,2d)$. *Then there are paths* $C(d)$ *and mappings* $\alpha, \beta, \beta', \gamma, \gamma'$ *defined on vertex pairs, and extended to sequences of vertex pairs, so that*

$$\begin{aligned} A(d) &= \beta(B(d)) + C(d), && (12)\\ C(d) &= \alpha(A(d-2)) + \beta'(B(d)), && (13)\\ B(d) &= \gamma(A(d-2)) + \gamma'(C(d-2)). && (14)\end{aligned}$$

Proof. Overview: The path $C(d)$ is simply a tail segment of $A(d)$. The mappings are given as follows: β and β' are defined on $G(d) \times G(d)$,

$$\beta(u,v) = (u, 0v_2v_3 \ldots v_{2d-2}1v_{2d}),$$

and β' is determined by β due to Lemma 4. Furthermore, α, γ, γ': $G(d-2) \times G(d-2) \to G(d) \times G(d)$. With $\overleftarrow{u}$ defined as the bitstring u reversed,

$$\alpha(u,v) = (0\overleftarrow{u}110, 0\overleftarrow{v}110). \qquad (15)$$

With $c = 2d - 4$,

$$\gamma(u_1 \ldots u_c, v) = (u_1 11 u_2 \ldots u_c 00, 10v01). \qquad (16)$$

We obtain γ' from γ by Lemma 5.

First we show, equivalent to (12) and (13), that

$$A(d) = \beta(B(d)) + \alpha(A(d-2)) + \beta'(B(d)). \qquad (17)$$

Note that only positions 1 and $2d-1$ in Q (corresponding to the missing label in $A(d)$ and $B(d)$, respectively) are changed by the mapping β, and these positions are constant throughout $B(d)$ by (7) and (9). The starting point (u^1, v^1) of $B(d)$ is given by (10), and in the first step of $B(d)$ label 1 is dropped in P. The path $A(d)$ is also started by dropping label 1 in P from e_0^d. Now $\beta(u^1, v^1) = e_0^d$, as required, and in the first step of $A(d)$ and $B(d)$ the label to be dropped is 1 in P. As (u^1, v^1) and e_0^d differ only in positions that are constant throughout $B(d)$, the path $B(d)$ maps to $\beta(B(d))$ and thereby represents the initial part of $A(d)$. By (11), the endpoint of $\beta(B(d))$ is

$$\beta(0^{d-1}1^d0, 1^{d-1}0^d1) = (0^{d-1}1^d0, 01^{d-2}0^{d-1}11). \quad (18)$$

The duplicate label is $2d-1$, which has been picked up in P. So in the next step of $A(d)$, label $2d-1$ is dropped in Q and label $2d-3$ is picked up, giving the vertex pair

$$(u^*, v^*) = (0^{d-1}1^d0, 01^{d-2}0^{d-2}110). \qquad (19)$$

(For the path $\pi(d, 2d)$, label d would be picked up instead at this stage, as stated in the proof of Lemma 5.) This is the edge of $A(d)$ which joins $\beta(B(d))$ to $\alpha(A(d-2))$ in (17).

We are now at the start of $C(d)$ and want to show that this path segment starts with $\alpha(A(d-2))$ with α in (15). Indeed, the starting vertex pair of $C(d)$ is $(u^*, v^*) = \alpha(e_0^{d-2})$. The duplicate label is $2d-3$, which is to be dropped in P in the next step. The subsequent steps are represented by $\alpha(A(d-2))$ since in the lower-dimensional polytope, label 1 is dropped, which is mapped by α to label $2d-3$ of the higher-dimensional polytope, considering α also as an injective map of labels, obtained in the obvious way from (15), namely $\alpha(k) = 2d-2-k$ for $1 \le k \le 2d-4$. Essentially, the subsequent steps in $A(d-2)$ map into higher dimension by (15) and by Lemma 6; we only need to check complementarity of the constant positions in higher dimension. In the higher dimension, position 1 with the missing label 1 is zero in both polytopes, consistent with (15). Positions $2d-1$ and $2d$ are also complementary by (15). For positions $2d-3$ and $2d-2$, we have complementarity because $2d-3$ is zero as it is obtained from the position with the missing label 1 in lower dimension. This shows that the initial segment of $C(d)$ is indeed $\alpha(A(d-2))$.

In the last step of $A(d-2)$, label 1 is picked up in Q. So in the last step of $\alpha(A(d-2))$, label $2d-2$ is picked up in Q. Then we are at the vertex pair $(v^*, u^*) = \alpha(e_1^{d-2})$, which is $(01^{d-2}0^{d-2}110, 0^{d-1}1^d0)$ by (19). We have shown that the initial part of $A(d)$ in (17) is $\beta(B(d)) + \alpha(A(d-2))$ and that the starting point and endpoint of $\alpha(A(d-2))$ are (u^*, v^*) and (v^*, u^*), respectively. Then the rest of the path $A(d)$ in (17) is obtained by Lemma 4: The next vertex pair, obtained from (v^*, u^*) by dropping label $2d-2$ in P, is

$$(u', v') = (01^{d-2}0^{d-1}11, 0^{d-1}1^d0), \qquad (20)$$

which agrees with Lemma 4 and (18). Thus, the remainder is the path $\beta(B(d))$ backwards but with the bitstrings for P and Q exchanged. However, using the symmetry of $B(d)$ in Lemma 5, this part of the path can be expressed as $\beta'(B(d))$ with a suitably defined mapping β', similar to β, which exchanges the bitstrings for P and Q. This shows (17).

We now show (14). The first part of $B(d)$ is indeed $\gamma(A(d-2))$: Both $B(d)$ and $A(d-2)$ start by dropping label 1 in P, and the starting point of $B(d)$ is $\gamma(e_0^{d-2})$. Then $B(d)$ proceeds like $\gamma(A(d-2))$ because of Lemma 6 and since complementarity holds for the constant positions in higher dimension, which is easily checked using (16). Next, by (17),

$$\gamma(A(d-2)) = \gamma[\beta(B(d-2)) + \alpha(A(d-4)) + \beta'(B(d-2))].$$

Now consider the starting point (u'', v'') of $\beta'(B(d-2))$, which is (u', v') given by (20) but with $d-2$ instead of d. Furthermore, consider the endpoint of $\beta'(B(d-2))$, that is, the endpoint e_1^{d-2} of $A(d-2)$. The images of these points under γ are

$$\begin{aligned}\gamma(u'', v'') &= \gamma(01^{d-4}0^{d-3}11, 0^{d-3}1^{d-2}0)\\ &= (01^{d-2}0^{d-3}1100, 10^{d-2}1^{d-2}001),\\ \gamma(e_1^{d-2}) &= \gamma(0^{d-2}1^{d-2}, 1^{d-2}0^{d-2})\\ &= (0110^{d-3}1^{d-2}00, 101^{d-2}0^{d-1}1).\end{aligned}$$

This shows that these two vertex pairs $\gamma(u'', v'')$ and $\gamma(e_1^{d-2})$ are mirror images of each other under the symmetry of $B(d)$ described in Lemma 5. This means that the endpoint $\gamma(e_1^{d-2})$ of $\gamma(A(d-2))$ is already in the second half of $B(d)$. The central part of $B(d)$, given by the last part of $\gamma(A(d-2))$, is $\gamma[\beta'(B(d-2))]$. Therefore, there is a mapping γ' so that

$$B(d) = \gamma[\beta(B(d-2)) + \alpha(A(d-4)) + \beta'(B(d-2))] + \gamma'[\alpha(A(d-4)) + \beta'(B(d-2))],$$

because the paths $A(d-4)$ and $B(d-2)$ are symmetric and therefore do not have to be written backwards. This representation of $B(d)$ is equivalent to (14) as claimed. □

Let a_n be the number of vertex pairs of $A(2n)$, which is one more than the length $L(2n,1)$ of that path. Let b_n and c_n be that number for $B(2n)$ and $C(2n)$, respectively. That is, for $n \geq 1$,

$$a_n = L(2n,1) + 1, \qquad b_n = L(2n,4n) - 2\,. \tag{21}$$

Then the concatenation of paths in (12) implies $a_n = b_n + c_n$, in (13) $c_n = a_{n-1} + b_n$, and in (14) $b_n = a_{n-1} + c_{n-1}$. Moreover, the paths $\pi(2,1)$ and $\pi(2,4)$ have length $4 = a_1 - 1 = b_1 + 2$. This shows that the numbers $b_1, c_1, a_1, b_2, c_2, a_2, \ldots$ are the Fibonacci numbers $2, 3, 5, 8, 13, 21, \ldots$ given by

$$f_0 = 1, \qquad f_1 = 2, \qquad f_{n+1} = f_n + f_{n-1} \tag{22}$$

for $n \geq 1$, that is,

$$a_n = f_{3n}, \qquad b_n = f_{3n-2}\,. \tag{23}$$

So both the lengths of $\pi(d,1)$ and of $\pi(d,2d)$ for even $d = 2n = 2,4,6,\ldots$ are given by every third Fibonacci number (minus one and plus two, respectively). These are the longest paths. They occur several times, since $L(d,1) = L(d,d)$ and $L(d,d+1) = L(d,d+2) = L(d,2d-1) = L(d,2d)$. This is due to the symmetry of the Gale evenness condition and of the labelings. Other paths $\pi(d,k)$ are given as sums of these paths in lower dimension. They are characterized, for all possible dropped labels k, in the following theorem.

Theorem 8 *The LH path lengths for any dropped label are characterized by* (21), (22), (23), *and*

(a) $L(d,k) = L(d,d-k+1)$ *and* $L(d,d+k) = L(d,2d-k+1)$ *for* $1 \leq k \leq d$*;*

(b) $L(d,k) = L(d,k+1)$ *for* $2 \leq k \leq d-2$ *and even* k, *and* $d+1 \leq k \leq 2d-1$ *and odd* k*;*

(c) $L(d,k) = L(k,1) + L(d-k,1)$ *for* $2 \leq k \leq d-2$ *and even* k*;*

(d) $L(d,d+k) = L(k,2k) + L(d-k+2, 2(d-k+2)) - 4 = b_{k/2} + b_{d/2-k/2+1}$ *for* $4 \leq k \leq d-2$ *and even* k.

Proof. Overview. Claim (a) is proved using a cyclic shift by d of each string in $G(d)$ followed by a reversal, which leaves $G(d)$ invariant and is compatible with the labelings l and l'. Claim (b) is proved like Lemma 4. For (c), the paths $A(k)$ and $A(d-k)$ are concatenated with extension mappings similar to (12), (13), (14). A similar argument applies to (d) using the paths $B(k)$ and $B(d-k+2)$. Using (b), cases (c) and (d) cover all possible dropped labels. For details see [31]. □

It is easy to see that the shortest path lengths are obtained as follows: If d is divisible by four, that is, $d/2$ is even, then the shortest path length occurs when dropping label $d/2$, and is given by $L(d,d/2) = 2a_{d/2} - 2$ according to Theorem 8(c). If $d/2$ is odd, then the shortest path length occurs for dropped label $3d/2$, where $L(d,3d/2) = L(d,3d/2+1) = 2b_{d/2+1}$ by Theorem 8(b) and (d). When $d/2$ is even, the path when dropping label $3d/2$ is only two steps longer than when dropping label $d/2$ since then $L(d,3d/2) = b_{d/2} + b_{d/2+1} = b_{d/2} + a_{d/2} + c_{d/2} = 2a_{d/2}$. Therefore, the shortest path results essentially when dropping label $3d/2$.

The Fibonacci numbers in (22) are given by the well-known explicit expression $f_n = K\theta^n + \overline{K}\,\overline{\theta}^{\,n}$ with $\theta, \overline{\theta} = 0.5 \pm 0.5\sqrt{5}$ and $K, \overline{K} = 0.5 \pm 0.3\sqrt{5}$, where $\theta = 1.618\ldots$ is the Golden Ratio. Then f_n is $K\theta^n$ rounded to the nearest integer since $\overline{K}\,\overline{\theta}^{\,n}$ is less than 0.5 and at any rate exponentially small. By Theorem 8(d), the sequence of shortest LH path lengths $L(2n,3n)$ for $n = d/2 = 1,2,3,\ldots$ is $4, 10, 16, 42, 68, 178, \ldots$, which is the sequence of Fibonacci numbers with every third number omitted, times two. These shortest lengths grow with the square root of the longest lengths, which is still exponential.

Corollary 9 *There are* $d \times d$ *games, for even* d, *where each LH path has length* $\Omega(\theta^{3d/4})$.

References

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

[2] Borodin, A., and R. El-Yaniv (1998), *Online Computation and Competitive Analysis*. Cambridge Univ. Press, Cambridge.

[3] Conitzer, V., and Sandholm, T. (2003), Complexity results about Nash equilibria. In: *Proc. 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, 765–771.

[4] Cottle, R. W., J.-S. Pang, and R. E. Stone (1992), *The Linear Complementarity Problem*. Acad. Press, San Diego.

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

[6] Gale, D. (1963), Neighborly and cyclic polytopes. In: *Convexity*, Proc. Symposia in Pure Math., Vol. 7, ed. V. Klee, American Math. Soc., Providence, Rhode Island, 225–232.

[7] Garcia, C. B., and W. I. Zangwill (1981), *Pathways to Solutions, Fixed Points, and Equilibria*. Prentice-Hall, Englewood Cliffs.

[8] Gilboa, I., and E. Zemel (1989), Nash and correlated equilibria: some complexity considerations. *Games and Economic Behavior* **1**, 80–93.

[9] Grünbaum, B. (2003), *Convex polytopes, 2nd ed.* Springer, New York.

[10] Fathi, Y. (1979), Computational complexity of LCPs associated with positive definite symmetric matrices. *Mathematical Programming* **17**, 335–344.

[11] Jansen, M. J. M. (1981), Maximal Nash subsets for bimatrix games. *Naval Res. Logistics Quarterly* **28**, 147–152.

[12] Kalai, G., and D. J. Kleitman (1992), A quasi-polynomial bound for the diameter of graphs of polyhedra. *Bull. Amer. Math. Soc.* **26**, 315–316.

[13] Klee, V., and P. Kleinschmidt (1987), The d-step conjecture and its relatives. *Math. of Oper. Res.* **12**, 718–755.

[14] Klee, V., and G. J. Minty (1972), How good is the simplex algorithm? In: *Inequalities, III*, Proc. Third Sympos., UCLA, 1969, ed. O. Shisha, Academic Press, New York, 159–175.

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

[16] Lemke, C. E. (1965), Bimatrix equilibrium points and mathematical programming. *Manag. Science* **11**, 681–689.

[17] Lipton, R. J., E. Markakis, and A. Mehta (2003), Playing large games using simple strategies. *Proc. 4th ACM conf. Electronic Commerce*, San Diego, 36–41.

[18] McKelvey, R. D., and A. McLennan (1996), Computation of equilibria in finite games. In: *Handbook of Computational Economics, Vol. I*, eds. H. M. Amman, D. A. Kendrick, and J. Rust, Elsevier, Amsterdam, 87–142.

[19] Megiddo, N. (1986), On the expected number of linear complementarity cones intersected by random and semi-random rays. *Mathematical Programming* **35**, 225–235.

[20] Megiddo, N., and C. H. Papadimitriou (1991), On total functions, existence theorems and computational complexity (Note). *Theoretical Computer Science* **81**, 317–324.

[21] Morris, W. D., Jr. (1994), Lemke paths on simple polytopes. *Math. of Oper. Res.* **19**, 780–789.

[22] Murty, K. G. (1978), Computational complexity of complementary pivot methods. *Mathematical Programming Study* **7**: *Complementary and fixed point problems*, 61–73.

[23] Nash, J. F. (1951), Non-cooperative games. *Annals of Mathematics* **54**, 286–295.

[24] Nisan, N., and A. Ronen (2001), Algorithmic mechanism design. *Games and Economic Behavior* **35**, 166–196. [Extended Abstract in *Proc. 31st STOC* (1999), 129–140.]

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

[26] Papadimitriou, C. H. (1994), *Computational Complexity*. Addison-Wesley, Reading, Mass.

[27] Papadimitriou, C. H. (2001), Algorithms, games, and the internet. In: *Proc. 33rd STOC*, 749–753.

[28] Rosenmüller, J. (1971), On a generalization of the Lemke–Howson algorithm to noncooperative N-person games. *SIAM J. Appl. Math.* **21**, 73–79.

[29] Roughgarden, T. (2002), *Selfish Routing*. PhD thesis, Cornell University.

[30] Savani, R. (2004), Challenge instances for NASH. Research Report LSE-CDAM-2004-14, London School of Economics.

[31] Savani, R., and B. von Stengel (2004), Exponentially many steps for finding a Nash equilibrium in a bimatrix game. Research Report LSE-CDAM-2004-03, London School of Economics.

[32] Shapley, L. S. (1974), A note on the Lemke–Howson algorithm. *Mathematical Programming Study* **1**: *Pivoting and Extensions*, 175–189.

[33] von Stengel, B. (1999), New maximal numbers of equilibria in bimatrix games. *Discrete and Computational Geometry* **21**, 557–568.

[34] von Stengel, B. (2002), Computing equilibria for two-person games. Chapter 45, *Handbook of Game Theory, Vol. 3*, eds. R. J. Aumann and S. Hart, North-Holland, Amsterdam, 1723–1759.

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

[36] Vorob'ev, N. N. (1958), Equilibrium points in bimatrix games. *Theory of Probability and its Appl.* **3**, 297–309.

[37] Wilson, R. (1971), Computing equilibria of N-person games. *SIAM J. Appl. Math.* **21**, 80–87.

[38] Winkels, H.-M. (1979), An algorithm to determine all equilibrium points of a bimatrix game. In: *Game Theory and Mathematical Economics*, eds. O. Moeschlin and D. Pallaschke, North-Holland, Amsterdam, 137–148.

[39] Yao, A. C. (1977), Probabilistic computation: towards a unified measure of complexity. In: *Proc. 18th FOCS*, 222–227.

[40] Ziegler, G. M. (1995), *Lectures on Polytopes*. Springer, New York.