Finding Nash Equilibria of Bimatrix Games

Rahul S. J. Savani

London School of Economics and Political Science

PhD

Abstract

This thesis concerns the computational problem of finding one Nash equilibrium of a bimatrix game, a two-player game in strategic form. Bimatrix games are among the most basic models in non-cooperative game theory, and finding a Nash equilibrium is important for their analysis.

The Lemke–Howson algorithm is the classical method for finding one Nash equilibrium of a bimatrix game. In this thesis, we present a class of square bimatrix games for which this algorithm takes, even in the best case, an exponential number of steps in the dimension d of the game. Using polytope theory, the games are constructed using pairs of dual cyclic polytopes with $2d$ suitably labelled facets in d-space. The construction is extended to two classes of non-square games where, in addition to exponentially long Lemke–Howson computations, finding an equilibrium by support enumeration takes exponential time on average.

The Lemke–Howson algorithm, which is a complementary pivoting algorithm, finds at least one solution to the linear complementarity problem (LCP) derived from a bimatrix game. A closely related complementary pivoting algorithm by Lemke solves more general LCPs. A unified view of these two algorithms is presented, for the first time, as far as we know. Furthermore, we present an extension of the standard version of Lemke's algorithm that allows one more freedom than before when starting the algorithm.

Contents

List of Figures

Acknowledgements

I am indebted to my PhD supervisor Bernhard von Stengel. His excellent supervision has been invaluable. He has always been available with advice, and co-authoring papers with him has been an extremely instructive and rewarding experience. Most of all, I have had great fun working with him. I must also thank Bernhard for introducing me to many of the people whose research is cited in this thesis.

I thank the participants of the Bellairs Workshop on "Polytopes, Games, and Matroids" in Barbados in March 2003, in particular the organizer, Komei Fukuda, and Walter Morris and Jörg Rambau, for stimulating discussions. I also thank Walter Morris, David Avis, Bernd Gärtner, Srihari Govindan, and Andrew McLennan for their hospitality and generous support.

I thank the members of the Mathematics Department for all their support. In particular, I thank Jackie Everid, David Scott, Jan van den Heuvel, Graham Brightwell, Mark Baltovic, Arndt von Schemde, and Nic Georgiou, who have always been there for me.

I would also like to thank the London School of Economics and Political Science (LSE), the Department of Mathematics at LSE, and the UK Engineering and Physical Sciences Research Council (EPSRC) for financial support.

Last but not least, I thank my family, especially my wife Raya, for all their love and encouragement.

Chapter 1

Introduction

In Section 1.1, we briefly describe some background material required for understanding this introductory chapter. In Section 1.2, we describe the contribution of this thesis. In Section 1.3, we describe the structure of this thesis, and give details of where some of the results have already been published. In Section 1.4, we describe related work.

1.1 Background

This thesis concerns the computational problem of finding one Nash equilibrium of a bimatrix game; we call this problem NASH. Bimatrix games are among the most basic models in non-cooperative game theory, and finding a Nash equilibrium is important for their analysis.

A bimatrix game is a two-player game in strategic form. The strategic form is specified by a finite set of players, 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 strategic form game has at least one equilibrium in mixed strategies (Nash (1951)). 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).

In computer science, the running time of a program is measured as a function of the size of the input; for the problem NASH, this is the number of bits required to specify the payoff matrices. Problems that can be solved in *polynomial* running time are considered as "computationally tractable" (see Garey and Johnson (1979), Papadimitriou (1994a)). At present, it is not known whether NASH can be solved in polynomial time. This has been called one of the two "most important concrete open questions" in theoretical computer science (Papadimitriou (2001)). A recent preprint of Chen and Deng (2005b), discussed in Section 5.1, suggests that a polynomial-time algorithm for NASH is unlikely to exist. Currently, not even a subexponential algorithm for the problem is known.

An equilibrium of a *zero-sum* bimatrix game (A,B), where $B = -A$, is the solution to a linear program (LP). Linear programs can be solved in polynomial time by the ellipsoid method or interior point methods (see Todd (2001) for a survey). No such method is known for finding Nash equilibria. One difficulty is that the set of Nash equilibria of a bimatrix game is generally not convex. Problems relating to that set tend to be computationally difficult, for example the problem of deciding whether a game has a unique Nash equilibrium (Gilboa and Zemel (1989), Conitzer and Sandholm (2003), Codenotti and Štefankovič (2005)). Finding all equilibria is therefore computationally intractable for larger games.

1.2 The contribution of this thesis

A standard method for NASH is the algorithm due to Lemke and Howson (1964), here called the LH algorithm. The algorithm, which is described in Section 2.2, provides an elementary and constructive proof that a bimatrix game has at least one Nash equilibrium. The first result of this thesis is a class of square games where this algorithm takes an exponential number of steps, which shows that the algorithm is not polynomial. The

LH algorithm is a pivoting method related to the simplex algorithm for linear programming (Dantzig (1963)). Klee and Minty (1972) constructed linear programs for which the simplex method with a certain pivot rule takes an exponential number of pivoting steps; similar examples have been constructed for other pivot rules (see Klee and Kleinschmidt (1987), Todd (2001)). The LH algorithm has no choice of its pivot rule, but a free choice, corresponding to a pure strategy of one of the players, of its first step (the first variable "to enter the basis"). In the games constructed here, the running time of the LH algorithm is exponential even for the *best* first choice of the algorithm. 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.

Our construction uses the theory of polyhedra (see for example Ziegler (1995) or Grünbaum (2003)). This geometric view gives a good insight into the structure of Nash equilibria of two-player games (see von Stengel (2002)). For each player, the set of his mixed strategies together with the best response payoff to the other player is described by a polyhedron. A vertex of each polyhedron is obtained by converting some inequalities into equations, which describe the support of a mixed strategy and its best responses. An equilibrium is given by a "complementary" vertex pair. The LH algorithm traverses "almost complementary" edges of the polyhedra until it reaches an equilibrium.

We use a standard construction of "dual cyclic polytopes". These are polyhedra for which the vertex-defining inequalities are known in arbitrary dimension (Gale (1963)). This purely combinatorial information can also be used to construct bimatrix games with many equilibria (von Stengel (1999)). In our construction, since both players have dual cyclic polytopes as their best response polyhedra, we call these $m \times n$ games the *double cyclic polytope games* $\Gamma(m,n)$, and in particular the square games just mentioned $\Gamma(d,d)$.

The LH paths are defined purely combinatorially in terms of the supports of, and best responses to, the mixed strategies that they trace. These correspond to known binary patterns that encode the vertices of dual cyclic polytopes. Linear recurrences for the various path lengths give rise to their exponential growth. For $d = 2,4,6,\ldots$, the length of the longest path for a $d \times d$ game $\Gamma(d,d)$ is given by every third Fibonacci number, which is proportional to $\phi^{3d/2}$, where $\phi = 1.618...$ is the Golden Ratio. Shorter path lengths are obtained by certain sums of these numbers, the shortest length being proportional to $\phi^{3d/4}$.

The construction described so far is closely related to Morris (1994), who used dual cyclic polytopes to produce exponentially long "Lemke paths" on polytopes. These paths correspond to a "symmetric" version of the LH algorithm, which finds symmetric equilibria of a symmetric game (although Morris did not consider this interpretation). The games obtained from Morris's construction have additional nonsymmetric equilibria. The standard LH algorithm always terminates, very quickly, at such a nonsymmetric equilibrium. These games are therefore not directly useful for our purpose of providing "challenge instances" for the problem NASH. After we found our construction of long LH paths for the bimatrix games $\Gamma(d,d)$, McLennan and Tourky (2005) noted an ingenious alternative proof of exponentially long LH paths based on Morris's construction and *imitation games*, as explained in Section 3.7. An imitation game is a square game (A,I), where I is the identity matrix. It is closely related to the symmetric game $(A,A^\top)$, since its *symmetric* equilibria correspond to the equilibrium strategies of player 2 in the imitation game.

The square games $\Gamma(d,d)$ have a unique, completely mixed equilibrium, as do the square games derived from Morris (1994) by McLennan and Tourky (2005). In both cases, the LH algorithm needs exponential time to find this equilibrium. However, for both games the equilibrium is quickly found by a simple algorithm called *support enumeration* (e.g., Dickhaut and Kaplan (1991), Porter, Nudelman, and Shoham (2004), or Bárány, Vempala, and Vetta (2005)). The support of a mixed strategy is the set of pure strategies that it plays with positive probability. All pure strategies in the support of an equilibrium strategy must have equal expected payoff. If, as here, the game is nondegenerate, the corresponding linear equations uniquely determine the mixed strategy probabilities of the other player. Support enumeration considers possible supports for both players and the solutions to the respective linear equations, and checks if these define an equilibrium. In a square game, it is natural to test the set of all pure strategies as an equilibrium support, which gives the equilibrium in the two classes of square games mentioned so far. That is, both our square games $\Gamma(d,d)$ and the square imitation games derived from Morris's construction are not "hard to solve" by other methods.

Therefore, we extend each of these two constructions to give two classes of non-square games where *both* the LH algorithm and, on average, support enumeration are

exponential. It appears that no general algorithm for NASH is known that can solve either of these classes of games efficiently, so we consider them both as classes of *hard-to-solve bimatrix games*. The first class are the $d \times 2d$ double cyclic polytope games $\Gamma(d, 2d)$, with the columns of the payoff matrices randomly permuted. For the second class, we construct $3d \times d$ games $\mathrm{T}(d)$ with one dual cyclic polytope and a *simplotope*, which is a product of simplices, here of d tetrahedra. This construction is inspired by Morris (1994) and McLennan and Tourky (2005). We call these games *triple imitation games*, due to their connection to imitation games.

One reason for presenting two classes is that there is a *tradeoff* between them. The games $\Gamma(d, 2d)$ have the longer LH paths, whereas the triple imitation games $\mathrm{T}(d)$ will take a greater expected number of attempts to find a Nash equilibrium by support enumeration. A further reason for including triple imitation games is the use of a new kind of polytope, the simplotope, as one of the best response polytopes for constructing games.

Another method of finding equilibria is to enumerate the vertex pairs of the best response polyhedra, and test them for the equilibrium property. In general, the number of vertices is exponential. However, there are many vertex enumeration methods (see e.g. Avis and Fukuda (1992) and the references therein), and we do not analyse whether they solve our games $\Gamma(d, 2d)$ or $\mathrm{T}(d)$ in expected exponential time, although we suspect that this is the case. Knowing the exact way that the games $\Gamma(d, 2d)$ and $\mathrm{T}(d)$ are constructed, it is possible to "unscramble" the hidden support by special pivoting steps, and thus arrive quickly at an equilibrium, as described in Section 3.6 for the games $\Gamma(d, 2d)$. However, this very specialized method only applies to these constructions, and does not compute an equilibrium for other games.

The Nash equilibria of a bimatrix game are solutions to a linear complementarity problem (LCP); see Cottle, Pang, and Stone (1992). The algorithm by Lemke (1965) is closely related to the LH algorithm. Whereas the LH algorithm only solves bimatrix games, Lemke's algorithm can solve more general LCPs. In Chapter 4, a unified view of these two algorithms is presented, for the first time, as far as we know. Furthermore, we present an extension of the standard version of Lemke's algorithm that allows one more freedom than before when starting the algorithm.

1.3 The structure of this thesis

In this section, we describe the structure of this thesis, and give details of where some of the results of the thesis have already been published.

Chapter 2 gives the relevant background material. In Section 2.1, we describe the *best response condition*, leading to the characterization of Nash equilibria of bimatrix games as completely labelled (complementary) vertex pairs of the best response polytopes. In Section 2.2, we describe a modification of the LH algorithm that finds a symmetric equilibrium of a symmetric bimatrix game. We explain this modified and simpler LH method because, to our knowledge, except in Savani and von Stengel (2004; 2006), it has not been described earlier (although it is straightforward), and because it leads naturally to the usual LH method.

In Chapter 3, we present two classes of hard-to-solve bimatrix games. In Section 3.2, we describe the well-understood dual cyclic polytopes, and their construction using the *moment curve*. Then, in Section 3.3, we present the class of games $\Gamma(m,n)$. In Section 3.4, we show that for the square games $\Gamma(d,d)$ the length of the shortest LH path grows exponentially with d. In Section 3.5, we show that if we randomly permute the columns of the games $\Gamma(d,2d)$, in addition to exponentially long LH computations, finding an equilibrium by support enumeration takes on average exponential time. The results in Sections 3.4 and 3.5 have been published in Savani and von Stengel (2004; 2006). In Section 3.6, we describe how to "unscramble" the random permutation and quickly find an equilibrium of $\Gamma(d,2d)$, if the rest of the construction is known. This result has appeared in Savani (2004). In Section 3.7, which is taken from Savani and von Stengel (2006), we explain how to interpret the "Lemke paths" of Morris (1994) as paths of the symmetric LH algorithm described in Section 2.2. All ordinary LH paths for these symmetric games are very short, and lead to nonsymmetric pure-strategy equilibria. We then explain the use of imitation games in this context, as suggested by McLennan and Tourky (2005), giving another class of square games with exponentially long LH paths. Extending this, we present in Section 3.8 the new $d \times 3d$ triple imitation games $\mathrm{T}(d)$, which (like the games $\Gamma(d,2d)$) are hard to solve by both support enumeration and the LH algorithm. The time complexity of the LH algorithm and support enumeration for all the mentioned classes of games is summarized in Figure 3.10.

In Chapter 4, we describe the algorithm by Lemke (1965). We then give a new, unified view of Lemke's algorithm and the LH algorithm, and an extension of Lemke's algorithm that allows one more freedom when starting the algorithm than before.

In Chapter 5, we conclude our findings and discuss open questions. Appendix A gives a number of supplementary results. In Section A.1, we describe how to generate the square games $\Gamma(d,d)$, and give an example of such a game. In doing so, we describe a second way to construct cyclic polytopes, the well-known *trigonometric moment curve* (see Ziegler (1995, p. 75f) or Grünbaum (2003, p. 67)). In Section A.2, we give tables of path lengths for all the constructed games, as well as examples of paths. A list of symbols is given at the end.

1.4 Related work

Our construction of the games $\Gamma(d,d)$ and $\mathrm{T}(d)$ is related to Morris (1994). There, Morris showed that Lemke paths cannot be used to address the Hirsch conjecture (Klee and Kleinschmidt (1987)). 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 (Kalai and Kleitman (1992)). A *polynomial* pivoting algorithm for NASH (or even for finding a symmetric Nash equilibrium of a symmetric game, using the symmetrization in (2.3) below), applied to zero-sum games, would answer that question as well.

Equilibria of zero-sum games are the solutions to an LP. The results by Murty (1980) and Goldfarb (1983; 1994) suggest that Lemke's algorithm can be exponential even when solving a zero-sum game. However, these results do not extend to the LH algorithm. First, the examples by Murty and Goldfarb define a single path of Lemke's algorithm, which is not an LH path. Second, even if they could be modified to define an LH path, the LP defines a cube like in the construction by Klee and Minty (1972). The endpoints of the exponentially long path on the cube are joined by a single edge. Hence, even if one could make the LH algorithm mimic that path, another LH path would be very short.

As mentioned above, the Nash equilibria of a bimatrix game are solutions to a linear complementarity problem (LCP). In Murty (1978) and Fathi (1979), LCPs are constructed

where for certain pivoting methods *one* path is exponentially long, in analogy to Klee and Minty (1972). In our case, *all* LH paths are exponentially long. Moreover, the examples of Murty and Fathi do not arise from games.

Equilibrium enumeration methods (see for example Vorob'ev (1958), Kuhn (1961), Winkels (1979), Jansen (1981), Audet, Hansen, Jaumard, and Savard (2001), and the survey in von Stengel (2002)) can be modified to terminate once the first equilibrium is found, and should be tested on the games constructed in this thesis 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 (Gilboa and Zemel (1989)). 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 (Garcia and Zangwill (1981)), which typically specialize to pivoting in the two-player case (for a generalization of LH to more than two players see Rosenmüller (1971) and Wilson (1971)), and algorithms for approximating fixed points (e.g. McKelvey and McLennan (1996) and Papadimitriou (1994b)).

Chapter 2

Bimatrix Games, Polytopes, and the Lemke–Howson algorithm

2.1 Nash equilibria and the best response condition

We use the following notation. 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. A mixed strategy for player 2 is such a vector y in $\mathbb{R}^n$. All vectors are column vectors; the row vector corresponding to x is written as the transpose $x^\top$. The support of a mixed strategy is the set of pure strategies that have positive probability. 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. Best responses are characterized by the following combinatorial condition, which we state only for a mixed strategy x of player 1.

Lemma 2.1 (Nash (1951)) *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.*

Proof: Let $(Ay)_i$ be the ith component of Ay, which is the expected payoff to player 1 when playing row i. Let $u = \max_i (Ay)_i$. Then

$$x^\top Ay = \sum_i x_i (Ay)_i = u - \sum_i x_i (u - (Ay)_i).$$

Since the sum $\sum_i x_i (u - (Ay)_i)$ is nonnegative, $x^\top Ay \leq u$. The expected payoff $x^\top Ay$ achieves the maximum u if and only if that sum is zero, that is, if $x_i > 0$ implies $(Ay)_i = u$, as claimed. □

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 the probability vectors, are the bottom left pure strategy pair $\big((0,1)^\top, (1,0)^\top\big)$ with expected payoffs $4, 2$ to players 1 and 2, respectively, the top right pure strategy pair $\big((1,0)^\top, (0,1)^\top\big)$ with expected payoffs $2, 4$, and the mixed strategy pair $\big((1/3, 2/3)^\top, (1/3, 2/3)^\top\big)$ with expected 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 2.1) and hence have equal expected payoff.

In a mixed equilibrium, the probabilities are uniquely given by the pair of supports if the corresponding sub-matrices 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 von Stengel (2002) for a detailed discussion). The LH algorithm can be extended to degenerate games by standard lexicographic perturbation techniques, which we discuss in Section 4.3. All of the games we construct are nondegenerate.

By Lemma 2.1, an equilibrium is given if any pure strategy of a player is either a best response (to the opponent's mixed strategy) or is played with probability zero (by the player himself). This can be captured by polytopes (see Ziegler (1995), Grünbaum (2003)) 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. We then extend this easily to nonsymmetric games. Let

$$S = \{ z \in \mathbb{R}^d \mid z \geq \mathbf{0},\ \ Cz \leq \mathbf{1} \} \tag{2.1}$$

where $\mathbf{0}$ and $\mathbf{1}$ denote vectors with all entries 0 and 1, respectively, and inequalities hold for all components. The right hand sides of the inequality $Cz \leq \mathbf{1}$, which are 1, represent *normalized* payoffs, as explained in Lemma 2.2 and subsequently. 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. 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. A facet is obtained by making one of the inequalities defining the polytope *binding*, that is, by converting it into an equality. The following lemma characterizes Nash equilibria in terms of polytope vertices, as already shown by Vorob'ev (1958).

Lemma 2.2 *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* (2.1)*, $z \neq \mathbf{0}$, $u = 1/\sum_i z_i$, and $z^\top(\mathbf{1} - Cz) = 0$, where z must be a vertex of S by nondegeneracy.*

Proof: Let $z \in S$, $z \neq \mathbf{0}$ and $u = 1/\sum_i z_i$. Then $u > 0$, and zu is a mixed strategy x. The condition $Cz \leq \mathbf{1}$ is equivalent to $Cx \leq \mathbf{1}u$. The orthogonality condition $z^\top(\mathbf{1} - Cz) = 0$ is equivalent to $x^\top(\mathbf{1}u - Cx) = 0$, so that for each positive component x_i of x (of which there is at least one), $(Cx)_i = u = \max_k (Cx)_k$. Thus, by Lemma 2.1, x is a best response to itself, that is, (x,x) is a symmetric equilibrium. Conversely, any such equilibrium (x,x), with $u = \max_k (Cx)_k > 0$, and $z = x \cdot 1/u$, gives a vector z with the stated properties.

The vector z is on d facets of S since for each i, either $z_i = 0$ or $(Cz)_i = 1$. If z was not a vertex but on a higher-dimensional face of S, any vertex of that face would be on additional facets, contradicting nondegeneracy of the game as S would then not be a simple polytope. □

In the game of chicken above, $z = (1/6, 1/3)^\top$ gives the symmetric equilibrium. The vector z has to be re-scaled 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 (see von Stengel (2002)).

2.2 The Lemke–Howson algorithm

The conditions in Lemma 2.2 define a *linear complementarity problem* (LCP) (see Cottle, Pang, and Stone (1992)), usually stated as follows: find z so that

$$z \geq \mathbf{0}, \qquad q + Mz \geq \mathbf{0}, \qquad z^\top (q + Mz) = 0, \tag{2.2}$$

here with data $M = -C$, $q = \mathbf{1}$. This LCP has a trivial solution $z = \mathbf{0}$, which is not a Nash equilibrium. However, $\mathbf{0}$ is an *artificial equilibrium*, which is the starting point of what we call the *symmetric LH algorithm*.

Given a nondegenerate symmetric game $(C, C^\top)$, the symmetric LH algorithm finds a nonzero vertex z of the polytope S in (2.1) so that $z^\top(\mathbf{1} - Cz) = 0$, giving a symmetric Nash equilibrium by Lemma 2.2.

It is useful to *label* the facets of S, as done by Shapley (1974). 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 labelled* (has all labels i), since then either $z_i = 0$ or $(Cz)_i = 1$ (or both, but this cannot occur since S is simple, so a completely labelled vertex has each label exactly once).

The LH algorithm is started from the completely labelled vertex $z = \mathbf{0}$ by choosing one label k that is initially *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 labelled 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 is 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. This is illustrated in Figure 2.1 for dimension 3.

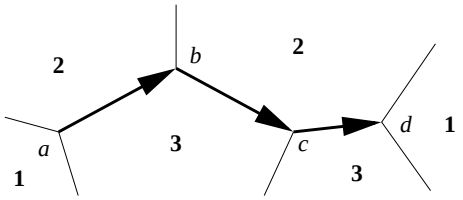

Figure 2.1 The symmetric LH algorithm in dimension 3. Point a is completely labelled, being adjacent to facets with labels $1,2,3$. Dropping label 1, it proceeds to point b picking up label 2, now duplicate. The next point is c with duplicate label 3, and finally d where the missing label 1 is picked up, which terminates the path.

As in the simplex algorithm (Dantzig (1963)), edge traversal is implemented algebraically by *pivoting* with variables entering and leaving a basis, the nonbasic variables representing the current facets. The only difference is the rule for choosing the next entering variable, which in the simplex algorithm depends on the objective function. Here, the *complementary pivoting* rule chooses the nonbasic variable with duplicate label to enter the basis.

In Figure 2.1, 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 2.1, 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 2.1, 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 (see Shapley (1974) and Garcia and Zangwill (1981)). Knowing the orientation of the artificial equilibrium, the orientation of any k-almost complementary edge can be determined directly.

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

$$z = \begin{pmatrix} x \\ y \end{pmatrix}, \qquad C = \begin{pmatrix} 0 & A \\ B^\top & 0 \end{pmatrix}. \tag{2.3}$$

The polytope S of dimension $d = m+n$ in (2.1) is then the *product* $P \times Q$ of the polytopes

$$P = \{\, x \in \mathbb{R}^m \mid x \geq \mathbf{0},\ \ B^\top x \leq \mathbf{1} \,\}, \qquad Q = \{\, y \in \mathbb{R}^n \mid Ay \leq \mathbf{1},\ \ y \geq \mathbf{0} \,\}. \tag{2.4}$$

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 2.2 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 $y = \mathbf{0}$, and then $(\mathbf{0},\mathbf{0})$ is the artificial equilibrium.

The LH algorithm is 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 by Shapley (1974). A label then represents a pure strategy that has probability zero or is a best response. A label i with $1 \leq i \leq m$ is a strategy of player 1 and determines the facet $x_i = 0$ of P or $(Ay)_i = 1$ of Q, corresponding to the respective ith inequality in (2.4) that becomes binding. A label j with $m+1 \leq j \leq m+n$ is a strategy of player 2 and determines the facet $(B^\top x)_j = 1$ of P or $y_j = 0$ of Q, which is the respective jth binding inequality in (2.4).

The LH path using 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 P is dropped in Q and vice versa. This is the standard view of the LH algorithm; for further details see von Stengel (2002).

We end this section with two similar, simple, but useful lemmas concerning the LH algorithm, which, as far as we know, have not appeared explicitly in the literature thus far, although they are implicit in the work of McLennan and Tourky (2005). Assume that P and Q are simple, which is true if the game is nondegenerate. Then, any vertex of P is defined by exactly m facets, and so has m labels. Likewise, any vertex of Q has n labels.

Lemma 2.3 *Every LH path on $P \times Q$ induces a simple path in each polytope P and Q, that is, no vertex of P or Q is ever left and visited again on an LH path.*

Proof: Suppose to the contrary that a vertex x of P is left and visited again on an LH path. This means that there are three vertex pairs (x,y), (x,y'), and (x,y'') with $y \neq y' \neq y''$ on the

LH path on $P \times Q$, and all three are k-almost-complementary, for some k in $\{1, \ldots, m+n\}$. The m labels of x define $n-1$ labels that y, y', and y'' must share. However, this is impossible, since the $n-1$ labels correspond to $n-1$ equations in n-space that define a line, which can only contain two vertices. The situation for Q is analogous. □

Lemma 2.4 *Let (x,y) be a completely labelled vertex pair of $P \times Q$. If (x,y') is part of an LH path on $P \times Q$, either $y' = y$ or (x,y) and (x,y') are connected by an edge on that LH path (i.e. y' and y connected by an edge in Q).*

Lemma 2.4 states that if a vertex x of P is part of some complementary vertex pair, then, for any LH path, it only appears as the endpoint of the induced path in P. The analogous statement is true for a vertex y of Q that is part of a complementary vertex pair.

Proof: Since (x,y') is part of an LH path, it is a k-almost-complementary vertex pair, for some k in $\{1, \ldots, m+n\}$. This means that x and y' have at least $m+n-1$ distinct labels. Either they have all $m+n$ labels and $y' = y$, or y' shares exactly one label with x, and then this is a duplicate label. Dropping this label in Q leads to the vertex y, along the edge connecting y' and y. □

Chapter 3

Hard-to-Solve Bimatrix Games

In this chapter, we construct hard-to-solve bimatrix games using the theory of polytopes. The study of bimatrix games via the best response polytopes can be traced back to the algorithm of Vorob'ev (1958) for finding all Nash equilibria of a bimatrix game (via a variant of Lemma 2.2), and Kuhn (1961), who simplified this algorithm. Important progress was made with the development of the algorithms by Lemke and Howson (1964) and Lemke (1965), which are major topics of this thesis. Shapley (1974) introduced facet labels, and described Nash equilibria of bimatrix games as completely labelled vertex pairs of the best response polytopes, which is crucial to our exposition throughout. Since then, there has been much relevant research into the theory of polytopes, LCPs, and computational complexity.

For constructing bimatrix games with certain properties, polytopes with a known combinatorial structure are useful. In a $d \times d$ game where each player's payoff matrix is the identity matrix, there are $2^d - 1$ many equilibria, which are all symmetric, corresponding to the non-empty subsets of $\{1, \ldots, d\}$. Both the best response polytopes are cubes, and in each polytope every vertex is part of a complementary vertex pair, with one vertex, the origin, corresponding to the artificial equilibrium. We explain the case $d = 3$ in full detail in Section 3.1. Quint and Shubik (1997) conjectured that these games achieve the maximal number of equilibria for any $d \times d$ square game, and proved it for $d \leq 3$. Keiding (1997), and independently McLennan and Park (1999), showed that it is true for $d = 4$, but the conjecture was refuted by von Stengel (1999), who constructed $d \times d$ games with more than 2^d many equilibria for $d \geq 6$, by labelling pairs of dual cyclic polytopes;

the case $d = 5$ is open. In this context, dual cyclic polytopes have two attractive features: their combinatorial structure is known in arbitrary dimension (Gale (1963)); for a polytope with a fixed number of facets they have the most number of vertices possible, according to the upper bound theorem of McMullen (1970). In this chapter, we use not only dual cyclic polytopes, but also simplotopes, which are the product of simplices, thus generalizing cubes.

In Section 3.1, we give an introduction to constructing games with labelled polytope pairs. In Section 3.2, we describe the well-understood dual cyclic polytopes. In Section 3.3, we construct the double cyclic polytope games $\Gamma(m,n)$, by labelling a pair of dual cyclic polytopes. In Section 3.4, we show that for the square games $\Gamma(d,d)$ the LH algorithm takes exponentially many steps for any dropped label. In Section 3.5, we show that, as well as producing long LH computations, the games $\Gamma(d,2d)$ also take on average exponential time to solve by support enumeration. This requires that the columns of the game $\Gamma(d,2d)$ are randomly permuted. We explain the square case $\Gamma(d,d)$ first, since the LH paths of the games $\Gamma(d,2d)$ are easily described in terms of those of $\Gamma(d,d)$. In Section 3.6, we describe how to "unscramble" the random permutation and quickly find an equilibrium of $\Gamma(d,2d)$, if the rest of the construction is known. In Section 3.7, we explain how to interpret the "Lemke paths" of Morris (1994) as paths of the symmetric LH algorithm described in Section 2.2. All ordinary LH paths for these symmetric games are very short, and lead to nonsymmetric pure-strategy equilibria. We then explain the use of imitation games in this context, as suggested by McLennan and Tourky (2005), giving another class of square games with exponentially long LH paths. Extending this, in Section 3.8, we present triple imitation games $\mathrm{T}(d)$, which are $3d \times d$ games that (like the games $\Gamma(d,2d)$) are hard to solve by both support enumeration and the LH algorithm. In fact, a pair of labelled dual cyclic polytopes may not have any completely labelled vertex pairs at all, and in Section 3.9 we give an example of a class of such labelled dual cyclic polytope pairs.

3.1 Constructing bimatrix games with polytopes

We construct games by defining P and Q in (2.4). These polytopes have a special form, where the first m inequalities in P and the second n inequalities in Q are nonnegativities. This is not restrictive, since it is only the combinatorial structure of the polytopes that matters. The first building block for constructing an $m \times n$ bimatrix game is a pair of polytopes P' and Q' of known combinatorial structure, with appropriate dimensions and number of facets; P' is in dimension m with $m+n$ facets, and Q' is in dimension n with $m+n$ facets. The $m+n$ facets of each of the polytope pairs are labelled with $1,\ldots,m+n$, which correspond to the pure strategies of the game. If the labellings gives rise to at least one completely labelled vertex pair, then the polytopes can be brought into the special form (2.4). The special form is achieved via an affine transformation, described in Proposition 3.1, which is the final step in the construction. A completely labelled vertex pair is chosen, and mapped by the affine transformation to the origin (the artificial equilibrium of the LH algorithm), as described in Proposition 3.1. Then, all the other completely labelled vertex pairs correspond to Nash equilibria.

Thus, we are only interested in the combinatorial structure of the polytopes, as encoded by their vertex sets. Throughout this chapter, we represent vertices as bitstrings. For a simple polytope in dimension d with f facets, each vertex is represented by a bitstring of length f with exactly d bits that are 1, where the ith bit, which corresponds to the ith facet, is 1 if and only if the vertex lies on the ith facet.

The labellings of the polytopes determine the order in which the facet-defining inequalites appear in the game matrices. This is illustrated in Proposition 3.1 and the following simple example. The example uses cubes and is thus also helpful for understanding the construction of triple imitation games in Section 3.8, which uses simplotopes, a generalization of cubes. For simplicity, we start with polytopes that already have the special form of (2.4). After the example, we explain the affine transformation required to contruct game matrices using general polytopes. We construct 3×3 games in which the polytopes P and Q are both combinatorial 3-cubes. Since $m = n = 3$, we have $f = 3+3 = 6$. To represent the vertices of the 3-cubes as bitstrings, we need an order of the facets that will correspond to the order of the positions in the bitsrings. By the symmetry of cubes, we need only know which facets are "opposite" each other. Suppose that the facet corre-

sponding to position i in the bitstring is opposite the facet corresponding to position $i+3$ for $i = 1,2,3$. Thus, for any bitstring u representing a vertex of the 3-cube, we have

$$u_i = 1 \iff u_{i+3} = 0 \text{ for } i = 1,2,3. \tag{3.1}$$

So a bitstring u represents a vertex of the 3-cube if and only if

$$u \in \{111000, 011100, 101010, 110001, 100011, 010101, 001110, 000111\}.$$

It is easy to directly write down inequalities defining a cube in dimension d, where d of the $2d$ facets are nonnegativities namely, $0 \leq x_i \leq 1$, for $i = 1,2,3$. Thus, we do not need to use an affine transformation to construct the game matrices, and $P' = P$ and $Q' = Q$. The artificial equilibrium is $(\mathbf{0},\mathbf{0})$, and so the first three inequalities in P are $x \geq \mathbf{0}$, which have labels 1,2,3, respectively, and the last three inequalities in Q are $y \geq \mathbf{0}$, which have labels 4,5,6, respecitvely. Now we label the remaining facets of P and Q. Without loss of generality, we label inequality $x_i \leq 1$ of P with label $i+3$ for $i = 1,2,3$. We consider three different labellings for Q.

(a) First, suppose that P and Q are labelled *identically*. Then every vertex of P is part of a completely labelled vertex pair, with the vertex in Q being the complement of the vertex in P'. For example, $(u,v) = (011100, 100011)$ is one of the eight completely labelled vertex pairs, with u having labels $\{2,3,4\}$ and v having labels $\{1,5,6\}$. The corresponding game has seven Nash equilibria, and the best response polytopes are

$$P = \{x \in \mathbb{R}^3 \mid x \geq \mathbf{0},\ Ix \leq \mathbf{1}\}, \qquad Q = \{y \in \mathbb{R}^3 \mid Iy \leq \mathbf{1},\ y \geq \mathbf{0}\},$$

where I as the identity matrix in $\mathbb{R}^{3\times 3}$. The corresponding game is shown in Figure 3.2.

(b) We now change the labelling fo Q' so that the corresponding game has a unique completely mixed equilibrium. The labelling of Q' is a permutation relative to the labelling of P'. The permutation keeps labels $4,5,6$ fixed, and maps label 1 to 3, 2 to 1, and 3 to 2. It is easy to check that the only completely labelled vertex pairs are $(u,v) = (111000, 000111)$ and $(u,v) = (000111, 111000)$, so the game has a unique completely mixed equilbrium, where each player plays all his strategies with positive probability.

(c) We consider one more labelling of Q. Again, facets $4,5,6$ get labels $4,5,6$ respectively. Facet 1 gets label 1, facet 2 gets label 3 and facet 3 gets label 2. It is easy to check that the set of completely labelled vertex pairs is

$$\{(111000,000111),(000111,111000),(100011,011100),(011100,100011)\}\ .$$

The polytopes P and Q with labels are illustrated in Figure 3.1.

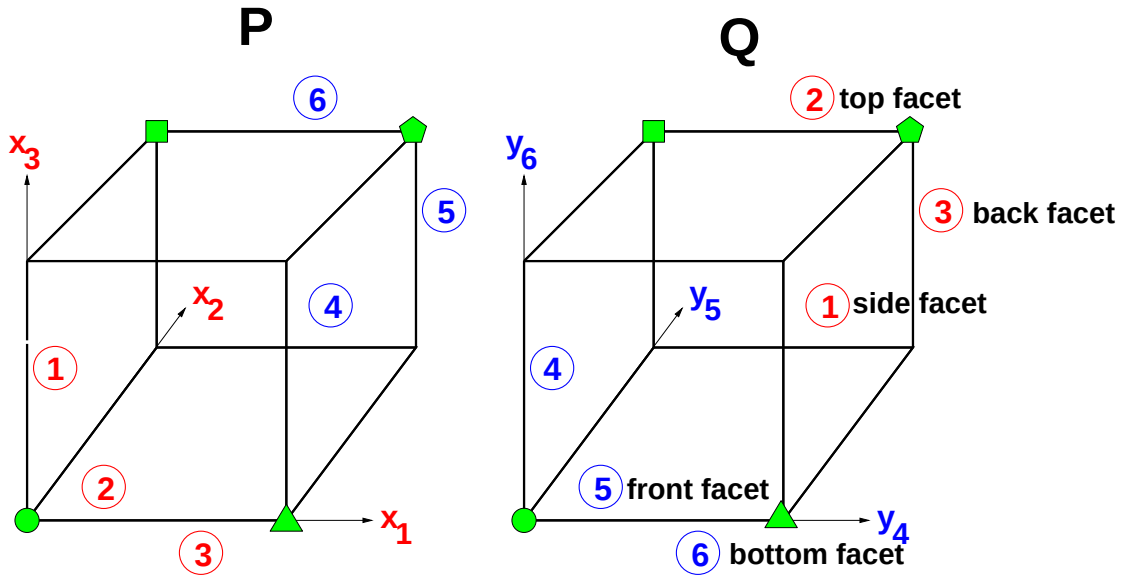

Figure 3.1 A pair of labelled 3-cubes with four completely labelled vertex pairs, which are highlighted. The labellings correspond to example (c). If the labels 3 and 2 were swapped in polytope in Q, every vertex is part of a completely labelled vertex pair, and the corresponding game has seven Nash equilibria (example (a)). If the labels 1 and 3 are swapped in Q, there are two compltely labelled vertex pairs and the corresponding game has a unique Nash equilirium (example (b)).

How do the different labelling affect the game matrices. Since we only change the labels of Q, only the payoff matrix A changes. The ith row of A corresponds to label i. Thus, the inequality $(Ay)_i$ should be that inequality that has label i. In each case the payoff matrix B is just the identity matrix, and the payoff matrices for player 1 are identity matrices with the rows permuted. The examples (a) and (b) easily generalize to arbitrary dimension, to give $d \times d$ games with $2^d - 1$ and 1 equilibrium, respectively. Notice that however we label a pair of d-cubes with $2d$ facets, the corresponding game will always have a completely mixed equilibrium.

For a pair of labelled polytopes that do not have the special form of (2.4), the labelling of the polytopes determines the order of the facet-defining equalities in P' and

(a)

I \ II	4	5	6
1	1, 1	0, 0	0, 0
2	0, 0	1, 1	0, 0
3	0, 0	0, 0	1, 1

(b)

I \ II	4	5	6
1	1, 0	0, 0	0, 1
2	0, 1	1, 0	0, 0
3	0, 0	0, 1	1, 0

(c)

I \ II	4	5	6
1	1, 1	0, 0	0, 0
2	0, 0	1, 0	0, 1
3	0, 0	0, 1	1, 0

Figure 3.2 Game (a), which corresponds to two identically labelled 3-cubes, has seven Nash equilibira. Game (b) has a unique completely mixed equilibrium. Game (c) has three equilibria. In each game, $B^\top = I$, and the matrix A is an identity matrix with permuted rows.

Q'. To get the payoff matrices A and B, we apply the affine transformation described in Proposition 3.1.

Proposition 3.1 (von Stengel (1999)) *Let P' be a simple m-polytope and Q' be a simple n-polytope, both with $m+n$ labelled facets, which have at least one complementary pair (x', y') of vertices. Then there are $m \times n$ matrices A and B defining P and Q in (2.4), a permutation of the labels $1, \ldots, m+n$ of P' and Q' yielding the labels of P and Q, and invertible affine transformations from P' to P and from Q' to Q that map (x', y') to $(\mathbf{0}, \mathbf{0})$. Furthermore, every complementary vertex pair of (P', Q') except (x', y') represents a Nash equilibrium of the bimatrix game (A, B).*

Proof: Permute the labels $1, \ldots, m+n$ in the same way for P' and Q' such that x' has labels $1, \ldots, m$ and y' has labels $m+1, \ldots, m+n$. This does not change the complementary pairs of (P', Q'). Let

$$P' = \{z \in \mathbb{R}^m \mid Cz \leq p,\ Dz \leq q\}$$

where $Cz \leq p$ represents the m inequalities for the facets $1, \ldots, m$ and $Dz \leq q$ the remaining n inequalities. For the vertex x', we have $Cx' = p$ and $Dx' < q$ since P' is simple. The m binding inequalities for x' are linearly independent since x' is a vertex, so C is invertible and $z \mapsto x = -Cz + p$ is an affine transformation with inverse $z = -C^{-1}(x - p)$. Let $P = \{x \in \mathbb{R}^m \mid -C^{-1}(x - p) \in P'\}$. Then, with $r = q - DC^{-1}p$,

$$P = \{x \in \mathbb{R}^m \mid -x \leq \mathbf{0},\ -DC^{-1}x \leq r\}.$$

Corresponding points of P and P' have the same labels. Since the vertex $\mathbf{0}$ of P corresponds to x' in P', $\mathbf{0} < r$. Thus, the jth row of $-DC^{-1}x \leq r$ can be normalized by

multiplication with the scalar $1/r_j$, so we can assume $r = \mathbf{1}$. Then P is defined as in (2.4) with the $n \times m$ transposed payoff matrix $B^\top = -DC^{-1}$. Similarly, we can find an $m \times n$ matrix A so that Q in (2.4) is an affine transform of Q'. The complementary vertex pairs of (P', Q') except (x', y') correspond to the Nash equilibria of (A, B) by construction. □

The effect of the affine transformation on the size of the numbers in the description of P' and Q' is polynomial. Thus, if the data required to specify P' and Q' is polynomial in $m+n$ (true for all the polytopes we consider), so will be the data for the game matrices A and B.

In Section 3.9, we show that it is possible to label a pair of polytopes, with each poltopes in dimension d with $2d$ facets, such that there are no completely labelled vertex pairs. Thus, not every pair of polytopes with the correct numbers of facets can be labelled arbitrarily to produce a game.

In the next section, we introduce dual cyclic polytopes, which we use in the construction of games with exponentially long LH paths.

3.2 Dual cyclic polytopes

We construct games by defining P and Q in (2.4) as the well-understood "dual cyclic polytopes" (see Ziegler (1995) or Grünbaum (2003)), similar to von Stengel (1999). These polytopes are in dimension d and have f facets, where $d = m$ for P and $d = n$ for Q, and in both cases $f = m+n$.

A standard way of obtaining a *cyclic polytope* P' in dimension d with f vertices is to take the convex hull of f points $\mu(t_i)$ on the *moment curve* $\mu\colon t \mapsto (t, t^2, \ldots, t^d)^\top$ for $1 \le i \le f$. However, the polytopes in (2.4) 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*

$$P'' = \{\, z \in \mathbb{R}^d \mid c_i^\top z \le 1,\ 1 \le i \le f \,\}. \tag{3.2}$$

Both P and Q in (2.4) will be dual cyclic polytopes with a special order of their inequalities corresponding to the facet labels. A suitable affine transformation, described in Proposition 3.1, gives P from P'', and Q in a similar manner, so that the first m inequalities (for the pure strategies of player 1) in P have the form $x \geq \mathbf{0}$, and the last n inequalities (for the pure strategies of player 2) in Q are $y \geq \mathbf{0}$. The last n inequalities $B^\top x \leq \mathbf{1}$ in P and the first m inequalities $Ay \leq \mathbf{1}$ in Q then determine the game (A,B). The game data is of polynomial size in $m+n$, since the coefficients c_i in (3.2) are clearly polynomial, and the affine transformation giving P from P'', and simlilarly Q, preserves polynomiality. So, the running time of an algorithm with the game as input is polynomial if and only if it is polynomial in $m+n$.

A vertex u of a dual cyclic polytope in dimension d with f facets is characterized by the *bitstring* $u_1u_2\cdots u_f$ of length f, 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 $f-d$ bits are 0. Assume that $t_1 < t_2 < \cdots < t_f$ when defining the kth facet of P'' by the binding inequality $(\mu(t_k) - \overline{\mu})z \leq 1$. Then the vertices of P'' are characterized by the bitstrings fulfilling the *Gale evenness* condition, due to Gale (1963): A bitstring represents a vertex if and only if any substring of the form $01\cdots10$ 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*. Except for in Section 3.9, we only consider *even* dimensions 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 $G(d,f)$ be the set of these bitstrings of length f with d ones fulfilling Gale evenness.

3.3 The double cyclic polytope games $\Gamma(m,n)$

For the rest of this chapter, both m and n are even, and $m \leq n$. The vertices of P and Q are described by the sets of bitstrings $G(m,m+n)$ and $G(n,m+n)$, respectively. The 1's in a bitstring encode the facets that the vertex belongs to. We need *facet labels* for the complementarity condition and the LH algorithm. The facet labels are defined by permutations l and l' of $1,\ldots,m+n$ for P and Q, respectively. For a vertex u of P, which we identify with its bitstring in $G(m,m+n)$, its labels are given by $l(k)$ where $u_k = 1$.

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. A vertex v of Q is identified with a bitstring in $G(n, m+n)$, and its labels are $l'(k)$ where $v_k = 1$, for $1 \leq k \leq m+n$. The kth facet of Q has label $l'(k)$. The permutation l' has the fixed points $l'(1) = 1$ and $l'(m) = m$, and otherwise exchanges adjacent numbers, as follows:

$$l'(k) = \begin{cases} k, & k = 1, m, \\ k + (-1)^k, & 2 \leq k \leq m-1, \\ k - (-1)^k, & m+1 \leq k \leq m+n. \end{cases} \tag{3.3}$$

Let $\Gamma(m,n)$ be a game defined in this way.

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

$$e_0 = (1^m 0^n, 0^m 1^n) \in G(m, m+n) \times G(n, m+n). \tag{3.4}$$

The following lemma states that in any Nash equilibrium of $\Gamma(m,n)$, player 1's strategy has full support.

Lemma 3.2 *Consider a Nash equilibrium of $\Gamma(m,n)$, represented by a pair of bitstrings (u,v) in $G(m,m+n) \times G(n,m+n)$. Then $u = 0^m s$ and $v = 1^m t$ for some bitstrings s and t of length n.*

Proof: The vertex pair (u,v) is completely labelled, and it is not the artificial equilibrium e_0. Either $u_m = 1$ or $v_m = 1$. We begin with the latter case, so $u_m = 0$. If $v_{m+1} = 1$, then $u_{m+2} = 0$ (via complementarity, since $l'(m+1) = m+2$) so $u_{m+1} = 0$ by Gale evenness, and thus $v_{m+2} = 1$. Continuing in that way, all 1's to the right of the mth bit v_m of v (which is 1) have to come in pairs. Similarly, if $v_{m-1} = 1$ (implied by $v_{m+1} = 0$, $v_m = 1$ and Gale evenness), then $u_{m-2} = 0$ by complementarity, which with $u_m = 0$ implies $u_{m-1} = 0$, and $v_{m-2} = 1$. This means that the 1's to the left of v_m come in pairs if there is a zero to the left of them. But then the run of 1's containing v_m has odd length and must include v_{m+n}, and then it is too long. Hence, there is no zero in v to the left of v_m, and $v = 1^m t$ and $u = 0^m s$ for some bitstrings s and t of length n, as claimed.

In the same way, $u_m = 1$ implies that all bits in u to the left of u_m are 1, and, since u has only m bits equal to one, all bits to right of u_m are zero, so that $(u,v) = e_0$, which is the artificial equilibrium and not a Nash equilibrium. □

Corollary 3.3 *The only Nash equilibrium of the game* $\Gamma(d,d)$ *is* $e_1 = (0^d 1^d, 1^d 0^d)$.

Proof: For $m = n = d$, the vertices u and v in Lemma 3.2 are bitstrings containing d 1's, so that $s = 1^d$ and $t = 0^d$. □

3.4 The square games $\Gamma(d,d)$ and long Lemke–Howson paths

In this section, we only consider square games where $d = m = n$. Then, by Corollary 3.3, all LH paths, for any missing label, lead from e_0 to e_1. We analyze these paths for square games. It will then be easy to describe the LH paths for the non-square games $\Gamma(d,2d)$, which are treated in Section 3.5.

Denote by $\pi(d,k)$ the LH path with missing label k for the game $\Gamma(d,d)$. We regard $\pi(d,k)$ 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 $G(d) \times G(d)$, where $G(d)$ abbreviates $G(d,2d)$. Let $L(d,k) = L$ be the length of that path.

	P								Q								
step	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**1**	**3**	**2**	**4**	**6**	**5**	**8**	**7**	step
1 ⟨	1̲	1	1	1	·	·	·	·									
	·	1	1	1	1	·	·	·	·	·	·	·	1	1̲	1	1	⟩ 2
3 ⟨	·	1̲	1	·	1	1	·	·									
5 ⟨	·	·	1̲	1	1	1	·	·	·	·	1	1̲	·	·	1	1	⟩ 4
7 ⟨	·	·	·	1̲	1	1	1	·	·	1	1	·	·	·	1	1̲	⟩ 6
9 ⟨	·	·	·	1	1̲	1	1	·	·	1̲	1	·	·	1	1	·	⟩ 8
11 ⟨	·	·	1̲	1̲	·	1	1	·	·	·	1̲	1̲	·	1	1	·	⟩ 10
13 ⟨	·	1	1	·	·	1̲	1	·	·	·	·	1	1̲	1	1̲	·	⟩ 12
15 ⟨	·	1̲	1	·	·	·	1	1	·	·	1	1̲	1	1	·	·	⟩ 14
17 ⟨	·	·	1̲	1	·	·	1	1	·	1	1	·	1	1̲	·	·	⟩ 16
19 ⟨	·	·	·	1̲	1	·	1	1	·	1	1	1	1̲	·	·	·	⟩ 18
	·	·	·	·	1	1	1	1	1	1	1	1	·	·	·	·	⟩ 20

Figure 3.3 The path $\pi(4,1)$, with vertices of P and Q as bitstrings. A dot represents a zero bit.

As an example, Figure 3.3 shows $\pi(4,1)$. The numbers at the top are the labels $l(k)$ and $l'(k)$ for $k = 1,\ldots,8$. The vertices of P and Q are shown as bit patterns, where for better visual distinction of the bits a zero bit is written as a dot. The 20 steps of this path are indicated at the side, where the odd-numbered steps change the vertex in P, and the even-numbered steps the vertex in Q. Step i changes the vertex pair (u^{i-1}, v^{i-1}) to (u^i, v^i). The starting point e_0 is the vertex pair $e_0 = (u^0, v^0) = (11110000, 00001111)$. Step 1 is to drop label 1 in P from u^0, so the bit u^0_1 changes from 1 to 0. By Gale evenness, this gives the bitstring 01111000 as the new vertex u^1 in P. In Figure 3.3, the bit 1 that is changed to 0 has a little downward arrow "v" underneath it, with the new bit that changes from 0 to 1 indicated with that arrow above the new bit 1 in the next vertex. In u^1, label 5 has been picked up, which is now duplicate and dropped from vertex v^1 in Q (where $v^1 = v^0$), giving the next vertex $v^2 = 00011011$ in step 2. In P, the vertex u^2 is unchanged, $u^2 = u^1$. The new duplicate label is 4. Hence, in step 3, label 4 is dropped in P, giving vertex $u^3 = 01101100$. In that manner, the path proceeds until it ends at $(u^{20}, v^{20}) = e_1$.

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. Figure 3.3 illustrates the symmetry of $\pi(d,1)$, which is stated in the next lemma, for $d = 4$.

Lemma 3.4 *Let $L = L(d,1)$ and let (u^i, v^i) be the ith 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 (3.3), 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. □

The symmetry of the path $\pi(d,2d)$ is less easy to state. Figure 3.4 shows this path for $d = 6$. In that picture, disregard the last vertex in P, and the first and last vertex in Q. Then the column labelled 12 in both P and Q has only zeros since 12 is the missing label. When this column in both P and Q is also disregarded, the bit pattern of the path shows a symmetry in each polytope by "pointwise reflection", where the point of reflection is in column 6 in step 18 of P, and at the vertex that stays fixed in Q during that step. The

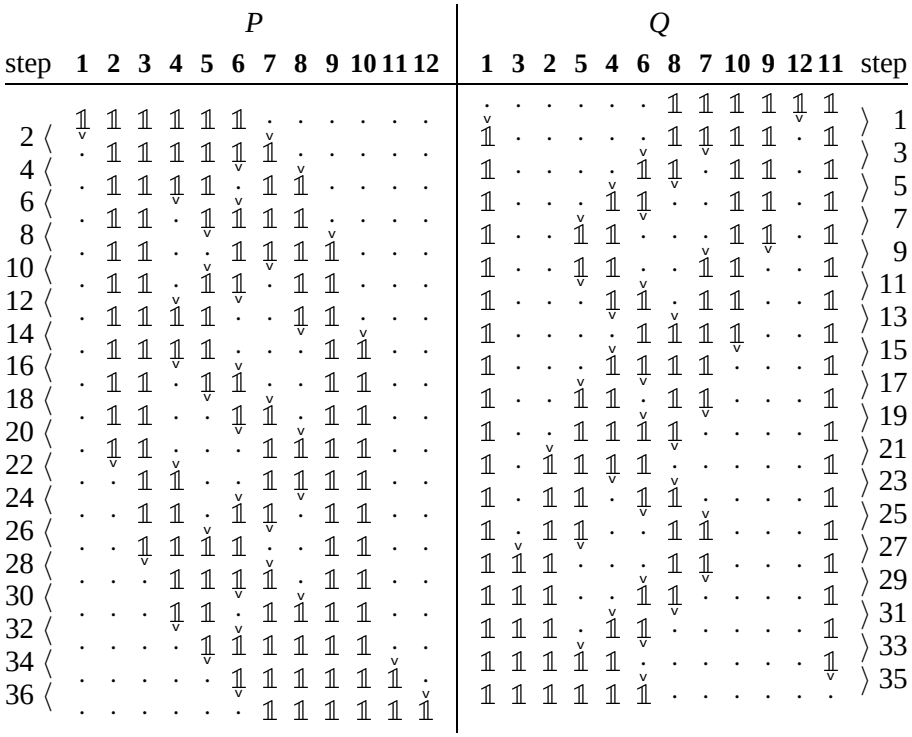

Figure 3.4 The path $\pi(6,12)$.

pointwise symmetry means that in each polytope, writing each bitstring backwards, while ignoring the bit corresponding to the missing label, gives the path in reverse direction (disregarding the first and the last two vertex pairs). For general d, this symmetry is stated in the next lemma.

Lemma 3.5 *Let $L = L(d,2d)$ and let (u^i, v^i) be the ith vertex pair of the path $\pi(d,2d)$ for $0 \le i \le L$. Let $B(d)$ be the sub-path $(u^1,v^1)\cdots(u^{L-2},v^{L-2})$ of $\pi(d,2d)$. Then for the vertex pairs of $B(d)$, for $1 \le i \le L-2$,*

$$u^i_k = u^{L-1-i}_{2d-k} \qquad (1 \le k \le 2d-1), \tag{3.5}$$

$$v^i_1 = v^{L-1-i}_{2d} = 1, \tag{3.6}$$

$$v^i_k = v^{L-1-i}_{2d-k} \qquad (2 \le k \le 2d-2), \tag{3.7}$$

$$u^i_{2d} = v^i_{2d-1} = 0. \tag{3.8}$$

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

Before we prove Lemma 3.5 we state several auxiliary definitions and lemmas that are used in subsequent proofs.

Suppose that the bitstrings $x = x_1x_2\ldots x_{2d}$ and $y = y_1y_2\ldots y_{2d}$ are two vertices of $G(d)$. Then they are connected by an edge if and only if they differ only by a substring of the form 1^p0 in x and 01^p in y or vice versa, for some even $p \geq 2$. That is, there are two positions i and j so that

$$\{x_ix_{i+1}\ldots x_j,\ y_iy_{i+1}\ldots y_j\} = \{1^p0,\ 01^p\}.$$

(If $j < i$, this uses the cyclic symmetry of the Gale evenness bitstrings, taking $2d+1$ as 1, like in step 1 in Q in Figure 3.4.) We say that the edge *crosses* the positions $i+1,\ldots,j-1$. For example, the vertices x and y of P joined by step 3 in Figure 3.3 are 01111000 and 01101100, where $i = 4,\ j = 6$, $x_4x_5x_6 = 110$, $y_4y_5y_6 = 011$ and the remaining positions of x and y are the same. This edge crosses only position 5. Recall that edges of LH paths are edges of the product polytope $P \times Q$, joining (u,v) to (u',v) or (u,v) to (u,v'). We say that such an edge crosses a position k if this holds for the respective edge joining u to u' in P or v to v' in Q.

To emphasize the dimension d, we write

$$e_0^d = (1^d0^d, 0^d1^d), \qquad e_1^d = (0^d1^d, 1^d0^d).$$

Lemma 3.6 *No edge of* $\pi(d,1)$ *crosses position* 1 *or* $2d$.

Proof: The first edge of $\pi(d,1)$ joins e_0^d to $(01^d0^{d-1}, 0^d1^d)$ and does not cross position 1 or $2d$. The same holds for the last edge joining $(0^d1^d, 01^d0^{d-1})$ to e_1^d. In any other edge, the bit in position 1 is zero in both polytopes, so the edge cannot cross position 1, nor the cyclically adjacent position $2d$. □

The next lemma concerns the first and last vertex pair of the sub-path $B(d)$ of $\pi(d,2d)$ defined in Lemma 3.5.

Lemma 3.7 *Let* $L = L(d,2d)$ *and let* (u^i,v^i) *be the ith vertex pair of the path* $\pi(d,2d)$ *for* $0 \leq i \leq L = L(d,2d)$. *Then*

$$(u^1,v^1) = (1^d0^d, 10^{d-1}1^{d-2}01), \tag{3.9}$$

$$(u^{L-2},v^{L-2}) = (0^{d-1}1^d0, 1^{d-1}0^d1). \tag{3.10}$$

Proof: See Figure 3.4 for an illustration of the case $d = 6$. After step 1 of $\pi(d, 2d)$, the vertex pair (u^1, v^1) as in (3.9) is reached. The last vertex pair of $\pi(d, 2d)$ is $(u^L, v^L) = e_1^d$. This is reached in step $L-1$ by picking up label $2d$ in P. Hence, the previous vertex pair is $(u^{L-1}, v^{L-1}) = (0^{d-1}1^d0, 1^d0^d)$, where label d is duplicate. The vertex pair (u^{L-2}, v^{L-2}) is therefore as in (3.10). □

The next lemma states conditions similar to (3.5) and (3.7) in Lemma 3.5, except that they concern labels rather than positions of bits. For the polytope Q, these conditions therefore do not describe the point-symmetry visible in Figure 3.4.

Lemma 3.8 *Let $L = L(d, 2d)$ and let (u^i, v^i) be the ith vertex pair of the path $\pi(d, 2d)$ for $0 \le i \le L$. Let r be the permutation of $\{1, \ldots, 2d\}$ defined by $r(k) = 2d - k$ for $1 \le k \le 2d-1$ and $r(2d) = 2d$. Then for $2 \le i \le L-2$, step i of $\pi(d, 2d)$ corresponds to step $L - i$ as follows: If label k is duplicate in the vertex pair (u^{i-1}, v^{i-1}), and hence dropped in step i to get (u^i, v^i), then label $r(k)$ is duplicate in (u^{L-i}, v^{L-i}), and hence picked up when reaching that vertex pair from (u^{L-1-i}, v^{L-1-i}) in step $L-i$. Furthermore, for $1 \le i \le L-2$,*

$$u^i_k = u^{L-1-i}_{r(k)} \qquad (1 \le k \le 2d), \tag{3.11}$$

$$v^i_{l'(k)} = v^{L-1-i}_{l'(r(k))} \qquad (1 \le k \le 2d). \tag{3.12}$$

Proof: The proof will be by induction on i. The permutation r serves as a relabelling to state (and prove) the symmetry of the path. It writes the labels $1, \ldots, 2d-1$ backwards, and keeps the missing label $2d$ fixed. For a vertex u of P, labels and positions are the same. Then r, seen as a reversal of the bitstrings and cyclic shift by one position, preserves the Gale evenness condition. The induction is therefore easy for P. Equations (3.9) and (3.10) clearly imply (3.11) for $i = 1$.

For a vertex v of Q, equation (3.12) has to be read as follows: the bit of vertex v^i in position $l'(k)$, which has label k (because l' is its own inverse), is equal to the bit of vertex v^{L-1-i} in position $l'(r(k))$, which has label $r(k)$.

The set $\{2, 3, \ldots, 2d-2\}$ is mapped to itself under both r and l'. Both bijections map d to itself, and it is easy to see that

$$l'(r(k)) = r(l'(k)) \qquad (2 \le k \le 2d-2). \tag{3.13}$$

Consequently, for the more restrictive case $2 \leq k \leq 2d-2$, equation (3.12) is equivalent to

$$v^i_{l'(k)} = v^{L-1-i}_{r(l'(k))} \qquad (2 \leq k \leq 2d-2,\ 1 \leq i \leq L-2). \tag{3.14}$$

Equation (3.14) implies that r can be applied to the *positions* of the bitstring v belonging to the set $\{l'(2), \ldots, l'(2d-2)\}$, which is equal to $\{2, \ldots, 2d-2\}$. If r could be applied to all positions of v, then r would be directly "compatible" with both the labels and the adjacency of vertices in Q. However, for $k \in \{1, 2d-1, 2d\}$, equation (3.13) does not hold, which complicates our proof. We have

$$\begin{aligned} l'(1) &= 1, & l'(r(1)) &= 2d, & r(l'(1)) &= 2d-1, \\ l'(2d-1) &= 2d, & l'(r(2d-1)) &= 1, & r(l'(2d-1)) &= 2d, \\ l'(2d) &= 2d-1, & l'(r(2d)) &= 2d-1, & r(l'(2d)) &= 1, \end{aligned} \tag{3.15}$$

As Figure 3.4 shows, the positions 1, $2d-1$, and $2d-2$ in Q are constant. Indeed, we will show

$$v^i_1 = 1, \qquad v^i_{2d-1} = 0, \qquad v^i_{2d} = 1 \qquad (1 \leq i \leq L-2). \tag{3.16}$$

The left two columns of (3.15) then show (3.12) also for $k = 1, 2d-1, 2d$. Instead of (3.12), we prove by induction on i the stronger assertions (3.16) and (3.14). By (3.9) and (3.10), they are true for $i = 1$.

The length L of $\pi(d, 2d)$ is even since the path starts in Q and ends in P. Hence, if i is odd or even, so is $L-i$.

As inductive hypothesis, suppose that (3.11), (3.16), (3.14), and therefore (3.12), hold for $i-1$ instead of i. By the above considerations, this implies that if k is the duplicate label of (u^{i-1}, v^{i-1}), to be dropped in step i, then $r(k)$ is the duplicate label of (u^{L-i}, v^{L-i}), to be picked up in step $L-i$.

Suppose first that i is even. Then step i from (u^{i-1}, v^{i-1}) to (u^i, v^i) is in P, that is, the duplicate label is dropped in u^{i-1} to give the new vertex u^i, and $v^{i-1} = v^i$. Thus, (3.16) and (3.14) hold trivially for i. Because r preserves Gale evenness, the edge connecting u^{i-1} to u^i in P corresponds to the edge connecting u^{L-i} to u^{L-1-i} as described. Hence, (3.11) holds for i, which completes the induction step for even i.

Secondly, let i be odd, where step i is in Q. Let k be the duplicate label of (u^{i-1}, v^{i-1}). In step i, label k is dropped in v^{i-1} to give the new vertex v^i, and $u^{i-1} = u^i$. Hence, (3.11)

holds trivially for i. If $k = 1$, then because (3.16) holds for $i-1$ by inductive hypothesis, changing $v^{i-1}_{l'(k)}$ to zero would give $v^{i}_{l'(2d)} = 1$ and thereby terminate the path, which is not possible. Hence, $k \neq 1$. Similarly, if $k = 2d-1$, then since (3.12) holds by inductive hypothesis, label $r(k)$ is duplicate in (u^{L-i}, v^{L-i}), where $r(k) = 1$ and so label $2d$ would be picked up in (u^{L-1-i}, v^{L-1-i}) when going along the edge in Q from v^{L-i} to v^{L-1-i}. So $k \neq 2d-1$, which shows the induction step for (3.16).

The duplicate label k therefore fulfills $2 \leq k \leq 2d-2$ (since obviously $k \neq 2d$), and so does the label that is picked up, where the positions 1, $2d-1$, and $2d$ are not crossed because of (3.16). By inductive hypothesis, (3.14) holds for $i-1$, and r preserves the adjacency of positions. Hence, step i in Q corresponds to the backwards step $L-i$ in Q as claimed, which shows that equation (3.14) also holds for i. This completes the induction. □

Proof of Lemma 3.5: The preceding proof also shows Lemma 3.5. Lemma 3.8 specifies the statement to be proved by induction, and concerns the labels. We also used the fact that the permutation r preserves the adjacency of positions, as stated in the equations of Lemma 3.5. We have shown these since (3.5) is equivalent to (3.11), equation (3.6) follows from (3.16), equation (3.7) is equivalent to (3.14), and (3.8) holds trivially. Label 1 is never again picked up in P since it is present in Q from step 1 onwards, by (3.16). □

The sub-path $B(d)$ of $\pi(d, 2d)$ defined in Lemma 3.5, and the path $\pi(d, 1)$, which we call $A(d)$, are "building blocks" for other such paths in higher dimension, by inserting constant bits in suitable positions. This is stated in the following central theorem. Two paths π and π' are concatenated by the following special path composition, which we denote by $\pi + \pi'$. Hereby, both π and π' are paths on $P \times Q = G(d) \times G(d)$. Let (u, v) be the last vertex pair of π and let (u', v') be the first vertex pair of π'. Then $\pi + \pi'$ is defined if (u, v) and (u', v') are joined by an edge in $P \times Q$, that is, either $u = u'$ and v is joined to v' by an edge of Q, or $v = v'$ and u is joined to u' by an edge of P. The length of the new path $\pi + \pi'$ is the sum of the lengths of π and π' plus one; the number of its *vertex pairs* is simply the respective sum.

Theorem 3.9 *Let $A(d) = \pi(d, 1)$ and let $B(d)$ be as in Lemma 3.5. 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

$$A(d) = \beta(B(d)) + C(d), \tag{3.17}$$

$$C(d) = \alpha(A(d-2)) + \beta'(B(d)), \tag{3.18}$$

$$B(d) = \gamma(A(d-2)) + \gamma'(C(d-2)). \tag{3.19}$$

We illustrate equation (3.17) for $d = 6$ using Figure 3.5, which shows the path $A(6)$. Comparing Figure 3.4 and Figure 3.5, we see that steps 1 to 33 of $A(6)$ look almost like steps 2 to 34 of $\pi(6,12)$, which are the 33 steps of $B(6)$. The only difference is that in $B(6)$, the bit in Q in column 1 is one and the bit in the column with label 12 is zero, whereas in $A(6)$ it is the other way around. The replacement of these two bits is performed by the mapping β, defined in (3.20). The path $C(d)$ in (3.17) is simply a tail segment of $A(d)$. In Figure 3.5, $C(6)$ consists of steps 35 to 88 of $A(6)$. Step 34 of $A(6)$ is the edge in Q joining the paths $\beta(B(6))$ and $C(6)$, represented by the "+" sign in (3.17) as defined before Theorem 3.9.

To illustrate equation (3.18), note that $C(6)$, beginning with step 35 of $A(6)$, starts like $\pi(4,1)$ shown in Figure 3.3, which is $A(4)$. The bitstrings in $A(4)$ are written backwards and extended by inserting a zero bit at the front and adding the bits 110 at the end of the bitstring in each polytope. This is done by the mapping α, which is defined in (3.21) , as are the other mappings in (3.18) and (3.19).

Proof of Theorem 3.9: Overview: The mappings are given as follows: β and β' are defined on $G(d) \times G(d)$, where

$$\beta(u,v) = (u, 0v_2v_3 \ldots v_{2d-2}1v_{2d}). \tag{3.20}$$

The mapping β' applies to a final segment of the the path $A(d)$, which is symmetric as stated in Lemma 3.4. Hence, β' is determined by β, which applies to an initial segment of $A(d)$ (see (3.23) below). The other mappings are 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,

$$\alpha(u,v) = (0\overleftarrow{u}110, 0\overleftarrow{v}110). \tag{3.21}$$

With $c = 2d-4$,

$$\gamma(u_1 \ldots u_c, v) = (u_1 11 u_2 \ldots u_c 00, 10v01). \tag{3.22}$$

The mapping γ' in (3.19) applies to a final segment of $B(d)$, which, as we will show, starts after the midpoint of $B(d)$. Thus, γ' is determined by γ due to the symmetry of $B(d)$ stated in Lemma 3.5.

First we show the following equation, which is equivalent to (3.17) and (3.18):

$$A(d) = \beta(B(d)) + \alpha(A(d-2)) + \beta'(B(d)). \tag{3.23}$$

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 (3.6) and (3.8). The starting point (u^1, v^1) of $B(d)$ is given by (3.9), 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 $A(d)$. For $d = 6$, Figures 3.4 and 3.5 show how steps 2 to 34 of $\pi(6, 12)$ map to steps 1 to 33 of $A(6)$ in this way. By (3.10), the endpoint of $\beta(B(d))$ is

$$\beta(0^{d-1}1^d0, 1^{d-1}0^d1) = (0^{d-1}1^d0, 01^{d-2}0^{d-1}11). \tag{3.24}$$

The duplicate label is $2d-1$, which has been picked up in P. So in the next step of $A(d)$ (which is step 34 in Figure 3.5), label $2d-1$ is dropped in Q and label $2d-3$ is picked up, giving the vertex pair

$$(u^*, v^*) = (0^{d-1}1^d0, 01^{d-2}0^{d-2}110). \tag{3.25}$$

(For the path $\pi(d, 2d)$, label d would be picked up instead at this stage, as in step 35 in Figure 3.4.) This is the edge of $A(d)$ which joins $\beta(B(d))$ to $\alpha(A(d-2))$ in (3.23).

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 (3.21). 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 (step 35 in Figure 3.5). 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; here, we consider α also as an injective map of labels, obtained in the obvious way from (3.21), 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 (3.21) and by

Lemma 3.6; 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 (3.21). Positions $2d-1$ and $2d$ are also complementary by (3.21). For positions $2d-3$ and $2d-2$, we have complementarity because $2d-3$ is zero, since 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 (this is step 20 in Figure 3.3). So in the last step of $\alpha(A(d-2))$, label $2d-2$ is picked up in Q (this is step 54 in Figure 3.5). 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 (3.25). We have shown that the initial part of $A(d)$ in (3.23) 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 (3.23) is obtained by Lemma 3.4: The next vertex pair, obtained from (v^*,u^*) by dropping label $2d-2$ in P (step 55 in Figure 3.5), is

$$(u',v') = (01^{d-2}0^{d-1}11, 0^{d-1}1^d0), \tag{3.26}$$

which agrees with Lemma 3.4 and its symmetric counterpart in (3.24) (in Figure 3.5, step 34 backwards). Thus, the remainder is the path $\beta(B(d))$ backwards but with the bitstrings for P and Q exchanged. Using the symmetry of $B(d)$ in Lemma 3.5, 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 (3.23).

We now show (3.19). 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 3.6 and since complementarity holds for the constant positions in higher dimension, which is easily checked using (3.22). Next, by (3.23),

$$\gamma(A(d-2)) = \gamma[\beta(B(d-2)) + \alpha(A(d-4)) + \beta'(B(d-2))]. \tag{3.27}$$

Now consider the starting point (u'',v'') of $\beta'(B(d-2))$, which is (u',v') given by (3.26) but with $d-2$ instead of d. (For $d=6$, (u'',v'') is the start of step 14 of $A(4)$ in Figure 3.3.) 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

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

(For $d = 6$, these two points are, respectively, the beginning of step 15 and the end of step 21 of $\pi(6,12)$ in Figure 3.4, corresponding to steps 14 and 20 of $B(6)$.) These two vertex pairs $\gamma(u'',v'')$ and $\gamma(e_1^{d-2})$ are point-symmetric images of each other under the symmetry of $B(d)$ described in Lemma 3.5. 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 $B(d)$ (steps 15 to 21 in Figure 3.4), given by the last part of $\gamma(A(d-2))$ in (3.27), 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 (3.19), 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 the number of vertex pairs of $B(2n)$ and $C(2n)$, respectively. That is,

$$a_n = L(2n,1) + 1, \qquad b_n = L(2n,4n) - 2 \qquad (n \geq 1). \tag{3.28}$$

Then the concatenation of paths in (3.17) implies $a_n = b_n + c_n$, in (3.18) $c_n = a_{n-1} + b_n$, and in (3.19) $b_n = a_{n-1} + c_{n-1}$. 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

$$f_0 = 1, \qquad f_1 = 2, \qquad f_{n+1} = f_n + f_{n-1} \qquad (n \geq 1), \tag{3.29}$$

that is,

$$a_n = f_{3n}, \qquad b_n = f_{3n-2} \qquad (n \geq 1). \tag{3.30}$$

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, as shown next, $L(d,1) = L(d,d)$ and $L(d,d+1) = L(d,d+2) = L(d,2d-1) = L(d,2d)$. This is due to the symmetry of the

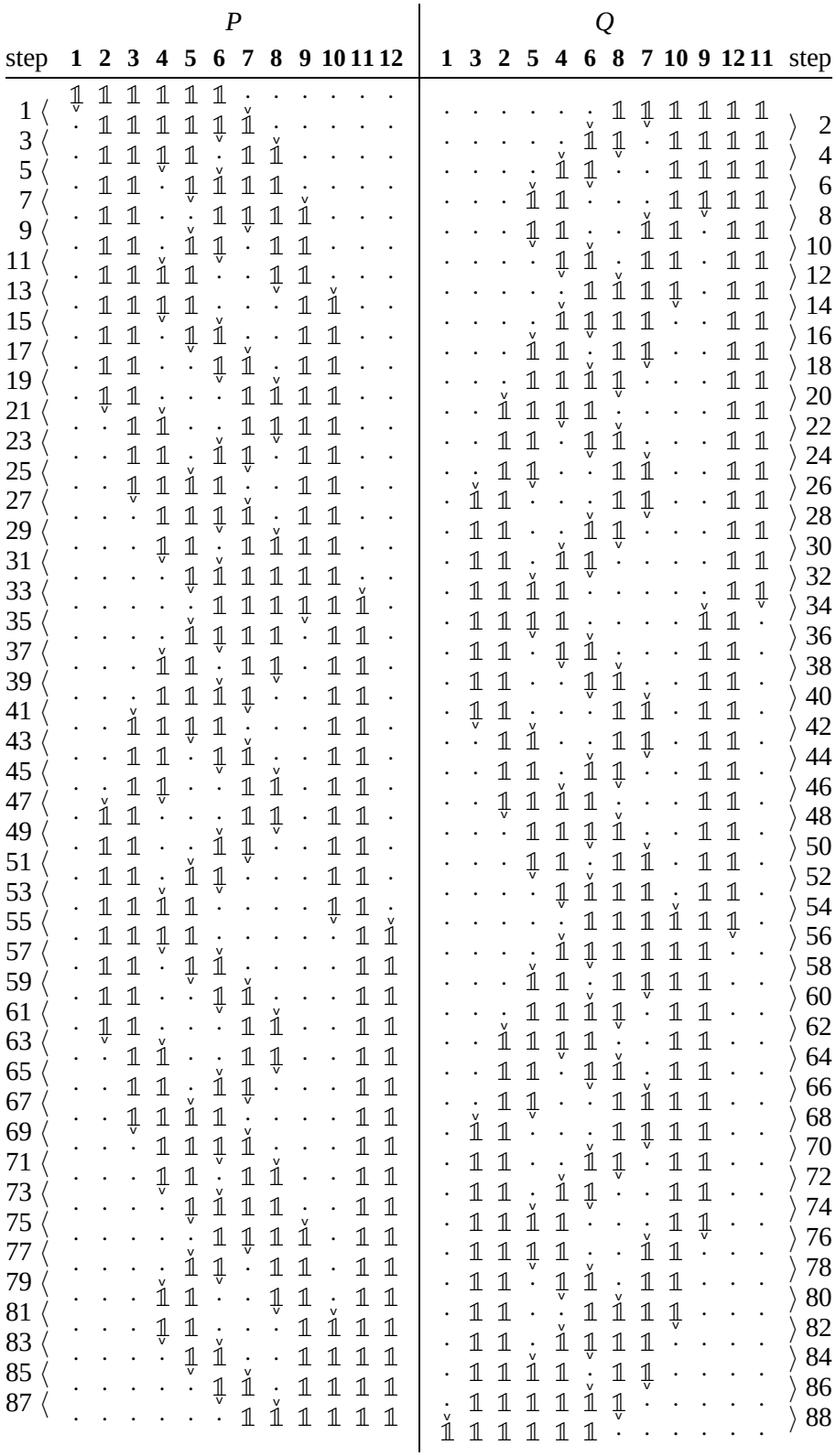

Figure 3.5 The path $A(6) = \pi(6,1)$.

Gale evenness condition and of the labellings. Other paths $\pi(d,k)$ are given as concatenations of these paths in lower dimension. They are characterized, for all possible missing labels k, in the following theorem.

Theorem 3.10 *The LH path lengths for any missing label are characterized by* (3.28), (3.29), (3.30)*, and*

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

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

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

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

The proof of this theorem is similar to the proofs of Lemma 3.5 and Theorem 3.9. Examples of paths where Theorem 3.10 applies can be found in Appendix A.2. Using (b), cases (c) and (d) cover all possible missing labels. Before we prove Theorem 3.10, we prove two lemmas.

For any paths A and B on $G(d) \times G(d)$, considered as sequences of vertex pairs, let AB denote the path A joined to the path B, where the endpoint of A is equal to the starting point of B. The length (number of edges) of AB is the sum of the lengths of A and B. The next lemma uses this concatenation of paths.

Lemma 3.11 *Let* k *be even and* $2 \le k \le d-2$*. Then*

$$\pi(d,k) = \alpha(A(k))\,\beta(A(d-k)) \tag{3.31}$$

with $\alpha : G(k) \times G(k) \to G(d) \times G(d) = P \times Q$*,*

$$\alpha(u,v) = (u_k \cdots u_1 1^{d-k} 0^{d-k} u_{2k} \cdots u_{k+1}, v_k \cdots v_1 0^{d-k} 1^{d-k} v_{2k} \cdots v_{k+1}), \tag{3.32}$$

and $\beta : G(d-k) \times G(d-k) \to G(d) \times G(d)$*,*

$$\beta(u,v) = (0^k u 1^k, 1^k v 0^k). \tag{3.33}$$

Furthermore, no edge of the path $\pi(d,k)$ *crosses position* $2d-k$ *or* $2d-k+1$*.*

Proof: The starting point of $\pi(d,k)$ is e_0^d. As required, $\alpha(e_0^k) = e_0^d$. In the first step of $\pi(d,k)$, label k is dropped in P. Position 1 of the lower dimensional polytopes, given by the bits u_1 and v_1 in (3.32), is mapped by α to position k in both P and Q in the higher dimension. In P, position k has label k, which is missing in $\pi(d,k)$. This missing label in the higher dimensional polytope P corresponds to the missing label 1 in the lower dimension. The last position $2k$ in the lower dimension, with bits u_{2k} and v_{2k}, is mapped to position $2d-k+1$ in the higher dimension. By Lemma 3.6, no edge of the path $A(k)$ crosses the positions 1 and $2k$ in the lower dimension, so inserting the substrings $1^{d-k}0^{d-k}$ or $0^{d-k}1^{d-k}$ between these bits, as done in (3.32), gives edges in P and Q, respectively. Furthermore, no edge of $\alpha(A(k))$ crosses position $2d-k$ or position $2d-k+1$.

The mapping α preserves the cyclic adjacency of labels. The first $L(k,1)$ steps of $\pi(d,k)$ are given by $\alpha(A(k))$ if the positions $k+1$ up to $2d-k$ of the higher dimensional polytopes are complementary. Complementarity of positions $k+2,\ldots,2d-k$ is immediate. Positions k and $k+1$ in P and Q, respectively, correspond to the missing label k of $\pi(d,k)$ and are thus both zero throughout. Finally, the bit in position $k+1$ in P, with label $k+1$, is 1 according to (3.32). That bit is complementary to the bit in position k in Q, which has label $k+1$, since this bit corresponds to the missing label in the lower dimensional polytope and is therefore zero throughout.

After $L(k,1)$ many steps, the vertex pair $\alpha(e_1^k)$ is reached, where

$$\alpha(e_1^k) = (0^k 1^{d-k} 0^{d-k} 1^k, 1^k 0^{d-k} 1^{d-k} 0^k) = \beta(e_0^{d-k}).$$

This is also the starting point of $\beta(A(d-k))$, as required. In a similar way as before, one can see that this is the second part of $\pi(d,k)$, which ends in $\beta(e_1^{d-k}) = e_1^d$. Lemma 3.6 for the lower-dimensional polytope and equation (3.33) imply that no edge of $\beta(A(d-k))$ crosses position $2d-k$ or $2d-k+1$. □

Lemma 3.12 *Let k be even and $2 \le k \le d$. Then*

$$\pi(d,d+k) = e_0^d + \gamma(B(k))\ \delta(B(d-k+2)) + e_1^d, \tag{3.34}$$

where γ: $G(k) \times G(k) \to G(d) \times G(d) = P \times Q$,

$$\gamma(u,v) = (u_1 1^{d-k} u_2 \cdots u_{2k} 0^{d-k}, v_1 0^{d-k} v_2 \cdots v_{2k} 1^{d-k}), \tag{3.35}$$

and δ: $G(d-k+2)\times G(d-k+2)\to G(d)\times G(d)=P\times Q$,

$$\delta(u,v)=(v_{d-k+2}\cdots v_{2d-2k+2}\,0^{k-2}1^k0\,v_2\cdots v_{d-k+1},\\ u_{d-k+2}\cdots u_{2d-2k+2}\,1^{k-2}u_{2d-2k+3}0^{k-1}u_1\cdots u_{d-k+1}).$$

No edge of $\pi(d,d+k)$ *crosses position* $d+k$ *or* $d+k+1$ *in* P.

Proof: According to (3.34), $\pi(d,d+k)$ is the concatenation of two paths in dimensions k and $d-k+2$. These do not sum to d, unlike in (3.31). This works because in the vertex pair $\delta(u,v)$, the vertex in P is obtained from v by ignoring the bits v_1, $v_{2d-2k+3}$, which are constant throughout $B(d-k+2)$ by Lemma 3.5, and adding $2k-1$ constant bits, and the vertex in Q is obtained from u by ignoring the bit $u_{2d-2k+4}$ and adding $2k-3$ constant bits.

It can be verified that both γ and δ preserve the adjacency of the labels used by the LH path, and complementarity. The path $\pi(d,d+k)$ starts as follows: In step 1, label $d+k$ is dropped from e_0^d in Q. The new vertex pair is $(1^d0^d,10^{d-1}1^{k-2}01^{d-k+1})$, which is equal to $\gamma(u^1,v^1)$ for the first vertex pair (u^1,v^1) of $B(k)$, as in (3.9) (with k instead of n). Then the path continues as described in (3.35) because, by Lemma 3.5, it first drops label 1 in P, and never picks it up again, and because the bits v_1 and v_{2k} in (3.35) stay constant according to (3.6). The last vertex pair of $\gamma(B(k))$ is, by (3.10) and (3.35), equal to

$$\gamma(0^{k-1}1^k0,1^{k-1}0^k1)=(01^{d-k}0^{k-2}1^k0^{d-k+1},10^{d-k}1^{k-2}0^k1^{d-k+1}).$$

This is equal to $\delta(1^{d-k+2}0^{d-k+2},10^{d-k+1}1^{d-k}01)$, which is δ applied to the first vertex pair of $B(d-k+2)$, using (3.9) with $d-k+2$ instead of d. The duplicate label is $d+k-1$, and is to be dropped in Q. The corresponding bit is in position $d-k$ in Q, and is the image of the bit u_1 under δ. As stated at the end of Lemma 3.5, this bit u_1 is indeed changed to zero in the first step of $B(d-k-2)$. The last vertex pair of $\delta(B(d-k+2))$ is

$$\delta(0^{d-k+1}1^{d-k+2}0,1^{d-k+1}0^{d-k+2}1)=(0^{d-1}1^k01^{d-k},1^d0^d)$$

with duplicate label d. This label has just been picked up in Q, and the corresponding bit in position d is the image of bit $u_{2d-2k+3}$ under δ. When label d is then dropped in P, the endpoint e_1^d is reached, which terminates the path $\pi(d,d+k)$. This completes the proof of (3.34). □

Proof of Theorem 3.10: For (a), let ψ be defined by $\psi(k) = d-k+1$ and $\psi(d+k) = 2d-k+1$ for $k = 1,\ldots,d$. This is a cyclic shift by d followed by a reversal of positions, which leaves the set $G(d)$ invariant, and maps e_0^d to itself. Furthermore, ψ commutes with the labellings l and l' of P and Q. Consequently, when the positions of the bitstrings representing the vertex pairs on the path $\pi(d,k)$ are permuted by ψ, one obtains the path $\pi(d,d-k+1)$, which has therefore the same length as $\pi(d,k)$.

To show (b), let $2 \le k \le d-2$. As in Lemma 3.4, the relabelling l' in (3.3) applied to both P and Q shows that $\pi(d,k)$ corresponds to the path $\pi(d,k+1)$ backwards, so these paths have the same length.

Claim (c) follows from (3.31) in Lemma 3.11.

According to (3.34) in Lemma 3.12, the length of $\pi(d,d+k)$ is the sum of the lengths of $B(k)$ and of $B(d-k+2)$ plus two (for the first edge from e_0^d and last edge to e_1^d), which shows (d). □

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 for missing label $d/2$, and is given by $L(d,d/2) = 2a_{d/4} - 2$ according to Theorem 3.10(c). If $d/2$ is odd, then the shortest path length occurs for missing label $3d/2$, where $L(d,3d/2) = L(d,3d/2+1) = 2b_{d/2+1}/2$ by Theorem 3.10(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 $L(d,3d/2) = b_{d42} + b_{d/4+1} = b_{d/4} + a_{d/4} + c_{d/4} = 2a_{d/4}$. Therefore, the shortest path results essentially when dropping label $3d/2$.

The Fibonacci numbers (3.29) have the well-known explicit expression (see, for example, Graham, Knuth, and Patashnik (1994))

$$f_n = K\phi^n + \overline{K}\,\overline{\phi}^n, \qquad \phi, \overline{\phi} = 0.5 \pm 0.5\sqrt{5}, \qquad K, \overline{K} = 0.5 \pm 0.3\sqrt{5},$$

where $\phi = 1.618\ldots$ is the Golden Ratio and $K = 1.170\ldots$. Then f_n is $K\phi^n$ rounded to the nearest integer since $\overline{K}\,\overline{\phi}^n$ is less than 0.5 and at any rate exponentially small. By Theorem 3.10(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 (multiplied by two) with every third number omitted. These shortest lengths grow with the square root of the longest lengths, which is still exponential.

Corollary 3.13 *There are $d \times d$ games, for even d, where the length of each LH path is at least proportional to $\phi^{3d/4}$.*

A construction using a similar labelling to (3.3) is possible for odd d, but there the path lengths are less symmetric than those in Theorem 3.10 for even d. We do not need this since it is trivial to obtain an odd-dimensional game from the next lower even dimension by adding a strictly dominated strategy for each player.

Not all almost complementary vertex pairs are part of the LH paths we have analysed so far. The almost complementary edges can form cycles, rather than being part of an LH path that leads to an equilibrium. We end the section with Figure 3.6, an example of an explicit cycle for dimension 6 and missing label 1.

step	1	2	3	4	5	6	7	8	9	10	11	12		1	3	2	5	4	6	8	7	10	9	12	11	step
	·	·	·	1	1	1	1	1	1	·	·	·		·	1	1	·	·	·	·	·	1	1	1	1		
1 ⟨	·	·	1	1	1	1	1	1	·	·	·	·		·	·	1	1	·	·	·	·	1	1	1	1	⟩ 2
3 ⟨	·	·	1	1	·	1	1	1	1	·	·	·		·	·	1	1	·	·	·	·	1	1	1	1	⟩ 4
5 ⟨	·	·	1	1	1	1	·	1	1	·	·	·		·	·	1	1	·	·	·	1	1	·	1	1	⟩ 6
7 ⟨	·	·	1	1	1	1	·	1	1	·	·	·		·	1	1	·	·	·	·	1	1	·	1	1	⟩ 8
	·	·	·	1	1	1	1	1	1	·	·	·		·	1	1	·	·	·	·	·	1	1	1	1	

Figure 3.6 A cycle of almost complementary edges in dimension 6 with missing label 1.

3.5 Non-square games $\Gamma(d, 2d)$ and support enumeration

So far, we have analyzed the $d \times d$ games $\Gamma(d,d)$. They have a unique equilibrium which is found by the LH algorithm after an exponential number of steps, for any missing label. However, the equilibrium is completely mixed, and is easily found by *support enumeration* (e.g., Dickhaut and Kaplan (1991), Porter, Nudelman, and Shoham (2004), or Bárány, Vempala, and Vetta (2005)). This simple algorithm tests the possible supports of equal size for both players, and checks if equating the expected payoffs to the other player in his support defines mixed strategies that are best responses to each other. There is only one pair of supports where both players use d strategies, so this is tested quickly.

In this section, we consider the $d \times 2d$ games $\Gamma(d,2d)$. By Lemma 3.2, in any Nash equilibrium of such a game, both players use mixed strategies with support size d. The following lemma states that for player 2, the supports of equilibrium strategies form only

an exponentially small fraction of the possible $\binom{2d}{d}$ supports of size d. The notation $S(d/2)$ is chosen to be consistent with von Stengel (1999).

Lemma 3.14 *Let $S(d/2)$ be the set of bitstrings of length $2d$ containing d ones of the form $s_1 s_2 \dots s_k$ where each substring s_i is either 00, 11, or 0110. Then (u,v) is a Nash equilibrium of the game $\Gamma(d,2d)$ if and only if $u = 0^d s$ and $v = 1^d t$ for $s,t \in S(d/2)$, where $s = s_1 s_2 \dots s_k$ and $t = t_1 t_2 \dots t_k$, and t_i is 11, 00, or 0110 if and only if s_i is 00, 11, or 0110, respectively, for $1 \le i \le k$. Asymptotically,*

$$|S(d/2)| \approx 0.81 \frac{2.414^d}{\sqrt{d}}, \qquad \binom{2d}{d} \approx 0.56 \frac{4^d}{\sqrt{d}}. \tag{3.36}$$

Proof: It is easy to see that the described bitstrings define Nash equilibria. The claims follow from von Stengel (1999): As in Proposition 3.2 of that paper, one can see that these are the only equilibria. An exact expression for the size of $S(d/2)$ is (3.6) on p. 564, and p. 566 gives an asymptotic formula, denoted by $\tilde{\sigma}(d/2)$ (for convenience, these formulas, and a table with the corresponding values for small d are given in Section A.3), with rounded parameters as in (3.36). The expression for $\binom{2d}{d}$ is based on Stirling's formula. □

The following theorem shows that the LH paths for $\Gamma(d,2d)$ are exponentially long, since they are closely related to the paths of the square game $\Gamma(d,d)$, which have length $L(d,k)$ when dropping label k.

Theorem 3.15 *Let $M(d,k)$ be the length of the LH path in the game $\Gamma(d,2d)$ when dropping label k, for $1 \le k \le 3d$. Then*

(a) *$M(d,k) = M(d,d+1-k)$ and $M(d,d+k) = M(d,3d+1-k)$, for $1 \le k \le d$;*

(b) *$M(d,k) = L(d,k)$ for even k and $2 \le k \le d$;*

(c) *$M(d,d+k) = L(d,d+k)$ for even k and $1 \le k \le d$;*

(d) *$M(d,2d+k) = L(d,1)+1$ for even k and $1 \le k \le d$.*

By condition (a) of this theorem, it suffices to consider only even labels k in conditions (b), (c), and (d), so these cover all possible missing labels k.

Proof of Theorem 3.15: Claim (a) is proved in the same way as Theorem 3.10(a), with the permutation ψ of $\{1,\ldots,3d\}$ defined by $\psi(k) = d+1-k$ for $1 \le k \le d$ and $\psi(d+k) = 3d+1-k$ for $1 \le k \le 2d$.

For (b), (c), (d), let k be even and $2 \le k \le d$. We consider the mappings $\varepsilon, \zeta, \eta : P \times Q = G(d,2d) \times G(d,2d) \to G(d,3d) \times G(2d,3d)$ defined by

$$\varepsilon(u,v) = (u_1 \cdots u_{2d-k} 0^d u_{2d-k+1} \cdots u_{2d}, v_1 \cdots v_{2d-k} 1^d v_{2d-k+1} \cdots v_{2d}), \quad (3.37)$$

$$\zeta(u,v) = (u_1 \cdots u_{d+k} 0^d u_{d+k+1} \cdots u_{2d}, v_1 \cdots v_{d+k} 1^d v_{d+k+1} \cdots v_{2d}), \quad (3.38)$$

$$\eta(u,v) = (u0^d, 1v_2 \cdots v_{2d} 1^{k-2} v_1 1^{d-k+1}). \quad (3.39)$$

Let $\rho(d,j)$ be the LH path for the game $\Gamma(d,2d)$ with missing label j, for any $j = 1,\ldots,3d$. We show that $\rho(d,k) = \varepsilon(\pi(d,k))$. Both paths start by dropping the same label k. If $k < d$, then by Lemma 3.11, no edge of the path $\pi(d,k)$ crosses position $2d-k$ or $2k-k+1$. This holds also when $k = d$, since by Theorem 3.10(a), $\pi(d,d)$ corresponds to $\pi(d,1)$, and Lemma 3.6 implies that $\pi(d,d)$ does not cross positions d and $d+1$. Therefore, it is possible to insert between positions $2d-k$ and $2k-k+1$ the bitstring 0^d in P, and 1^d in Q, as in (3.37), which implies $\rho(d,k) = \varepsilon(\pi(d,k))$, as claimed. This proves (b).

Next, we show that $\rho(d,d+k) = \zeta(\pi(d,d+k))$. By Lemma 3.12, no edge of $\pi(d,d+k)$ crosses position $d+k$ or $d+k+1$ in P, so it is possible to insert the bitstring 0^d between these positions, as done in (3.38). To obtain complementarity, 1^d is inserted between positions $d+k$ and $d+k+1$ in Q. These positions in Q are crossed by some edges of the path $\pi(d,d+k)$, but because a contiguous string of 1's is inserted, those steps of $\pi(d,d+k)$ are mapped by ζ to the respective steps of $\rho(d,d+k)$ as well. This shows the claim, giving (c).

Finally, we show $\rho(d,2d+k) = (1^d 0^{2d}, 0^d 1^{2d}) + \eta(\pi(d,1))$. Dropping label $2d+k$, the first step of $\rho(d,2d+k)$ reaches the vertex pair $(1^d 0^{2d}, 10^{d-1} 1^d 1^{k-2} 01^{d-k+1})$, which is equal to $\eta(e_0^d)$. In this vertex pair, the duplicate label is 1, which is dropped in P in the next step 2 of $\rho(d,2d+k)$. Beginning with this step, $\rho(d,2d+k)$ is equal to $\eta(\pi(d,1))$ (from step 1 onwards), for the following reasons: the vertex pairs are almost complementary; by Lemma 3.6, the bitstring 0^d can be inserted at the end of u in (3.39), and 1^{k-2} can be inserted between v_{2d} and v_1; finally, 1^{d-k+2} can be cyclically inserted

between positions v_1 and v_2, which does not affect the steps in the second polytope. This proves (d). □

In any Nash equilibrium of the game $\Gamma(d, 2d)$, the strategy of player 1 has full support. The supports of equilibrium strategies of player 2 define the set $S(d/2)$. By (3.36), these form an exponentially small fraction F,

$$F = \frac{|S(d/2)|}{\binom{2d}{d}} \approx 1.44 \times 0.6^d \tag{3.40}$$

of all supports of size d for player 2 (the support of size d for player 1 is unique). This is the success probability F of a support enumeration algorithm that tests a single random support of size d. We show in Lemma 3.16 that a support enumeration algorithm that tests d-sized supports of player 2 uniformly at random (without replacement) has to test an exponential number of supports on average before finding an equilibrium. In this lemma, the set $S(d/2)$ is called E. Let U be the set of *all* d-sized subsets of $\{1,\ldots,2d\}$, so $|U| = \binom{2d}{d}$.

Lemma 3.16 *Consider a $d \times 2d$ game where a pair of supports defines a Nash equilibrium if and only if both supports have size d, and player 2's support belongs to the set E, a set of d-sized subsets of $\{1,\ldots,2d\}$. A support enumeration algorithm that tests supports picked uniformly at random without replacement from the set of all d-sized subsets of $\{1,\ldots,2d\}$ has to test an expected number of*

$$\frac{\binom{2d}{d} - |E|}{|E|+1} + 1 \tag{3.41}$$

supports before finding an equilibrium support.

Proof: To find the expected number of guesses required to find an equilibrium we use a standard argument (Motwani and Raghavan (1995), p. 10). Assume some order of enumeration for the elements of U. The elements of $U \setminus E$, which we index by i, with $1 \le i \le \binom{2d}{d} - |E|$, correspond to non-equilibrium supports. Let W_i be the *indicator variable* that takes the value 1 if the ith element of $U \setminus E$ precedes all members of E in the enumeration of U, and 0 otherwise. Then $W = \sum_{i=1}^{|U|-|E|} W_i$ is the random variable equal to the number of supports checked before the first equilibrium is found. Then, using the linearity of expectation, we have

$$\mathbb{E}W = \mathbb{E}\Big(\sum_{i=1}^{|U|-|E|} W_i\Big) = \sum_{i=1}^{|U|-|E|} \mathbb{E}(W_i) = \sum_{i=1}^{|U|-|E|} \frac{1}{|E|+1} = \frac{\binom{2d}{d} - |E|}{|E|+1}.$$

This shows that the expected number of support guesses until an equilibrium is found is given by (3.41), as claimed. □

For $E = S(d/2)$, the number in (3.41) is about 0.7×1.66^d. Support enumeration therefore takes exponential time on average for a game $\Gamma(d, 2d)$ [1].

Corollary 3.17 *There are $d \times 2d$ games, for even d, where the length of each LH path is at least proportional to $\phi^{3d/4} \approx 1.43^d$, and where a support enumeration algorithm has to test on average about 0.7×1.66^d many supports of size d before it finds an equilibrium.*

It is natural to ask if we can construct games that are "hard to solve" even if details of the construction are known. If an algorithm knows the set E, it could simply pick a member of E. To avoid this, suppose we randomly permute the columns of the game with a uniformly chosen random permutation p. This is equivalent to permuting the positions of bitstrings in the set E with p, thus giving a new set of equilibrium supports $\bar{E}$. For the LH algorithm, the permutation does not affect the possible path lengths. How effectively does this "hide" the equilibrium supports from a support enumeration algorithm that knows E?

First, we consider a general set E (not necessarily $S(d/2)$). If a support enumeration algorithm knows the set E (but of course not $\bar{E}$), it may be possible to do better than test a random sequence of a supports. That is, a random permutation of the strategies cannot completely "hide" the structure of E. For example, if $E = \{u \in U \mid u_1 = 1\}$, so equilibrium supports are exactly those that use the first strategy, an element of $\bar{E}$ can be found in at most two guesses for anty d, by testing a bitstring x and then its complement $\bar{x}$.

What about for the games $\Gamma(d, 2d)$, where $E = S(d/2)$? In fact, in this case it is also possible to do better than testing a completely random order of supports, by exploiting the structure of E. We give an example for $d = 2$, so $S(d/2) = S(1) = \{1100, 0011, 0110\}$. Denote by p^{-1} the inverse of p. The following enumeration order is better than random: For the first guess, test any bitstring u in U. The second guess is derived from the first guess by swapping a single zero bit with a single one bit of u uniformly at random to give u' (there are four ways to do this). The third guess is derived from the first guess u by swapping a different 0/1 pair uniformaly at random to give u'' (i.e., swapping one of the

[1] When looking at *all* potential supports, rather than just those with full support for the player with d strategies, the number of guesses is obviously even larger, although Lemma 3.16 no longer applies.

remaining three 0/1 pairs). The fourth guess can be any support not tested so far, since this will be an equilibrium.

If the first guess u is not an equilibrium, then $p^{-1}(u) \in \{1010, 0101, 1001\}$. This occurs with probability 1/2. If $p^{-1}(u) \in \{1010, 0101\}$, then u' will be an equilibrium for three out of the four possible swaps of 0/1 pairs (only if $p^{-1}(u') = 1001$, is u' not an equilibrium). If $p^{-1}(u) = 1001$, then u' will be an equilibrium for two of the four possible swaps of 0/1 pairs (only if $p^{-1}(u') \in \{1010, 0101\}$, is u' not an equilibrium). Thus, using Baye's rule, if the first and second guess are unsuccesful, with probability 1/2 we have $p^{-1}(u') = 1001$ (and the third guess is defnitely successful), and with probaility 1/2 we have $p^{-1}(u') \in \{1010, 0101\}$ (and the third guess is successful with probability 2/3). This gives an expected number of guesses of

$$
\begin{aligned}
& 1 * (1/2) + \\
& 2 * (1 - 1/2)(2/3 * 3/4 + 1/3 * 1/2) + \\
& 3 * (1 - 1/2)(1 - (2/3 * 3/4 + 1/3 * 1/2))(1/2 + 1/2 * 2/3) + \\
& 4 * (1 - 1/2)(1 - 2/3)(1 - (1/2 + 1/2 * 2/3)) \\
= & 1 * 1/2 + 2 * 2/3 + 3 * 5/36 + 4 * 1/36 \\
= & 1\frac{25}{36} = 1.69444...
\end{aligned}
$$

Thus, this support enumeration scheme beats a completely random one, which gives a expected number of guesses of 1.75. It is open whether it is possible to devise a sequence of supports using $E = S(d/2)$ that give a polynomial number of expected guesses before finding an element of $\bar{E}$; it seems unlikely that this is possible. In any case, this question is somewhat artificial, since there is no reason to suppose that one would know the set E, but not the whole construction. Knowing the precise construction of $\Gamma(d, 2d)$, with the complete structure of the polytopes, except for the random permutation of the columns, it is possible to find an equilibrium in d very special pivoting steps; we describe this method in Section 3.6. However, these steps are not in any way suitable for finding an equilibrium of a general bimatrix game.

The games $\Gamma(d, 2d)$ with randomly permuted columns seem to be "hard to solve" for any known general-purpose algorithm that finds a Nash equilibrium of a bimatrix game. Enumerating the vertices of the polytopes P or Q in (2.4) is another way of finding an equilibrium (see von Stengel (2002) for a survey). A natural starting point for vertex enu-

meration is the vertex pair $(\mathbf{0},\mathbf{0})$ of $P \times Q$. In both polytopes, any Nash equilibrium vertex is at least d edges away from that starting point by Lemma 3.2, and there are an exponential number of such vertices. Therefore, one should expect that finding an equilibrium by vertex enumeration takes long as well. Since there are many vertex enumeration methods, an analysis is not attempted here. So far, the games $\Gamma(d,2d)$ seem hard for any *general* algorithm that finds an equilibrium of a bimatrix game.

3.6 Quickly finding an equilibrium of $\Gamma(d,2d)$ with permuted columns

In this section, we describe a specialized method that finds one Nash equilibrium of a game $\Gamma(d,2d)$ with randomly permuted columns in d pivoting steps; it is not a general method, and only works for these games.

For a game $\Gamma(d,2d)$ (without randomly permuted columns) the equilibrium supports of player 2 are given by Lemma 3.14. In particular, following the notation of that lemma, one equilibrium is given by $s = 1^d0^d, t = 0^d1^d$, that is,

$$(u,v) = (0^d1^d0^d, 1^d0^d1^d), \tag{3.42}$$

which corresponds to player 2 playing her *first* d strategies (columns). Assume that a permutation π of $\{1,\ldots,2d\}$ is used to permute the columns of a game $\Gamma(d,2d) = (A,B)$, to give new payoff matrices $(\overline{A},\overline{B})$.

The best response polytopes of the game $(\overline{A},\overline{B})$ are

$$\overline{P} = \{x \in \mathbb{R}^d \mid x \geq \mathbf{0},\ \overline{B}^\top x \leq \mathbf{1}\}, \tag{3.43}$$

$$\overline{Q} = \{\overline{y} \in \mathbb{R}^{2d} \mid \overline{A}\overline{y} \leq \mathbf{1},\ \overline{y} \geq \mathbf{0}\}, \tag{3.44}$$

which are identical to the polytopes

$$P = \{x \in \mathbb{R}^d \mid x \geq \mathbf{0},\ B^\top x \leq \mathbf{1}\}, \tag{3.45}$$

$$Q = \{y \in \mathbb{R}^{2d} \mid Ay \leq \mathbf{1},\ y \geq \mathbf{0}\} \tag{3.46}$$

except that the order of the inequalities $B^\top x \leq \mathbf{1}$ of P and $y \geq \mathbf{0}$ of Q have been permuted by π to give $\overline{B}^\top x \leq \mathbf{1}$ and $\overline{y} \geq \mathbf{0}$. So, for $i = 1\ldots d$, label i corresponds to inequality i in $\overline{P}$.

For $i = d+1 \ldots 3d$, label i corresponds to inequality $\pi(i-d)+d$. The method finds the equilibrium of $(\overline{A},\overline{B})$ that corresponds to (3.42) using the Gale evenness condition, and by pivoting in $\overline{P}$, the best response polytope of player 1.

The origin is a vertex of $\overline{P}$ and is represented by the bitstring 1^d0^{2d}, where the first d inequalities in (3.43), which correspond to $x \geq \mathbf{0}$, are tight. We pivot away from the origin in $\overline{P}$ by dropping label 1 (the variable x_1 enters the basis, and we leave the facet corresponding to $x_1 = 0$). We reach a new vertex v_1, picking up label $\pi(1)+d$. Next, we drop label d from v_1, thereby reaching the a new vertex v_2, and picking up label $\pi(2)+d$. We perform $d-2$ more pivoting steps. For $i = 2,\ldots,d-1$, we drop label $\pi(i-1)+d$ (which was picked up in the previous step) from v_i, thereby reaching the vertex v_{i+1}, and picking up label $\pi(i+1)+d$. We can now stop, since we have found an equilibrium (we could continue pivoting in this manner to discover the complete permutation π). An equilibrium support of player 2 is given by the strategies corresponding to columns $\{\pi(i) : i = 1,\ldots d\}$. We illustrate the case $d = 4$ in the Figure 3.7.

What if the rows are permuted as well? We can still find an equilibrium quickly, now in at most $2d+1$ steps. The method first does between 2 and $d+1$ pivots in Q. These pivot steps reveal the two rows in $\overline{A}$ that the permutation has mapped from the rows 1 and d in A, although we cannot distinguish between them. The method then does d pivots in P, as above. However, since we cannot distinguish between $\pi(1)$ and $\pi(d)$, either we find the equilibrium corresponding to (3.42), as before, or we find the equilibrium corresponding to $(u,v) = (0^{2d}1^d, 1^{2d}0^d)$.

3.7 Morris's construction, the games $\Gamma_M(d)$, and imitation games

Morris (1994) considers "Lemke paths" on simple d-dimensional polytopes T with $2d$ facets. A vertex v of T is given, and the d facets incident to v have labels $1,\ldots,d$. The remaining d facets have also labels $1,\ldots,d$, so each label appears twice. A Lemke path starts at v by dropping a label k and traversing the unique edge leaving the facet with label k. The endpoint of that edge is a new facet which has either label k, which terminates the path, or a duplicate label, which is then dropped by leaving the other facet with that

labels in P	1	2	3	4	5	6	7	8	9	10	11	12
labels in $\overline{P}$	1	2	3	4	$\pi(1)+d$	$\pi(2)+d$	$\pi(3)+d$	$\pi(4)+d$	$\pi(5)+d$	$\pi(6)+d$	$\pi(7)+d$	$\pi(8)+d$
$0 \in \overline{P}$	1	1	1	1	0	0	0	0	0	0	0	0
v_1	0	1	1	1	1	0	0	0	0	0	0	0
v_2	0	1	1	0	1	1	0	0	0	0	0	0
v_3	0	1	1	0	0	1	1	0	0	0	0	0
v_4	0	1	1	0	0	0	1	1	0	0	0	0
eq. support	0	0	0	0	1	1	1	1	0	0	0	0

Figure 3.7 Permuted labels of $\Gamma(d,2d)$ with $d = 4$.

label. The path continues in that manner until another completely labelled vertex is found. Applied to the polytope S as in (2.1) with vertex $v = \mathbf{0}$, with facet labels $1,\ldots,d$ for the inequalities $z \geq \mathbf{0}$, and $1,\ldots,d$ for $Cz \leq \mathbf{1}$, we have called this the symmetric LH algorithm. Any polytope T with vertex v can be affinely mapped to S with v mapped to $\mathbf{0}$. By Lemma 2.2, the completely labelled vertices (apart from $\mathbf{0}$) then correspond to the symmetric equilibria of the symmetric bimatrix game $(C,C^\top)$. However, this interpretation is not considered in Morris (1994).

Morris constructs exponentially long Lemke paths by taking for T the dual cyclic polytope in dimension d with $2d$ facets, for both odd and even d, suitably labelled as follows. In Section 3.2, we have identified such a polytope with the set $G(d,2d)$ of Gale evenness bitstrings, which describe the vertices of T in terms of the facets they lie on. In the order of the bits in those bitstrings, the facets $1,\ldots,2d$ are given the labels

$$l(k) = k, \quad 1 \leq k \leq d, \qquad l(d+k) = \begin{cases} d, & k = 1, \\ d-k, & k \text{ even and } 2 \leq k < d, \\ d+2-k, & k \text{ odd and } 2 \leq k \leq d, \\ 1, & k \text{ even and } k = d. \end{cases} \tag{3.47}$$

For $d = 6$, for example, the 12 facets, corresponding to the positions in a bitstring in $G(6,12)$, are labelled with $1,2,3,4,5,6,6,4,5,2,3,1$. Denote the bimatrix games $(C,C^\top)$ obtained from Morris's examples by $\Gamma_M(d)$. In analogy to Corollary 3.3, it is easy to see

that then the only completely labelled vertices of T are 1^d0^d (which is the starting point v of a Lemke path, our artificial equilibrium), and 0^d1^d, which corresponds to a completely mixed symmetric equilibrium.

However, these bimatrix games $\Gamma_M(d)$ have a large number of nonsymmetric equilibria, in particular two pure strategy equilibria that are always found by the ordinary, nonsymmetric LH algorithm after two or three steps, for *any* missing label. For illustration, we consider the game $\Gamma_M(6)$. Since the bimatrix game (A,B) is $(C,C^\top)$, the two polytopes P and Q in (2.4) are

$$P = \{x \in \mathbb{R}^d \mid x \geq \mathbf{0},\ Cx \leq \mathbf{1}\}, \qquad Q = \{y \in \mathbb{R}^d \mid Cy \leq \mathbf{1},\ y \geq \mathbf{0}\}, \tag{3.48}$$

so they are the same polytopes as S in (2.1), except that the first d inequalities $x \geq \mathbf{0}$ in P are labelled with $1,2,\ldots,d$, whereas these inequalities $y \geq \mathbf{0}$ in Q are labelled $d+1,d+2,\ldots,2d$. The inequalities in $Cx \leq \mathbf{1}$ in P correspond to the pure strategies of player 2. Since P is the dual cyclic polytope, this means that the 12 facets of P, as positions in a bitstring in $G(6,12)$, have labels $1,2,3,4,5,6,12,10,11,8,9,7$, corresponding to the interpretation of the second d labels in (3.47) as strategies of player 2. In other words, the labels $12,10,11,8,9,7$ mean that the second set of d inequalities defining the dual cyclic polytope appear, respectively, as rows $6,4,5,2,3,1$ of C, just as in the symmetric game. In the same way, Q is the dual cyclic polytope, with its 12 facets labelled $7,8,9,10,11,12,6,4,5,2,3,1$. Figure 4 shows the LH path for this game with missing label 9. It terminates at the pure strategy equilibrium $(d,d+1)$ where player 1 plays his last strategy (which has label d) and player 2 plays her first strategy (which has label $d+1$). It is easily shown that in the game $\Gamma_M(d)$, every LH path terminates either at this equilibrium or its symmetric counterpart $(1,2d)$, if d is even. If d is odd, every LH path of $\Gamma_M(d)$ leads to either $(2,2d)$ or $(d,d+2)$. For any d, every LH path is only two or three steps long.

	P												Q											
step	**1**	**2**	**3**	**4**	**5**	**6**	**12**	**10**	**11**	**8**	**9**	**7**	**7**	**8**	**9**	**10**	**11**	**12**	**6**	**4**	**5**	**2**	**3**	**1**	step
2 ⟨	1	1	1	1	1	1	·	·	·	·	·	·	1	1	1	1	1	1	·	·	·	·	·	·	⟩ 1
	1	1	1	1	1	·	·	·	·	·	·	1	1	1	·	1	1	1	1	·	·	·	·	·	⟩ 3
													·	1	1	1	1	1	1	·	·	·	·	·	

Figure 3.8 An LH path for the symmetric bimatrix game $\Gamma_M(6)$.

The symmetric equilibria of a symmetric game $(C,C^\top)$ correspond to the *arbitrary* equilibria of another, closely related game. This is the *imitation game* (C,I) (introduced in McLennan and Tourky (2005)), which is a square game where the payoff matrix to player 2 is the identity matrix I.

Proposition 3.18 (McLennan and Tourky (2005)) *The mixed strategy pair (y,y) is a Nash equilibrium of the symmetric game $(C,C^\top)$ if and only if there is some strategy x so that (x,y) is a Nash equilibrium of the imitation game (C,I).*

After our construction of the games $\Gamma(d,d)$, described in Savani and von Stengel (2004; 2006), McLennan and Tourky (2005) made the ingenious observation that the LH paths for imitation games (C,I), projected to the polytope Q of player 2, give the paths of the symmetric LH algorithm.

Proposition 3.19 (McLennan and Tourky (2005)) *Let $(C,C^\top)$ be a nondegenerate $d \times d$ symmetric game $(C,C^\top)$. The steps of the symmetric LH algorithm applied to this game with missing label k, for $1 \le k \le d$, correspond exactly to the even-numbered steps of the LH path for the imitation game (C,I) with missing label k, and to the odd-numbered steps of the LH path for (C,I) with missing label $d+k$.*

Proof: For the imitation game (C,I), the polytope Q in (2.4) is equal to the polytope S in (2.1). However, the d inequalities $y \ge \mathbf{0}$ in Q have labels $d+1,\ldots,2d$ rather than $1,\ldots,d$ in S. The polytope P is the d-cube $\{x \in \mathbb{R}^d \mid x \ge \mathbf{0},\ x \le \mathbf{1}\}$. Hence, any edge of P drops some label i and picks up label $d+i$, or vice versa, for some $i \in \{1,\ldots,d\}$. Any step of the symmetric LH algorithm is an edge of S, and thereby represents an edge of Q. It is easy to see that it corresponds to an even- or odd-numbered step of the LH path for (C,I), as claimed. □

Consequently, also noted by McLennan and Tourky (2005), the games $(C,C^\top) = \Gamma_M(d)$ give rise to exponentially long LH paths for the imitation games (C,I). It follows from the results by Morris (1994) that the longest such path has length proportional to $(1+\sqrt{2})^{d/2} \approx 1.55^d$, and the shortest path length proportional to $(1+\sqrt{2})^{d/4} \approx 1.25^d$. For our square games $\Gamma(d,d)$, these numbers are $\phi^{3d/2} \approx 2.06^d$ and, by Corollary 3.13, $\phi^{3d/4} \approx 1.43^d$, respectively. (For imitation games, the polytope P is the d-cube, so no LH

path can have more than 2^d steps.) Thus, the games $\Gamma(d,d)$ have longer LH paths. More significantly, however, they can be extended to the non-square games $\Gamma(d,2d)$ described in the previous section, which are also hard to solve by support enumeration. In contrast, the imitation games (C,I) for $\Gamma_M(d)$, which are necessarily square, are easy to solve by support enumeration. However, based on the ideas of Morris's construction and imitation games, in the next section we construct triple imitation games that are hard for both the LH algorithm and support enumeration.

3.8 The triple imitation games $\mathbf{T}(d)$

The second class of hard-to-solve bimatrix games we construct are $3d \times d$ games, where only Q is a dual cyclic polytope, this time in dimension d with $4d$ facets. Like for the double cyclic polytope games, the facets of Q have labels given by a suitable *permutation* of the order of the facets used in a combinatorial description of the vertices of such a polytope, the Gale evenness condition. We call these games *triple imitation* games since the payoff matrix of player 2 has the form of three identity matrices stacked on top of each other. (If it was a square game with a single identity matrix, it would be an imitation game (see Section 3.7 and McLennan and Tourky (2005)). The resulting polytope P is a product of d tetrahedra. As a product of simplices, it is also called a simplotope, a generalization of a cube.

As for the double cyclic polytope games, all equilibria of the triple imitation games are such that the player with the smaller number d of pure strategies uses all of them with positive probability, so the equilibria have full support for that player. In an equilibrium, both players have equal sized support, so only a d-sized subset of the $3d$ strategies of the other player can be that player's equilibrium support. The games have an exponential number of equilibria, but they form an exponentially small fraction of all possible supports, even when restricted to the supports of size d for each player. Consequently, a random guess of such a support takes exponential expected time. We characterize in Section 3.8 below the sets of supports that define an equilibrium. By permuting player 1's $3d$ strategies randomly, any sequence of support guesses is no better than random.

(Lemma 3.16 could have been stated more generally; it also applies to this case of hiding d-sized supports among $3d$ strategies, with an obvious change to the proof.)

A $3d \times d$ *triple imitation game* $\mathrm{T}(d)$ is defined as follows. The polytope Q is a dual cyclic polytope with $4d$ facets in dimension d. The polytope P is given by

$$P = \{\, x \in \mathbb{R}^{3d} \mid x \geq \mathbf{0},\ \ x_i + x_{d+i} + x_{2d+i} \leq 1 \text{ for } 1 \leq i \leq d \,\}. \tag{3.49}$$

According to (2.4), this means that the payoff matrix B used in the inequalities $B^\top x \leq \mathbf{1}$ is given by $B^\top = [I_d\, I_d\, I_d]$ with I_d as the $d \times d$ identity matrix.

The $4d$ inequalities of P are labelled as in (3.49). The inequalities of Q are those of P'' in (3.2) for $f = 4d$, labelled by the following labelling function l''. Again, l'' is its own inverse, so the $l''(k)$th inequality of P'' is the kth inequality of Q. The first, second, third, and fourth set of d inequalities get labels p_k, q_k, r_k, and s_k, respectively, where for $1 \leq k \leq d$,

$$\begin{aligned}
l''(k) &= p_k = k, \\
l''(d+k) &= q_k = 2d+1-k, \\
l''(2d+k) &= r_k = 2d+k, \\
l''(3d+k) &= s_k = \begin{cases} 4d & k = 1, \\ 4d+1-k-(-1)^k, & 2 \leq k \leq d-1, \\ 3d+1, & k = d. \end{cases}
\end{aligned} \tag{3.50}$$

An example for $d = 6$ of l'' is given in the proof of Lemma 3.20 below. Morris (1994) used a dual cyclic polytope in dimension d with $2d$ facets, where for $1 \leq k \leq d$, facets k and $s_k - 2d$ both have label k. In our construction, the role of facet k can here be taken by p_k or r_k. In this section, we show that each LH path in a triple imitation game corresponds to one of the "Lemke paths" of Morris (1994).

Similar to Lemma 3.2, the next lemma states that all Nash equilibria in $\mathrm{T}(d)$ have full support for the player with fewer pure strategies. These games have also an exponential number of equilibria.

Lemma 3.20 *The $3d \times d$ triple imitation game $T(d)$ has $3^{d/2}$ Nash equilibria, and in each equilibrium player 2 uses all d strategies with positive probability.*

Proof: According to (3.49), the polytope P is a product of d tetrahedra. Hence, in any vertex x of P, for each $k = 1, \ldots, d$, exactly three of the four inequalities $x_k \geq 0$, $x_{d+k} \geq 0$, $x_{2d+k} \geq 0$, and $x_k + x_{d+k} + x_{2d+k} \leq 1$ are binding, and the other holds as a strict inequality. Since there are four choices for each k for the non-binding inequality, P has 4^d vertices. In order to get an equilibrium (x, y), the non-binding inequality for x in P determines a facet of Q that the vertex y must lie on, that is, a 1 in the bitstring v in $G(d, 4d)$ that represents y. Because l'' in (3.50) is its own inverse, the complementary vertex pairs of $P \times Q$ are therefore given by those v in $G(d, 4d)$ where for $1 \leq k \leq d$,

$$\big(v(p_k), v(q_k), v(r_k), v(s_k)\big) \in \big\{(1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1)\big\}, \qquad (3.51)$$

that is, exactly one of the four bits $v(p_k)$, $v(q_k)$, $v(r_k)$, $v(s_k)$ is 1, and the others are 0, where $v(i)$ is the ith bit of v, for $1 \leq i \leq 4d$. To see this, suppose, for example, that $v(q_k) = 1$. Then the vertex is on the q_kth facet of Q, which has label $l''(q_k) = l''(d + d + 1 - k) = 2d + 1 - (d + 1 - k) = d + k$. So this label $d + k$ is present in Q, and by complementarity, absent in P, so the $(d + k)$th inequality for x in P is non-binding, that is, $x_{d+k} > 0$. By the combinatorial structure of P, therefore $x_k = x_{2d+k} = 0$ and $x_k + x_{d+k} + x_{2d+k} = 1$. These binding inequalities of P have labels k, $2d + k$, and $3d + k$, which are the labels of the p_kth, r_kth, and s_kth facet of Q, respectively. By complementarity, this implies $v(p_k) = v(r_k) = v(s_k) = 0$.

Suppose that $v(s_d) = 1$. We use (3.51) to show that this gives the artificial equilibrium. Namely, then $v(p_d) = v(q_d) = v(r_d) = 0$, and since position $r_d = 3d$ of the bitstring is next to $s_d = 3d + 1$, this requires that $v(3d + 2) = v(s_{d-2}) = 1$ by Gale evenness. By (3.51), $v(p_{d-2}) = v(q_{d-2}) = v(r_{d-2}) = 0$. This is shown for $d = 6$ in Figure 3.9.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$l''(i)$	1	2	3	4	5	6	12	11	10	9	8	7	13	14	15	16	17	18	24	22	23	20	21	19
$(*)$	1	2	3	4	5	6	6	5	4	3	2	1	1	2	3	4	5	6	6	4	5	2	3	1
$v(i)$				0		0	0		0							0		0	1	1				

Figure 3.9 Labels of Q for $\mathrm{T}(d)$ with $d = 6$

Here, the entries of row $(*)$ show k where $l''(i) \in \{p_k, q_k, r_k, s_k\}$. Because of the "gap" between each of $v(p_d)$ and $v(p_{d-2})$ etc., Gale evenness implies $v(p_{d-1}) = v(q_{d-1}) =$

$v(r_{d-1}) = 0$. Thus, $v(s_{d-1}) = v(3d+3) = 1$ by (3.51). Gale evenness requires $v(3d+4) = 1$, which leads to the same argument as before. Repeating this shows that $v(s_k) = 1$ for all $k = 1, \ldots, d$.

All other complementary vertex pairs give Nash equilibria of the game. If $v(s_d) = 0$, then exactly one of $v(p_d)$, $v(q_d)$, or $v(r_d)$ is 1 by (3.51). If, say, $v(q_d) = v(d+1) = 1$, then $v(p_d) = v(d) = 0$, so that by Gale evenness, $v(q_{d-1}) = 1$. This holds analogously for r_d or p_d instead of q_d. Hence, (3.51) implies, in particular, that $v(s_{d-1}) = 0$. In the same way as for the artificial equilibrium, this implies $v(s_k) = 0$ for all $k = 1, \ldots, d$, which shows the full support of the equilibrium. The 1's in v come in pairs, where $v(i) = v(i+1) = 1$ for odd i, for any of the three choices p_i, q_{2d+1-i}, or r_{i-2d} for i, subject to (3.51). This gives $3^{d/2}$ Nash equilibria. □

For $\mathrm{T}(d)$, $|U(d)| = \binom{3d}{d} = \Theta((27/4)^d/\sqrt{d})$, using Stirling's formula. Hence, asymptotically $|U(d)| - |N(d)|$ is $|U(d)|$, and $E(d)$ in (3.41) is exponential for the games $\mathrm{T}(d)$, as it is for the games $\Gamma(d, 2d)$.

Given the description of the equilibria of $\mathrm{T}(d)$ in Lemma 3.20, it is not hard to see that the LH paths for these games correspond to one of the "Lemke paths" of Morris (1994). We only give a short summary here. Essentially, a dropped label corresponding to one of the $3d$ pure strategies of player 1 changes to another vertex of the simplotope P. This corresponds to one edge traversal in the kth tetrahedron, for some k where $1 \leq k \leq d$. Correspondingly, the respective bit $v(s_k)$ of the vertex v in the artificial equilibrium is changed from 1 to 0. (This happens directly when the initially dropped label corresponds to a strategy of player 2.) The steps in Q, which happen every other time, then mimic the steps in Morris's path, except that the cyclic symmetry of the bitstrings in $G(d, 4d)$ that represent Q is used so that throughout the path, either only bits (in positions) p_k are affected, or only bits r_k, in addition to the bits s_k; the bits q_k stay at 0. This mimicking is analogous to the use of (3.51) above. As a result, the bit pattern in $G(d, 4d)$ in the equilibrium that is found is either $1^d 0^{3d}$ or $0^{2d} 1^d 0^d$.

The length of the shortest path is $\Theta((1+\sqrt{2})^{d/4}/\sqrt{d})$, given as $1.24...^d$ in (**??**).

Figure 3.10 summarizes the time complexity for the games $\Gamma(d, d)$, $\Gamma(d, 2d)$, and $\mathrm{T}(d)$ for both support enumeration and the best-case behaviour of the LH algorithm.

time complexity for challenge instances (constants omitted)	expected number of guesses among all supports	expected number of guesses among full supports	length of shortest LH path
$\Gamma(d,d)$	2^d	1	$1.43...^d$
$\Gamma(d,2d)$	$2.79...^d/\sqrt{d}$	$1.65...^d/\sqrt{d}$	$1.43...^d$
$\mathrm{T}(d)$	$5.47...^d/\sqrt{d}$	$3.89...^d/\sqrt{d}$	$1.24...^d$

Figure 3.10 Summary of time complexity of support enumeration and the LH algorithm for the games $\Gamma(d,d)$, $\Gamma(d,2d)$, and $\mathrm{T}(d)$.

3.9 A pair of labelled dual cyclic polytopes with no completely labelled vertex pair

Let d be odd. Consider two identical dual cyclic polytopes R and S in dimension d with $2d$ facets. The vertices of R and S are described by the sets of bitstrings $G(d,2d)$. The facet labels are defined by permutations l and l' of $1,\ldots,2d$ for R and S, respectively. For a vertex u of R, its labels are given by $l(k)$ where $u_k = 1$. The kth facet of R has label $l(k) = k$, so l is simply the identity permutation. A vertex v of S has labels $l'(k)$ where $v_k = 1$, for $1 \leq k \leq 2d$. The kth facet of S has label $l'(k)$. The permutation l' is defined as follows:

$$l'(k) = \begin{cases} k, & k = 1,2,2d-1,2d \\ k-(-1)^k, & 3 \leq k \leq 2d-2, \end{cases} \tag{3.52}$$

Remark 3.21 *No vertex of the product polytope $R \times S$ is completely labelled.*

Proof: Suppose, to the contrary, that there is a completely labelled vertex pair $(u,v) \in G(d,2d) \times G(d,2d)$. Because d is odd, for any vertex $w \in G(d,2d)$, we have $w_1 = 1$, $w_{2d} = 1$, or both, by Gale evenness. Suppose $u_1 = 1$, which is without loss of generality by symmetry. Then, by complementarity, $v_1 = 0$, so in turn $v_{2d} = 1$ and $u_{2d} = 0$. We

depict the situation for $d = 5$.

	R										S									
label	1	2	3	4	5	6	7	8	9	10	1	2	4	3	6	5	8	7	9	10
(u,v)	1									0	0									1

Suppose $u_{2d-1} = 1$. Then $u_{2d-2} = 1$, by Gale evenness. So, by complementarity, $v_{2d-3} = v_{2d-1} = 0$. Then $v_{2d-2} = 0$, by Gale evenness. By repeatedly applying complementarity and Gale evenness in this way, we see that $u_3 = u_4 = \cdots = u_{2d-2} = 1$, which is a contradiction, because u has exactly d bits that are 1. So $u_{2d-1} = 0$. Similarly, $v_2 = 0$. Thus, we have the following situation.

	R										S									
label	1	2	3	4	5	6	7	8	9	10	1	2	4	3	6	5	8	7	9	10
(u,v)	1	1							0	0	0	0							1	1

So $u_3 = 1$, by Gale evenness, since d is odd. Then, $v_4 = 0$ by complementarity, and consequently $v_3 = 0$ by Gale evenness. Thus, by complementarity, $u_4 = 0$, and consequently $u_5 = 1$ by Gale evenness. By repeatedly applying complementarity and Gale evenness in this way, we see that $u_1 = u_2 = \cdots = u_{2d-2} = 1$, which is a contradiction, because u has exactly d bits that are 1. □

Chapter 4

A Unified View of Complementary Pivoting Algorithms for Bimatrix Games

The LH algorithm finds one solution to an LCP (2.2) derived from a bimatrix game. The closely related algorithm by Lemke (1965), which we describe in Section 4.3, solves more general LCPs. In Section 4.4, we describe the LCP map and *complementary cones* view of LCPs (e.g., Megiddo (1986)). The LCP corresponding to a bimatrix game can, without loss of generality, be defined by a *positive* matrix. We show that this implies that the LCP map is surjective. Then, in Section 4.6, we explain Lemke's algorithm using this complementary cones view (e.g., Eaves and Scarf (1976)), which is useful for analyzing the *expected* running time of Lemke's algorithm for LCPs, and the corresponding self-dual parametric simplex algorithm for LPs (for more details, see Section 5.2).

In Section 4.7, we present a new unified view of the LH algorithm and Lemke's algorithm via complementary cones; so far such a view was not known. Finally, we extend Lemke's algorithm to give more freedom when starting the algorithm. Before we describe Lemke's algorithm for general LCPs, in Sections 4.1 and 4.2 we first justify restricting our attention to finding symmetric equilibria of symmetric bimatrix games with *cost* matrices, as this simplifies our exposition of complementary cones.

The new research in this chapter appears in Sections 4.5, 4.7, and 4.8.

4.1 Finding symmetric equilibria of symmetric games

In this section, we argue that we can focus on finding symmetric equilibria of symmetric bimatrix games, rather than finding Nash equilibria of general bimatrix games. The symmetrization (2.3) shows that the symmetric problem is no easier than the general one, and since symmetric equilibria of a game $(C, C^\top)$ are exactly the Nash equilibria of the imitation game (C, I) (see Section 3.7), the symmetric problem is also no harder than the general one. Thus, the problem of finding a symmetric equilibrium of a symmetric game is polynomially equivalent to finding a Nash equilibrium of a general bimatrix game.

The imitation game construction shows that a symmetric game must have at least one symmetric equilibrium. The symmetric LH algorithm described in Section 2.2 also shows this, as does a simple modification of Nash's original proof of the existence of Nash equilibria in general finite games, which also appears in his seminal paper (Nash (1951)). It is also worth noting that the standard LH algorithm finds an *odd* number of equilibria for a nondegenerate game, as equilibria are all endpoints of paths, with one endpoint being the artificial equilibrium. Then, since a symmetric game has an even number of nonsymmetric equilibria (they come in pairs, because the players can be exchanged), it must have at least one symmetric equilibrium.

4.2 Costs instead of payoffs

As stated at the start of Section 2.2, the symmetric equilibria of a symmetric bimatrix game $(C, C^\top)$ defined by *payoff* matrices are the nonzero solutions of an LCP (M, q), with $M = -C$ and $q = \mathbf{1}$. Since zero is a solution of this LCP, but is not a Nash equilibrium, it is sometimes convenient to rewrite the bimatrix game with *costs* instead of payoffs, with costs obtained from payoffs by negation. In fact, the original LH algorithm, which we discuss below, used the cost setup.

In the cost setup, as we did for payoffs, we use normalized payoffs of 1, which only works if all costs are positive, as we now describe. Given a symmetric $n \times n$ game defined by a cost matrix C for player 1, against a mixed strategy y of player 2 , the expected costs

for the rows are given by Cy. The *best response cost* against y is

$$\min_i (Cy)_i \,. \tag{4.1}$$

Denote by e_k the unit vector with kth component equal to 1 and all others 0.

Lemma 4.1 *The best response cost is always positive if and only if $C > 0$.*

Proof: Suppose $c_{ij} \leq 0$ for some i and j. Then against $y = e_j$ the best response is i with cost $c_{ij} \leq 0$. If $C > 0$ then $Cy > 0$, since $y \geq 0$ and $\mathbf{1}y = 1$. □

If $C > 0$, then, with $y \geq 0$ and $\mathbf{1}y = 1$, $\min_i (Cy)_i = v$ if and only if $(Cy)_i \geq v$ for all i, and $(Cy)_i = v$ for some i. Then, we can divide by v, which is positive by Lemma 4.1, and preserve the inequalities. This gives $(Cy')_i \geq 1$ for all i, and $(Cy')_i = 1$ for some i, with $y' = y \cdot 1/v$, and $y' \geq 0$ and $y' \neq 0$. The requirement that $C > 0$ is not restrictive, since a constant can be added to all entries without changing the equilibria.

So, rather than considering the polyhedron

$$H = \{(y, v) \in \mathbb{R}^n \times \mathbb{R} \mid y \geq \mathbf{0}, \quad \mathbf{1}y = 1, \quad Cy \geq \mathbf{1}v\},$$

we consider the polyhedron

$$W = \{z \in \mathbb{R}^n \mid z \geq \mathbf{0}, \quad Cz \geq \mathbf{1}\}, \tag{4.2}$$

which is a projective transformation of the polyhedron

$$\{(y, v) \in \mathbb{R}^n \times \mathbb{R} \mid y \geq \mathbf{0}, \quad \mathbf{1}y = 1, \quad Cy \geq \mathbf{1}v, \quad v > 0\}.$$

This projective transformation has normalised the best response costs to 1. See von Stengel (2002, p. 1735) for details of the corresponding projective transformation for payoffs. We have used such a normalization for payoffs in Section 2.1; W is the analogue of S in (2.1). The polyhedron H is not full dimensional, but having eliminated the equation $\mathbf{1}y = 1$ by the projective transformation, the polyhedron W is full dimensional. The polyhedron W is *not bounded*, unlike S, and is not a polytope; it is like the polytope S "turned inside out". Each of the orthant-facets is a facet of W, corresponding to an unplayed pure strategy. In S, the corresponding facets all intersect at the origin, which is the artificial equilibrium. With costs, there is no artificial equilibrium, which is exactly the trivial solution of the LCP we have chosen to avoid. Except for this, the facet incidences of W and

S are the same, which would not be the case if C was not positive, as some orthant-facet would be "chopped off". Figure 4.1 gives an example, where

$$C = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}, \tag{4.3}$$

so the polyhedron is $\{(z_1, z_2)^\top \in \mathbb{R}^2 \mid z_1 \geq 0, \quad z_2 \geq 0, \quad 2z_1 + z_2 \geq 1, \quad z_1 + 3z_2 \geq 1\}$, and the game has a unique, completely mixed symmetric equilibrium.

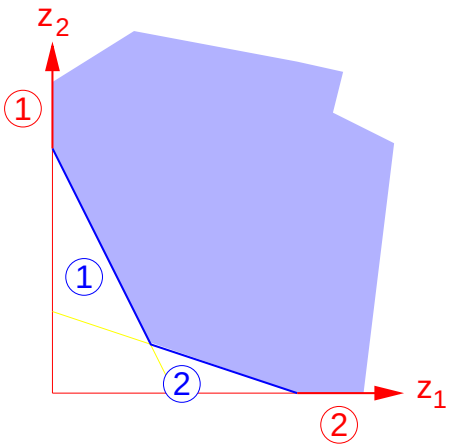

Figure 4.1 The best response polyhedron for cost matrix C

Equilibria of the symmetric game $(C, C^\top)$ defined by a positive cost matrix C are those points in W that satisfy the complementarity condition

$$z^\top (Cz - \mathbf{1}) = 0. \tag{4.4}$$

Thus the symmetric equilibria of $(C, C^\top)$ are the solutions of the LCP (M, q), with $M = C$ and $q = -\mathbf{1}$.

With the cost setup, the LH path with missing label k comes in along the extreme ray of W that is part of the coordinate axis for z_k (where $z_i = 0$ for all $i \neq k$), until it hits some facet of the polyhedron. As usual, if the label of this facet is the missing label k we are done, otherwise it is a duplicate label and we continue pivoting, according to the complementary pivot rule. The path terminates when the label k is picked up, that is, when either z_k or the slack variable $w_k = 1 - (Cz)_k$ leaves the basis.

4.3 Lemke's algorithm

In this section, we describe the complementary pivoting algorithm due to Lemke (1965), which can solve a general LCP (2.2), and not just a bimatrix game. Lemke's algorithm is closely related to the LH algorithm. In Section 4.7, we describe a unified view of these two algorithms (as applied to bimatrix games).

Our exposition in this section, except for Corollary 4.3, which is trivial, and Figure 4.2, is to a large extent taken verbatim from Koller, Megiddo, and von Stengel (1996), which in turn refers to Murty (1988, pp. 63–84) and Cottle, Pang and Stone (1992, pp. 270–280 and 336–342).

Given an n-vector q and an $n \times n$ matrix M, the LCP is to find two n-vectors z and w so that

$$\begin{aligned} z \geq 0, \qquad & w \geq 0, \\ & w = q + Mz, \\ & z^\top w = 0, \end{aligned} \tag{4.5}$$

or to determine that no such vectors exist. The system (4.5) is equivalent to the definition (2.2) of an LCP given above, with the slack vector w introduced. Lemke's algorithm either finds a solution to (4.5) or terminates in a well-defined way, called *ray termination*, without a solution. For certain classes of matrices, ray termination implies that the LCP has no solution. Theorem 4.2, proved below, is such a result for the special vector q and matrix M in our application; since that LCP has always at least one solution, a solution will be found by Lemke's algorithm.

Complementary Pivoting

Because the vectors z and w in (4.5) are nonnegative, the orthogonality condition $z^\top w = 0$ is equivalent to

$$z_i w_i = 0 \qquad \text{for } i = 1, \ldots, n. \tag{4.6}$$

Since this means that z_i or w_i has to be zero, each of these two variables is called the *complement* of the other, for $1 \leq i \leq n$.

It is useful to generalize the system (4.5) by introducing an n-vector d with positive components (for example, $d = (1,\ldots,1)^\top$), called the *covering* vector, and an auxiliary variable z_0. Let I denote the $n \times n$ identity matrix. The generalized problem is that of finding $w \geq 0$, $z_0 \geq 0$, and $z \geq 0$ so that

$$Iw - dz_0 - Mz = q \tag{4.7}$$

and (4.6) hold. A solution w, z_0, z to this problem is called *almost complementary*. It is a solution to (4.5) iff $z_0 = 0$.

In (4.7), the vector q is represented as a nonnegative linear combination of certain columns of the matrix $[I, -d, -M]$. Like the simplex and LH algorithms, Lemke's algorithm computes with *basic* feasible solutions of this system. For the moment, we assume that these are all *nondegenerate*, so that the basic variables are always positive; this can always be achieved by slightly perturbing q; we will come back to the corresponding lexicographic rules that are applied in the case of degeneracy.

Lemke's algorithm is very similar to the LH algorithm, using the same complementary pivoting rule. In the algorithm, almost complementary basic feasible solutions to (4.7) are iteratively changed by pivoting operations. An initial solution of this kind is easily found. Let $z = 0$, which implies (4.6). If z_0 is sufficiently large and $w = q + dz_0$, then w is nonnegative since $d > 0$, and (4.7) is fulfilled. The set of these almost complementary solutions is called the *primary ray*. Let z_0 be minimal such that $q + dz_0 \geq 0$ (and $z_0 \geq 0$). If $z_0 = 0$, that is, $q \geq 0$, then the LCP (4.5) is solved with $w = q$, $z = 0$. Otherwise, z_0 is positive, and at least one component w_i of the vector $w = q + dz_0$ is zero. Then the variables w_j for $j \neq i$ and z_0 are basic variables since the corresponding columns of I and the covering vector d are linearly independent. This yields the first almost complementary basic feasible solution to (4.7).

For the central step of the algorithm, consider any almost complementary basic feasible solution where z_0 is basic. The n basic variables are positive by nondegeneracy, and include at most one variable of each complementary pair by (4.6). Thus, there is precisely one such pair z_i, w_i where both variables are nonbasic and have value zero (this is the duplicate label that we know from the LH algorithm). If either variable is increased while maintaining the linear relationship (4.7), this will still be an almost complementary solution as long as the variables stay nonnegative. For the initial solution, increasing w_i

results in the primary ray. Therefore, the complement z_i of w_i is increased until some other basic variable x_j becomes zero. This is implemented as a pivot with z_i as entering and x_j as leaving variable. If the leaving variable x_j is z_0, a solution to the original LCP is at hand. Otherwise, there is again a pair z_j, w_j of complementary variables that are both nonbasic. One of them is the variable x_j that has just left the basis. Again, its *complement* is chosen as the new entering variable, and the process is repeated.

Lemke's algorithm has a geometric view. The set of all nonnegative solutions to (4.7) is a polyhedron where the basic feasible solutions are the vertices. For this system, the almost complementary solutions are certain edges of the polyhedron. The algorithm traces a unique almost complementary *path* consisting of such edges. The path starts with the primary ray, whose endpoint corresponds to the first basis. A move from vertex to vertex along an edge represents a pivoting operation, which is uniquely determined by the pair of complementary variables that are both nonbasic at the respective vertex, and by the direction on the path.

The leaving variable is decided according to the complementary pivoting rule. For the simplex algorithm for linear programming a different rule, based on improving the objective function, is used to determine the leaving variable. In both cases, and for pivoting in general, the entering variable is chosen so as to preserve feasibility, This is done using the so-called *minimum ratio test*, which we now describe briefly.

Entering and Leaving Variables

We want to pivot from one basic solution of the system of linear equations $Ax = b$, to another. Consider a basic feasible solution with basis B, and let N denote the index set of the nonbasic columns. The following equations are equivalent for any x:

$$\begin{aligned} Ax &= b, \\ A_B x_B + A_N x_N &= b, \\ x_B &= A_B^{-1} b - A_B^{-1} A_N x_N. \end{aligned} \tag{4.8}$$

Equation (4.8), called a "dictionary" by Chvátal (1983, p. 98), expresses the basic variables x_B in terms of the nonbasic variables x_N, in particular for the basic feasible solution with $x_N = 0$. Assume that all components of x_N are kept zero except x_i. We denote the

entering column $A_B^{-1}A_i$ by e, and the right hand side $A_B^{-1}b$ by r. Then, (4.8) has the form

$$x_B = r - e\,x_i\,. \tag{4.9}$$

For $x_i = 0$, (4.9) represents the current basic feasible solution $x_B = r$. How long does x_B stay nonnegative if x_i is gradually increased? If e has no positive components, then x_i can be made arbitrarily large (for the simplex algorithm this would signify an unbounded LP, for Lemke's algorithm this is ray termination). Otherwise, some components e_j of e are positive. These impose an upper bound on the choice of x_i in (4.9) so that x_B stays nonnegative, by the following minimum ratio test:

$$x_i = \min\{r_j/e_j \mid e_j > 0\}\,. \tag{4.10}$$

The components of the vector $r - e\,x_i$, like the rows of A_B^{-1}, are indexed with the elements j of B. With x_i determined by (4.10), at least one of these components is zero, and the corresponding variable x_j is made the leaving variable and becomes nonbasic. We obtain a new feasible solution where the entering variable x_i is set as in (4.10), and the remaining basic variables in x_B (except x_j) are changed according to (4.9).

The system $Ax = b$ is here (4.7), with $A = [I, -d, -M]$ and $x = (w, z_0, z)^\top$. The basic variables x_j for $j \in B$ are given by z_0 and one variable of each complementary pair z_j, w_j except one, that is, for $1 \le j \le n$, $j \ne i$. The entering variable x_i is either z_i or w_i depending on which of these two variables has just left the basis. In (4.9), there is a maximum choice of x_i such that $r - e\,x_i \ge 0$ *provided* the entering column e has positive components. It may indeed happen that $e \le 0$, and in that case x_i can be increased indefinitely; the resulting almost complementary solutions constitute what is called the *secondary ray*. In that case, Lemke's algorithm stops without a solution to the LCP (4.5), just as the simplex algorithm terminates when processing an unbounded LP. This is called *ray termination* of Lemke's algorithm. As mentioned, it can be excluded for certain q and M; we will describe a special case below.

If the entering column e in (4.9) has a positive component, then the minimum ratio test (4.10) determines the value of the entering variable x_i and a unique leaving variable because after pivoting, the new basis is nondegenerate. In that way, the algorithm generates a sequence of almost complementary basic feasible solutions that each have a unique predecessor and successor (joined by the edges of the almost complementary path). Thus,

a basis cannot be repeated. If Lemke's algorithm does not terminate with a secondary ray, it will therefore find a solution to the LCP (4.5), since eventually z_0 must leave the basis.

In summary, Lemke's algorithm consists of the following steps. We refer to the system (4.7) as $Ax = b$ as described, and identify a basis B of that system by its set of basic variables, a subset of $\{w_1, \ldots, w_n, z_0, z_1, \ldots, z_n\}$. All nonbasic variables have value zero. We determine the first almost complementary basic feasible solution using a pivot where z_0 enters and w_i leaves the basis, starting with an infeasible basic solution where $w = b$, $z_0 = 0$, and $z = \mathbf{0}$.

0. (Initialization.) Input q, d, M. Let the basic variables be $w_1, \ldots, w_n$, with $w = b = q$ and $z_0 = 0$, $z = \mathbf{0}$. If $b \geq \mathbf{0}$, then stop: this is a solution to (4.5)[1]. Otherwise, consider the index i such that $-b_i/d_i$ is maximal, where $b_i < 0$; that index is unique by nondegeneracy. Pivot with w_i leaving and z_0 (at value $-b_i/d_i$) entering the basis. The resulting basic solution is feasible and almost complementary. Choose the complement z_i of w_i as the new entering variable.

1. If the entering column e in (4.9) is nonpositive, then stop: ray termination, no solution to (4.5) has been found. Otherwise, the minimum ratio test (4.10) determines the leaving variable, which is unique by nondegeneracy.

2. Pivot. If the leaving variable has been z_0, then stop: a solution to (4.5) is found. Otherwise, choose the complement of the most recent leaving variable as the new entering variable. Go back to Step 1.

Degeneracy Resolution

The almost complementary path computed by Lemke's algorithm is unique only if the leaving variable in Step 1 is unique. If this is not the case, then the system (4.7) has degenerate basic feasible solutions, and Lemke's algorithm may cycle unless the leaving variable is chosen in a systematic way.

In Lemke's algorithm, degeneracy can be resolved by the lexicographic method. Intuitively, the vector q is slightly perturbed by replacing it by $q(\varepsilon) = q + (\varepsilon, \ldots, \varepsilon^n)^\top$, where

[1] We omit this check for solving bimatrix games: with a cost setup $b = q = (-1, \ldots, -1)^\top \leq \mathbf{0}$ and the check is irrelevant.

ε is positive but vanishingly small. The minimum ratio test (4.10) is replaced by the *lexico-minimum ratio test*, which determines the leaving variable uniquely. It preserves the invariant that all computed basic solutions are *lexico-feasible*. Equivalently, all basic variables in the perturbed system are positive; since that system is nondegenerate, the computed almost complementary path will again be unique. For details see Chvátal (1983), for example.

The invariant must be established for the first almost complementary basic feasible solution computed in Step 0, given by $w = q + dz_0$. This solution may be degenerate if there is a tie among the maximal ratios $-q_i/d_i$ for $q_i < 0$. If (4.7) is perturbed, replacing q by $q(\varepsilon)$, then these ratios are $-q_i/d_i - \varepsilon^i/q_i$, so that the maximum is attained uniquely for the *largest* index i among those ties. This rule has to be applied in Step 0 (it is not correct to break ties arbitrarily as suggested by Murty (1988, p. 80)). In that way, the first almost complementary basic solution is lexico-feasible, and using subsequently the lexico-minimum ratio test, Lemke's algorithm will terminate after a finite number of steps.

Excluding Ray Termination

We want to use Lemke's algorithm for computing an equilibrium of a bimatrix game. For that, it is important to exclude ray termination. Theorem 4.2, which excludes ray termination under certain conditions, and its application to finding a Nash equilibrium of an *extensive game* (game tree), appeared in Koller, Megiddo and von Stengel (1994; 1996). The theorem is based on Theorem 4.4.13 of Cottle, Pang and Stone (1992, p. 277); as mentioned by these authors (p. 377), the theorem is partly implicit in the works by Lemke (1965) and Cottle and Dantzig (1968), as we will indicate. We restate and prove this theorem here, for the general case where degeneracy is allowed. It is assumed that Lemke's algorithm uses a positive covering vector d and operates with lexicographic degeneracy resolution as described above. For a more general study of excluding ray termination see Eaves (1971).

Theorem 4.2 *If* (i) $z^\top Mz \geq 0$ *for all* $z \geq 0$*, and* (ii) $z \geq 0$*,* $Mz \geq 0$ *and* $z^\top Mz = 0$ *imply* $z^\top q \geq 0$*, then Lemke's algorithm computes a solution of the LCP* (4.5) *and does not terminate with a secondary ray.*

Proof: Let M and q fulfill (i) and (ii), and assume to the contrary that Lemke's algorithm terminates with a secondary ray. The endpoint of this ray shall be denoted by $x = (w, z_0, z)^\top$. This is a basic feasible solution of (4.7), where the vector x_B of basic variables includes z_0 since it would otherwise solve the LCP (4.5). We assume first that this solution is nondegenerate, that is, all components of x_B, in particular z_0, are positive.

Ray termination implies that the entering column e in equation (4.9) is nonpositive. The elements of the secondary ray result if x_i in that equation takes any nonnegative value. These elements shall be written as $x + \lambda\tilde{x}$ for $\lambda \geq 0$. The vector $\tilde{x} = (\tilde{w}, \tilde{z}_0, \tilde{z})^\top$ is nonnegative; its components are the components of $-e$, a one for its ith component (with $\lambda = x_i$), and zero otherwise. In particular, $\tilde{x} \neq 0$.

The elements $x + \lambda\tilde{x}$ of the secondary ray are solutions to (4.7), that is,

$$w + \lambda\tilde{w} = q \; + \; d(z_0 + \lambda\tilde{z}_0) \; + \; M(z + \lambda\tilde{z}) \tag{4.11}$$

for all $\lambda \geq 0$, and these solutions are almost complementary,

$$(z + \lambda\tilde{z})^\top (w + \lambda\tilde{w}) = 0\,. \tag{4.12}$$

Equation (4.11) with $\lambda = 0$ and $\lambda = 1$ implies

$$\tilde{w} = d\tilde{z}_0 + M\tilde{z}\,. \tag{4.13}$$

This implies $\tilde{z} \neq 0$: otherwise, $\tilde{z}_0 > 0$ since $\tilde{x} \neq 0$ and thus $\tilde{w} \neq 0$ in (4.13), which implies $w + \lambda\tilde{w} > 0$ because $d > 0$, so that by (4.12), $z = 0$, which means that the secondary ray is the primary ray, contradicting the fact that we have a path.

Equation (4.12) and $\tilde{x} \geq 0$ imply $\tilde{z}^\top \tilde{w} = 0$, and thus by (4.13),

$$0 = \tilde{z}^\top \tilde{w} = \tilde{z}^\top d\tilde{z}_0 + \tilde{z}^\top M\tilde{z}\,.$$

This equation has been stated by Lemke (1965, p. 687, equation (20) with $\tilde{z}_0 = u_0$, $\tilde{z} = u$), and by Cottle and Dantzig (1968, p. 116, equation (37)). It implies $\tilde{z}_0 = 0$ since $\tilde{z}$ is nonnegative and nonzero and $d > 0$, and since the last term is nonnegative by assumption (i). Thus, $\tilde{z}^\top M\tilde{z} = 0$, and by (4.13), $\tilde{w} = M\tilde{z} \geq 0$. Assumption (ii) implies $\tilde{z}^\top q \geq 0$. We derive

a contradiction to this conclusion as follows, using (4.12), (4.11), and (i):

$$\begin{aligned} 0 &= (z+\lambda\tilde{z})^\top (w+\lambda\tilde{w}) \\ &= (z+\lambda\tilde{z})^\top (q+dz_0+M(z+\lambda\tilde{z})) \\ &\geq (z+\lambda\tilde{z})^\top (q+dz_0) \\ &= z^\top (q+dz_0)+\lambda\tilde{z}^\top (q+dz_0). \end{aligned}$$

The last term is nonpositive for all $\lambda \geq 0$ only if $\tilde{z}^\top (q+dz_0) \leq 0$, or equivalently, $\tilde{z}^\top q \leq -\tilde{z}^\top (dz_0) < 0$, contradicting (ii).

We have shown that (i) and (ii) exclude ray termination unless (4.7) has a degenerate solution where z_0 is a basic variable but has value zero. This does not pose any problem since if there is a tie in Step 1 of the algorithm, one could first check if z_0 can leave the basis even before invoking the lexicographic rule, which then yields a solution of the LCP. However, we show that this additional check, although it might possibly shorten the computation, is not necessary.

That is, assume that there is a secondary ray with endpoint $x = (w, z_0, z)^\top$ as above where z_0 is basic, but now has value zero. Because this basic feasible solution has been computed using lexicographic degeneracy resolution, there is a perturbation of (4.7) where q is replaced by $q(\varepsilon) = q + (\varepsilon, \ldots, \varepsilon^n)^\top$ for some small positive ε, where the same basis defines a (perturbed) solution x that is nondegenerate so that z_0 is positive. For the perturbed system, there is still a secondary ray since the nonpositive entering column e in (4.9) does not depend on q. With the same argument as before, we can now conclude $\tilde{z}^\top b(\varepsilon) < 0$, which is again a contradiction to (ii) since $\tilde{z}^\top q(\varepsilon) = \tilde{z}^\top q + \tilde{z}^\top (\varepsilon, \ldots, \varepsilon^n)^\top > \tilde{z}^\top q$. This shows that the theorem holds even if Lemke's algorithm encounters degenerate solutions, provided it uses the lexicographic method. □

Corollary 4.3 *Lemke's algorithm (with $d > 0$) will always find a Nash equilibrium of a bimatrix game.*

Proof: We can assume w.l.o.g. that $M > 0$, in which case the conditions of Theorem 4.2 are trivially satisfied. □

We end this section with Figure 4.2, an illustration of using Lemke's algorithm with covering vector $d = (2,1)^\top$ to find a symmetric equilibrium of the 2×2 symmetric game with C as in (4.3) (so Figure 4.2 corresponds to Figure 4.1).

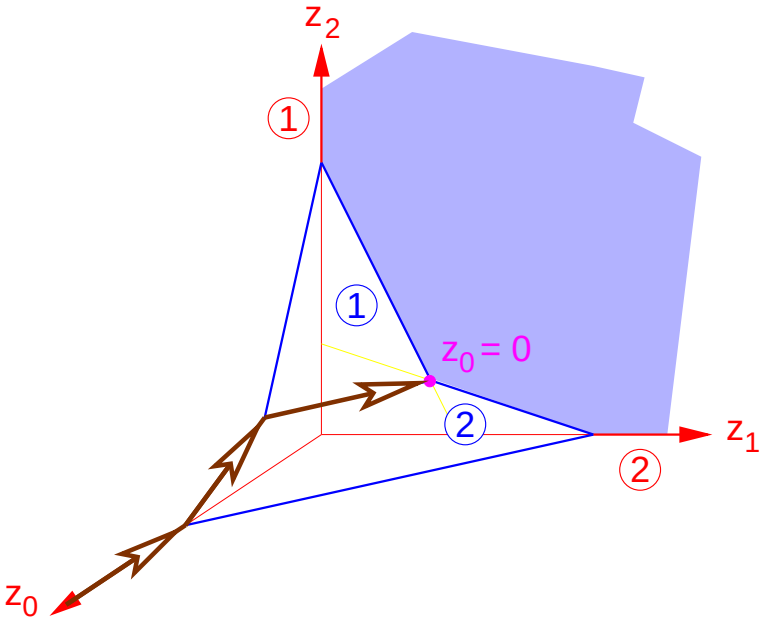

Figure 4.2 An illustration of Lemke's algorithm

4.4 The LCP map and complementary cones

In this section, we describe the LCP map and the complementary cones view of LCPs and complementary pivot methods (e.g., Eaves and Scarf (1976)). A linear program is a special case of an LCP (e.g., Smale (1983)), and Lemke's algorithm applied to a linear program corresponds to a variant of the simplex algorithm called the self-dual parametric simplex algorithm (Dantzig (1963, p. 245)). The complementary cones view is particularly useful for the probabilistic analysis of Lemke's algorithm (Megiddo (1986)), and the self-dual parametric simplex algorithm (Smale (1983), Adler and Megiddo (1985)).

We begin with some definitions. First, the definition of a cone, which is the usual, well-known operator. It is applied to a set of vectors, $X = \{x^1, \ldots, x^N\} \subseteq \mathbb{R}^n$, and produces the set of all positive combinations of these vectors,

$$\text{cone}(X) = \{\sum_{i=1}^{N} \lambda_i x^i \mid \lambda \in \mathbb{R}^N, \lambda \geq \mathbf{0}\}. \tag{4.14}$$

Let $\alpha \subseteq \{1,\ldots,n\}$. Next, we define the usual orthants of $\mathbb{R}^n$, where α denotes the set of positive unit vectors of the basis of the orthant. The α–orthant of $\mathbb{R}^n$ is

$$\text{cone}(\{e_i, -e_j \mid i \in \alpha, j \notin \alpha\}). \tag{4.15}$$

It is relevant because the LCP map, introduced below, maps orthants to complementary cones, which we define next. Given an $n \times n$ matrix M, denote by M_i the ith column of M. The complementary cone $C(\alpha)$ is

$$\text{cone}(\{M_i, -e_j \mid i \in \alpha, j \notin \alpha\}). \tag{4.16}$$

Nonnegative n-vectors z and w are a solution to the LCP (4.5) if and only if they satisfy the complementarity condition (4.6), and

$$-q = Mz - w\ . \tag{4.17}$$

With $\alpha = \{i \mid z_i > 0\}$, this is equivalent to $-q$ belonging to a complementary cone, i.e.,

$$-q \in C(\alpha) = cone(\{M_i, -e_j \mid i \in \alpha, j \notin \alpha\}). \tag{4.18}$$

See Figure 4.3 for an illustration. Let $x^+ = max\{x_i, 0\}$ and $x^- = min\{x_i, 0\}$. The LCP map, $F : \mathbb{R}^n \to \mathbb{R}^n$, is defined by

$$F(x) = Mx^+ + x^-. \tag{4.19}$$

The LCP map F, which is piecewise linear, maps orthants to complementary cones, that is, $F(\alpha\text{–orthant}) = C(\alpha)$, and F is the identity map on the negative orthant ($\alpha = \emptyset$).

4.5 Surjective LCP map for $M > 0$

In this section, we show that if $M > 0$ then the LCP map F is surjective.

Lemma 4.4 *For $M > 0$, F is surjective, that is, given $p \in \mathbb{R}^n$ there exists $x \in \mathbb{R}^n$ such that $F(x) = Mx^+ + x^- = p$.*

Proof: Let $p \in \mathbb{R}^n$ be given. We find an x such that $F(x) = p$ in two stages. First, we find the positive components of x. Let $\alpha = \{i \mid p_i > 0\}$. Consider the following LCP, which

uses only the rows $i \in \alpha$,

$$\forall i \in \alpha, \quad x_i \geq 0, \quad \sum_{j \in \alpha} (m_{ij}/p_i) x_j \geq 1, \quad x_i (\sum_{j \in \alpha} (m_{ij}/p_i) x_j - 1) = 0 \; . \tag{4.20}$$

This LCP has $M > 0$ and $q = -\mathbf{1}$. Therefore, the solutions of (4.20) are the symmetric Nash equilibria of the $|\alpha| \times |\alpha|$ symmetric game with cost matrix for player 1 equal to (m_{ij}/p_i) for $i, j \in \alpha$. Since every symmetric game has at least one symmetric Nash equilibrium (see Section 4.1), at least one solution, say $x^* \in \mathbb{R}^\alpha$, to (4.20) exists. Let $\beta = \{i \mid x_i^* > 0\}$. If $i \in \beta$, we have $(\sum_{j \in \alpha} (m_{ij}/p_i) x_j = 1$ by complementarity. We set $x_i = x_i^* > 0$ for $i \in \beta$, giving the positive components of x.

Second, we choose for all $k \notin \beta$ the nonpositive components of x. By definition, we have $p_k \leq 0$ for $k \notin \alpha$. Since M is positive and $x_j > 0$ for $i \in \beta$, we have $\sum_{j \in \beta} m_{kj} x_j > 0$ for all k, in particular $k \notin \beta$. By complementarity in (4.20), we have $\sum_{j \in \beta} m_{kj} \geq p_k$ for $i \in \alpha \setminus \beta$. So, for $k \notin \beta$, we can set $x_k \leq 0$ such that

$$\sum_{j \in \beta} m_{kj} x_j + x_k = p_k \; .$$

Thus, we have $F(x) = p$. □

A matrix M is called a P-matrix if all its principal minors are positive, i.e. $det(M_{\alpha\alpha}) > 0$ for all $\alpha \subseteq \{1, \ldots, n\}$. Equivalently, M is a P-matrix if for any q the LCP (M, q) has a unique solution. This is the case if and only if the LCP map is *bijective*; see e.g., Cottle, Pang, and Stone (1992). As an example, the matrix $M = C$ in (4.3) is a P-matrix, since $det(M_{1,1}) = 2 > 0$, $det(M_{2,2}) = 3 > 0$, and $det(M_{12,12}) = det(M) = 5 > 0$, and Figure 4.3 shows the corresponding bijective LCP map.

4.6 Lemke's algorithm and complementary cones

In this section, we describe how Lemke's algorithm can be viewed in terms of complementary cones and the LCP map (see Megiddo (1986) or Eaves and Scarf (1976)).

The algorithm with covering vector d, can be seen as inverting the piecewise linear map $F(x)$ along the line segment $[-d, -q]$:

$$F(x) = Mx^+ + x^- = (-d)(1-t) + (-q)(t) \quad (0 \leq t \leq 1) \tag{4.21}$$

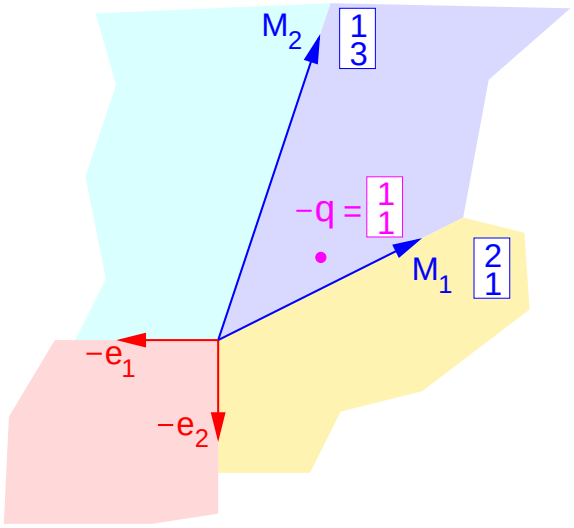

Figure 4.3 A bijective LCP map F with a P-matrix M. With $q = (1,1)^\top$, the unique solution of the LCP has support $\{z_1, z_2\}$, since $-q$ is contained uniquely in the complementary cone spanned by columns M_1 and M_2. If we exchange the columns of M it is no longer a P-matrix, the LCP map is not injective, and the LCP has three solutions, with supports $\{z_1\}$,$\{z_2\}$, and $\{z_1, z_2\}$, corresponding to the complementary cones spanned by M_1 and $-e_2$, $-e_1$ and M_2, and M_1 and M_2, respectively.

Figure 4.4 gives an illustration. At each basic solution in the standard view of the algorithm, the basic variables correspond to a nonzero components of x in (4.21), where there are exactly $n-1$ such variables excluding z_0, so we are on at some point on a facet of a complementary cone. Each pivot in the standard view of the algorithm corresponds to crossing one of the complementary cones from one facet to another. Since the LCP map is not necessarily bijective, we must specify which complementary cone is crossed next, if there is more that one cone on the other side of the current cone-facet (this clarifies what is meant by "inverting" the map F along $[-d, -q]$). In each pivoting step, one variable enters the basis and one variable leaves. It is the entering variable that determines which complementary cone is crossed next. As in the standard view of Lemke's algorithm, the entering variable is determined by the complementary pivot rule: The duplicate label (the complement of the label that was just picked up) enters the basis.

Even if the LCP map is surjective the algorithm may terminate with a secondary ray, rather than a solution to the LCP; see Section 4.8 for an example.

We now show the correspondence between (4.21) and $w \geq \mathbf{0}$, $z \geq \mathbf{0}$, $z_0 \geq 0$, (4.6), and (4.7). We divide (4.21) by t, the parameter of the convex combination of $-d$ and $-q$, which only works for positive t.

$$\text{for } t > 0 : Mx^+(1/t) + x^-(1/t) = (-d)(1-t)/t + (-q)$$

$$Mz - w = (-d)z_0 + (-q), \quad z \geq 0, w \geq 0, \text{ and (4.6) holds.}$$

Thus, the change of variables $z = x^+/t$, $w = -x^-/t$, and $z_0 = (1-t)/t$ shows the correspondence. So, $0 < t \leq 1$ corresponds to $\infty > z_0 \geq 0$.

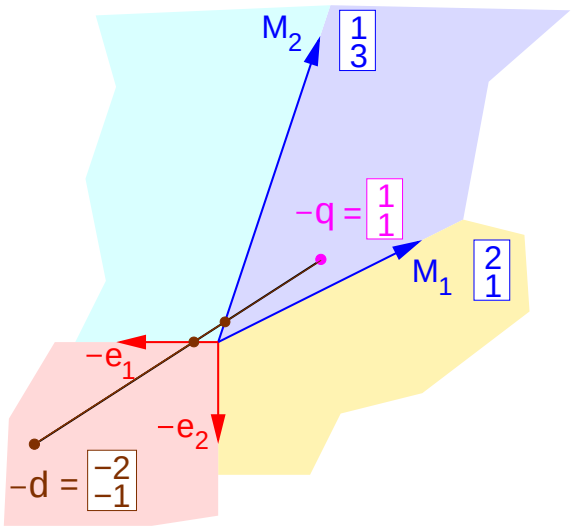

Figure 4.4 Lemke's algorithm as inverting the piecewise linear LCP map F along the line segment $[-d, -q]$. The computation involves four basic solutions, corresponding to the endpoints $-d$ and $-q$, and the facets of the complementary cones that the line segment crosses. Thus, the computation involves three pivoting steps.

Lemke's algorithm succeeds only if

$$F^{-1}[(1-t)(-d) + t(-q)] \neq \emptyset \text{ for every } t \in [0,1] \,. \tag{4.22}$$

Furthermore, viewed in terms of complementary cones, it is clear that when the LCP map is surjective (so (4.22) is satisfied) ray termination can only occur if $-d$ is contained in more than one complementary cone. Ray termination is related to the non-monotonicity of t, or equivalently of $z_0 = (1-t)/t$. When the LCP map is not bijective, the preimage of part of the line segment $[-d, -q]$ may be "reflected back", so t decreases, into a cone (overlapping the previous cone) in an unbounded direction. As shown by Theorem 4.2, this is not possible for bimatrix games when starting in the negative orthant, i.e. $d >$

0 (because the preimage of the negative orthant under F is unique). For $M > 0$, ray termination is not possible in dimension 2, but an example in dimension 3 is given at the end of Section 4.8.

In Section 4.8, we describe a simple extension of Lemke's algorithm: Rather than requiring $d > 0$, and thus starting in the negative orthant (with $\alpha = \emptyset$ in (4.16)), we can start in other complementary cones. In that case the algorithm may fail, even for bimatrix games, when more than one complementary cone contains $-d$. For a general LCP, Lemke's algorithm may fail even if $d > 0$, but starting in other cones may allow one to find a solution. Of course, if there is at least one solution, one may be lucky and decide to start in one of the complementary cones containing $-q$; in this case, the extension is equivalent to guessing the support of a solution, solving a small system of linear equations by pivoting, and checking for feasibility.

4.7 A unified view of Lemke's algorithm and the Lemke–Howson algorithm

In this section, we give a unified view of the LH algorithm and Lemke's algorithm. The standard initialization of Lemke's algorithm requires that $d > 0$, and traditionally the covering vector $d = \mathbf{1}$ is used. We show that the LH algorithm, with missing label k, corresponds to Lemke's algorithm with a covering vector $d = -e_k$. Since this covering vector is not positive, the first issue we must deal with is initialization: We initialize by pivoting z_0 into the basis and w_k out. After this pivot, the basic solution is still infeasible. According to the complementary pivot rule, we pivot z_k in (M_k is positive), and we start in the cone $C(\{k\})$.

Lemma 4.5 *Lemke's algorithm with a covering vector $d = -e_k$, initialized in this way, corresponds to the Lemke–Howson (LH) algorithm, with missing label k.*

Proof: The variable z_0, which does not feature in the standard LH algorithm, is only involved in the first and last pivots of Lemke's algorithm with $d = -e_k$. In the last pivot z_0 finally leaves the basis to give an equilibrium. There are two cases to distinguish:
(i) $z_k > 0$ in the equilibrium at the end of the LH path. Then, in the final pivot the entering

variable is identical to the entering variable of the final pivot in the standard LH algorithm. In this case, Lemke's algorithm with $d = -e_k$ takes one more pivot than the standard LH algorithm.

(ii) $z_k = 0$ in the equilibrium at the end of the LH path. Then the penultimate pivot is identical to the final pivot of the standard LH algorithm, and in the final pivot the entering variable is w_k. So in this case, Lemke's algorithm with $d = -e_k$ takes two more pivots than the standard LH algorithm. □

We illustrate these two cases with the symmetric LH algorithm applied to the following 3×3 symmetric game $(C, C^\top)$, defined by the cost matrix

$$C = \begin{pmatrix} 4 & 1 & 2 \\ 1 & 4 & 2 \\ 2 & 2 & 1 \end{pmatrix}.$$

It has a unique pure strategy symmetric equilibrium, where player 1 plays the third row ($z_1 = 0$, $z_2 = 0$, and $z_3 > 0$). Thus, applying the symmetric LH algorithm with z_0 will require one extra pivot with missing label 3, but two extra pivots with missing labels 1 or 2. This is shown in Figure 4.5 for missing label 3, and in Figure 4.6 for missing label 2.

LH no z_0		LH with z_0	
Pivot		Pivot	
In	Out	In	Out
		z_0	w_3
z_3	w_3	z_3	z_0

Figure 4.5 Pivots for the symmetric LH algorithm with missing label 3, with and without z_0.

LH no z_0		LH with z_0	
Pivot		Pivot	
In	Out	In	Out
		z_0	w_2
z_2	w_1	z_2	w_1
z_1	w_3	z_1	w_3
z_3	z_1	z_3	z_1
w_1	z_2	w_1	z_2
		w_2	z_0

Figure 4.6 Pivots for the symmetric LH algorithm with missing label 2, with and without z_0.

4.8 Starting Lemke's algorithm in arbitrary cones

The standard version of Lemke's algorithm uses a positive covering vector d, and inverts the piecewise linear map F along the line segment $[-d,-q]$. In the previous section we showed that, when taking $d=-e_k\leq 0$, which corresponds to starting in the cone $C(\{k\})$, Lemke's algorithm corresponds to the LH algorithm with missing label k. In fact, we can start the algorithm in any cone, as we describe in this section. However, unlike when $d>0$ or $d=-e_k$, we are not guaranteed to avoid ray termination for $M>0$. Despite this potential problem, extending Lemke's algorithm in this way may be useful. Indeed, when Lemke's algorithm is run on a general LCP, there is no way to guarantee avoiding ray termination.

We can try any d, provided we know $\bar{x}$ with $F(\bar{x})=-d$. So, for example, as we will do, we can choose d in an arbitrary cone and let $d=-F(\bar{x})$. Instead of specifying a covering vector, we specify an $\bar{x}\in\mathbb{R}^n$. In fact, the covering vector d is not explicitly used in this extended version of the algorithm, only $\bar{x}$. In the standard version of Lemke's algorithm we do use d, but this is because d is positive, so $F^{-1}(-d)=-d=\bar{x}$.

Here is the modified initialization of Lemke's algorithm. Steps 1 and 2 are the same as the standard Lemke's algorithm on page 74. As usual, $\alpha=\{i\mid x_i>0\}\subseteq\{1,\ldots,n\}$.

0. (Initialization.) Input $q,\bar{x},M$. Let the basic variables be $w_1,\ldots,w_n$, with $w=b$ and $z=0$. At this point z_0 is *not* in the system. Pivot the variables $\{z_i\mid i\in\alpha\}$ into the basis, in any order, *with the respective complement as the leaving variable*. If $b\geq 0$, then stop: this is a solution to (4.5). Otherwise, augment the system with the nonbasic variable z_0, with coefficients $|\bar{x}|$. Then, consider the index i such that $-b_i/d_i$ is maximal, where $b_i<0$; that index is unique by nondegeneracy. Pivot with w_i leaving and z_0 (at value $-b_i/d_i$) entering the basis. The resulting basic solution is feasible and almost complementary. Choose the complement z_i of w_i as the new entering variable.

   Define the matrix B as follows.

$$B_i=\begin{cases}M_i, & i\in\alpha,\\ -e_i, & \text{otherwise.}\end{cases}\tag{4.23}$$

We could have included z_0 in the system from the beginning (with coefficients d), but this would be a little wasteful, since the coefficients of z_0 after the first $|\alpha|$ pivots are just the $|\bar{x}_i|$'s by construction: The pivoting operations that bring the variables $\{z_i \mid i \in \alpha\}$ into the basis are equivalent to multiplying the original system (4.7) by the inverse of the matrix B in (4.23). So

$$B^{-1}d = |\bar{x}|. \tag{4.24}$$

This modification of Lemke's algorithm first tests if there is an equilibrium corresponding to the support α defined by $\bar{x}$. If there is one, the algorithm terminates after the initialization step. This is done in essentially the same way as a support enumeration algorithm, by solving the small system of linear equations and then testing for feasibility. So, this extension of Lemke's algorithm can easily be combined with support enumeration.

We end this section with a 3×3 example of ray termination for $M > 0$.

$$\mathit{Input}: q = (-1,-1,-1)^\top, \quad \bar{x} = (-1,1,1)^\top, \quad M = \begin{pmatrix} 4 & 1 & 3 \\ 1 & 3 & 2 \\ 2 & 2 & 1 \end{pmatrix}. \tag{4.25}$$

We start at the infeasible solution $z = 0$, $w = q$, which as a dictionary is

$$\begin{aligned} w_1 &= -1 + 4z_1 + \ z_2 + 3z_3 \ , \\ w_2 &= -1 + \ z_1 + 3z_2 + 2z_3 \ , \\ w_3 &= -1 + 2z_1 + 2z_2 + \ z_3 \ . \end{aligned} \tag{4.26}$$

Pivoting in z_2 and then z_3 gives the dictionary

$$\begin{aligned} w_1 &= -3 + 13z_1 + 5w_2 - 7w_3 \ , \\ z_2 &= -1 - \ 3z_1 - \ w_2 + 2w_3 \ , \\ z_3 &= -1 + \ 4z_1 + 2w_2 - 3w_3 \ . \end{aligned} \tag{4.27}$$

If w and z were nonnegative we would have an equilibrium, and be done. This is not the case, so we augment the system with the nonbasic variable z_0, which has coefficients $|\bar{x}|$. As mentioned previously, we could have included z_0 in the system from the beginning, but this would be a little wasteful, since the coefficients of z_0 are just the $\bar{x}_i$'s by construction. The two pivoting operations that put the variables z_2 and z_3 into the basis are equivalent

to multiplying the original system (4.26) by the inverse of $[-e_1\ M_2\ M_3]$, which is

$$\begin{pmatrix} -1 & 1 & 3 \\ 0 & 3 & 2 \\ 0 & 2 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} -1 & 5 & -7 \\ 0 & -1 & 2 \\ 0 & 2 & -3 \end{pmatrix},$$

and $[-e_1\ M_2\ M_3]^{-1}d = |\bar{x}|$ by (4.24). The augmented dictionary is

$$\begin{aligned} w_1 &= -3 + 13z_1 + 5w_2 - 7w_3 + 1z_0\ , \\ z_2 &= -1 - \ 3z_1 - \ w_2 + 2w_3 + 1z_0\ , \\ z_3 &= -1 + \ 4z_1 + \ 2w_2 13w_3 + 1z_0\ . \end{aligned}$$

Next we bring z_0 into the basis and start complementary pivoting. So z_0 enters the basis and w_1 leaves. Then z_1 enters the basis and z_3 leaves, which gives the dictionary

$$\begin{aligned} 9z_0 &= -1 + -4w_1 - 6w_2 + 11w_3 + 13z_3\ , \\ 9z_1 &= -2 + -7w_1 - 6w_2 + \ 4w_3 - \ \ z_3\ , \\ 9z_2 &= -4 + \ \ 1w_1 - 3w_2 + 17w_3 + 16z_3\ . \end{aligned}$$

Next we try to bring w_3 into the basis, but because all coefficients of w_3 are positive, there is no restriction on feasibility as w_3 is increased, and w_3 and thus z_0 can become arbitrarily large. So, the algorithm terminates with a secondary ray.

In summary, we may start Lemke's algorithm in any complementary cone, as described above. When $M > 0$, which includes the case of bimatrix games, we are guaranteed to terminate with a solution if $d > 0$. Starting in other complementary cones, we may get ray termination, as the previous example shows, but this extension of Lemke's algorithm may still be useful (also for more general LCPs).

Chapter 5

Conclusions and Open Questions

5.1 Recent results on PPAD-completeness

In complexity theory, function problems are those that, as well as requiring a "yes/no" answer to a decision problem, require also a solution to accompany a "yes" answer. The class FNP of function problems in NP are those where a solution is verifiable in polynomial time. If the decision problem is known to have a "yes" answer for any valid input, the problem is said to be *total*. NASH belongs to the complexity class TFNP of total function problems in NP (Papadimitriou (1994a, p. 229)): For the two-player case, with the bimatrix game as input, the required output are the mixed strategy probabilities, where the equilibrium property is verified in polynomial time. The class TFNP does not have complete problems unless NP = co-NP (Megiddo and Papadimitriou (1991)).

Subclasses of TFNP are characterzied by the "proof technique" used to show totality of the problem. For the subclass of TFNP called PPAD, the problems are known to have a solution by a polynomial parity argument for directed graphs (Papadimitriou (1994b, 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. Formally, PPAD is defined as those problems in TFNP that are reducible to the following problem:

END OF THE LINE:

Given two boolean circuits S and P, each with n input bits and n output bits, such that $P(0^n) = 0^n \neq S(0^n)$, find an input $x \in \{0,1\}^n$ such that $P(S(x) \neq x$ or $S(P(x)) \neq x \neq 0^n$.

Intuitively, END OF THE LINE creates an exponential-size directed graph with vertex set $\{0,1\}^n$ and an edge from x to y whenever both $y = S(x)$ and $x = P(y)$ (S and P stand for successor and predecessor, respectively). All nodes in this graph have indegree and outdegree at most one, and there is at least one source, namely 0^n, so according to the parity argument there must be a sink. We seek either a sink, or a source other than 0^n.

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 neighbour 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. NASH belongs to PPAD, since using the LH algorithm it is easily reduced to END OF THE LINE. The LH algorithm uses a trivial *artificial equilibrium* as the 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" (successor). 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 (Shapley (1974), see von Schemde (2005) for a recent discussion); see also Figure 2.1 above.

The parity argument may be inefficient if the paths are not of polynomial length. We have shown explicitly that this inefficiency may occur for NASH, by giving games that produce exponentially long LH paths.

For more than two players, a game may have only equilibria with irrational solutions (Nash (1951)). N-NASH, which is the problem of finding an *approximate* Nash equilibrium of an N-player game, also belongs to PPAD, shown by a reduction to END OF THE LINE via Brouwer's Theorem and Sperner's Lemma. Recently, there have been several important contributions towards understanding the computational complexity of N-NASH and NASH, in a series of preprints, which we now describe.

The paper by Papadimitriou and Goldberg (2005) shows how to reduce N-NASH to 4-NASH. In the paper by Daskalakis, Goldberg, and Papadimitriou (2005), it is shown that 4-NASH is PPAD-complete, which implies that *any* other problem in the complexity class

PPAD, which includes most other problems related to equilibrium computation (Papadimitriou (1994b)), can be reduced to 4-NASH. Then, Chen and Deng (2005a), and independently Daskalakis and Papadimitriou (2005), showed that 3-NASH is PPAD-complete, and very recently, Chen and Deng (2005b) showed that 2-NASH is also PPAD-complete. This would mean that a two-player game represents already the full complexity of the Nash equilibrium problem (with respect to polynomial-time algorithms). This result makes it unlikely that a polynomial-time algorithm for 2-NASH (and therefore NASH) exists, since such an algorithm would give a polynomial-time algorithm, for example, for computing Brouwer fixpoints. For large classes of algorithms that compute Brouwer fixed points of a given function (namely "black box" algorithms that merely evaluate the function, rather than using its description), Hirsch, Papadimitriou, and Vavasis (1989) have shown exponential lower bounds.

5.2 Open questions

A linear program (LP) can be formulated as an LCP, which captures the complementary slackness conditions that characterize a pair of optimal solutions to the primal and dual LP. Applied to such an LCP, Lemke's algorithm corresponds to the self-dual parametric simplex algorithm for solving LPs (Dantzig (1963, p. 245)). A special case of this is a parametric simplex algorithm where the right-hand side is parameterized, which Murty (1980) has shown to be exponential. Another special case is the parametric-objective simplex algorithm, which Goldfarb (1983; 1994) has shown to be exponential.

For linear programming, despite the worst-case exponential behaviour of all known pivoting algorithms, they tend to work well in practice; this is also backed up by theoretical results (Spielman and Teng (2004)). For various models of random input data, the expected running time of the simplex algorithm is polynomial, in contrast to the worst-case exponential behaviour (see Todd (2001, p. 422) for a survey). Smale (1983) and Adler and Megiddo (1985) give a probabilistic analysis for the self-dual parametric simplex algorithm, using its description as a special case of Lemke's algorithm.

This raises the following questions in the context of games. First, what is the expected running time of the LH algorithm? Megiddo (1986) analyzes Lemke's algorithm for

random general LCPs (not derived from games), and shows that its expected running time is exponential for the standard version of Lemke's algorithm, and quadratic for a modified version. Bárány, Vempala, and Vetta (2005) show that with high probability, random games have equilibria with small support, so that an equilibrium is quickly found by support enumeration. Although this result does not concern the LH algorithm, it suggests that random games are not "hard to solve".

Secondly, is there a randomized variant of Lemke's algorithm that solves our games quickly on average? Von Stengel, van den Elzen, and Talman (2002) use a covering vector derived from a starting pair of mixed strategies to solve two-player games with Lemke's algorithm, and give a game-theoretic interpretation of the resulting path. The starting pair can be chosen randomly. Section 4.8 shows that more general random choices of the covering vector are also possible (which is true for any LCP).

For the same model of random input data used by Smale (1983), Megiddo (1986) shows that the expected running time of Lemke's algorithm is exponential when the covering vector is $(1,\ldots,1)^\top$, but quadratic when the covering vector is $(\varepsilon,\varepsilon^2,\ldots,\varepsilon^n)^\top$. If $d=-q$, degeneracy resolution must be used, since the path will go through the origin, which is in every complementary cone. By using appropriate lexicographic degeneracy resolution, it may be possible to emulate the good behaviour that results from the covering vector $(\varepsilon,\varepsilon^2,\ldots,\varepsilon^n)^\top$, used by Megiddo. Such a starting point has the advantage that ray termination is not possible. Initial empirical evidence is encouraging, and this certainly deserves further theoretical study. An obvious related question is how the games constructed in this thesis are solved by such an algorithm. If the algorithm performs well in these cases, the natural question is whether there are hard instances for such an algorithm, i.e. Lemke's algorithm with $d=-q$ and appropriate lexicographic degeneracy resolution. Furthermore, when lexicographic degeneracy resolution is used the algorithm will depend on the order of the rows (or equivalently the order of the powers in the lexicographic covering vector), so randomly permuting the payoff matrices is another possibility for achieving good expected running times.

A related paper is Megiddo (1985), which discusses how the parametric self-dual simplex method with different starting points, which corresponds to Lemke's algorithm

with different covering vectors, can reproduce seemingly different algorithms, such as the variable dimension algorithm of van der Heyden (1980).

A different open problem is how to generate "numerically stable" game matrices with our construction. We use cyclic polytopes, using points on the moment curve, which give rise to notoriously ill-conditioned matrices. As a consequence, numerical problems arise when the pivoting steps are implemented using floating-point arithmetic. It would be good to have "hard instances" of games without this additional complication. These numerical problems may possibly be avoided by using points on the so-called trigonometric moment curve (see Ziegler (1995, p. 75f) or Grünbaum (2003, p. 67)). An open question is the required numerical accuracy of these points.

The recent claim by Chen and Deng (2005b) that NASH is PPAD-complete, in conjunction with the results of Hirsch, Papadimitriou, and Vavasis (1989), suggests that is unlikely that a polynomial-time algorithm exists for NASH. In any case, since not even a subexponential algorithm is known for NASH, the existence of such an algorithm is an intriguing open question.

A well-known open problem is the following: Given a matrix M and a vector q, either find a solution to the LCP (M,q), or display a proof that M is not a P-matrix. This problem is, like NASH, in PPAD; the reduction to END OF THE LINE uses Lemke's algorithm, which will find a solution to the LCP (M,q) if M is a P-matrix (ray termination proves that M is not a P-matrix). As for any problem in TFNP, if the problem is NP-hard then NP = co-NP (Megiddo (1988) and Megiddo and Papadimitriou (1991)). The computational complexity of this problem is an intriguing open question. Is this problem PPAD-complete? The problem of deciding if a matrix is *not* a P-matrix is NP-complete (Coxson (1994)).

Appendix A

A.1 Generating game matrices for $\Gamma(d,d)$

In this appendix, we describe how to obtain games from representations of cyclic polytopes. In principle, this has already been described in Proposition 2.1 of von Stengel (1999) for general polytopes. In our case, as well as in the construction of games with a large number of equilibria in von Stengel (1999), the polytopes are dual cyclic polytopes in dimension d with $2d$ facets, with a labelling of the facets of each polytope that has a certain structure. This structure allows a further simplification: Only one of dual cyclic polytopes, say P, has to be brought into the form (2.4), where d of the inequalities are simply nonnegativities and the other d inequalities define the payoff matrix of one player, here B. The other polytope, and thus the payoff matrix A of the other player, is then simply obtained by a suitable permutation of the rows and columns of $B^\top$. We first explain this construction, which is summarized in Proposition A.1 below.

Secondly, we apply this to a representation of cyclic polytopes in dimension $d = 6$ derived from the so-called trigonometric moment curve. In this low dimension, the coordinates on that curve can be approximated by small integers, which gives rise to small game matrix entries.

As indicated at the beginning of Section 3.2, a standard way of obtaining a cyclic polytope in dimension d with $2d$ vertices is, first, to consider $2d$ points $\mu(t_i)$ on the moment curve $\mu\colon t \mapsto (t,t^2,\ldots,t^d)^\top$ for $1 \le i \le 2d$. Suppose that $t_1 < t_2 < \cdots < t_{2d}$. Then the vertices of that polytope are characterized by the 0-1 strings fulfilling the Gale evenness condition. The *polar* (or dual) polytope (see see Ziegler (1995) or Grünbaum (2003)) is obtained by translating the polytope so that it has the origin $\mathbf{0}$ in its interior, for exam-

ple by subtracting the arithmetic mean $\bar{\mu}$ of the points $\mu(t_i)$ from each such point. The resulting vectors $c_i = \mu(t_i) - \bar{\mu}$ then define the polar cyclic polytope

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

As described in von Stengel (1999, p. 560), if $P' = \{\, z \in \mathbb{R}^d \mid Cz \leq \mathbf{1},\ Dz \leq \mathbf{1} \,\}$ with $d \times d$ matrices C and D, then an affine transformation of P' is given by

$$P = \{\, x \in \mathbb{R}^d \mid x \geq \mathbf{0},\ -DC^{-1}x \leq r \,\}, \qquad r = \mathbf{1} - DC^{-1}\mathbf{1}. \tag{A.2}$$

Since $\mathbf{0}$ is a vertex of the simple polytope P, the vector r is positive, and the second d inequalities in (A.2) can be re-normalized so that the right hand side is one. With the diagonal matrix S with entries $s_{ii} = 1/r_i$ with r as in (A.2), and $s_{ij} = 0$ for $i \neq j$, we can rewrite (A.2) as

$$P = \{\, x \in \mathbb{R}^d \mid x \geq \mathbf{0},\ -SDC^{-1}x \leq \mathbf{1} \,\}. \tag{A.3}$$

Affine transformations leave the combinatorial structure (that is, the face incidences) of a polytope unchanged, so P is a dual cyclic polytope with vertices characterized by Gale evenness strings. These Gale evenness strings refer to the $2d$ inequalities defining P according to the ordering in (A.3), that is, $x_1 \geq 0$ being the first inequality obtained from the first point $\mu(t_1)$ on the moment curve, $x_2 \geq 0$ corresponding to $\mu(t_2)$, and so on.

Consider the polytope $\overline{Q}$ defined by

$$\overline{Q} = \{\, \bar{y} \in \mathbb{R}^d \mid -SDC^{-1}\bar{y} \leq \mathbf{1},\ \bar{y} \geq \mathbf{0} \,\}, \tag{A.4}$$

which is identical to P in (A.3) except that the first and last d inequalities are interchanged.

In a dual cyclic polytope like P in (A.3), each inequality defines a facet (obtained by converting the inequality to an equality). We say that a facet of P has *label* k (for $k = 1, \ldots, 2d$) if it corresponds to the kth inequality in the description of the polytope in (A.3). Similarly, a facet of $\overline{Q}$ has label k if it corresponds to the kth inequality in (A.4). If P and $\overline{Q}$ in (A.3) and (A.4) are the polytopes P and Q in (2.4), they define a symmetric bimatrix game with payoff matrices (A, B) where $B^\top = A = -SDC^{-1}$.

The vertices of a dual cyclic polytope are given by the sets of d facets each vertex lies on. Encoded as bitstrings, these sets are characterized by the Gale evenness condition explained at the beginning of Section 3.2, with $G(d)$ as the set of these bitstrings. We

assume d is even. Then the Gale evenness condition is preserved by a cyclic rotation of the bitstrings, in particular by d positions, as used in the definition (A.4) of $\overline{Q}$. Thus, the vertices of both P and $\overline{Q}$ correspond to the Gale evenness strings in the set $G(d)$. A bitstring u in $G(d)$ defines the vertex x of P obtained by converting the kth inequality in (A.3) to an equality whenever $u_k = 1$, for $k = 1, \ldots, 2d$. In the same manner, v in $G(d)$ defines the vertex y of $\overline{Q}$ where the kth inequality in (A.4) is binding whenever $v_k = 1$.

In our construction, as well as in von Stengel (1999), the polytopes P and Q in (2.4) are dual cyclic polytopes but the games are not symmetric, because the facets of Q are not labelled in their original order. Instead, a certain permutation λ is used to obtain Q from $\overline{Q}$, by letting the kth facet of $\overline{Q}$ in the description (A.4) have label $\lambda(k)$ in Q, for $k = 1, \ldots, 2d$. In our construction, we used the permutation $\lambda = l'$ defined in (3.3). With $\lambda(S) = \{\lambda(k) \mid k \in S\}$ for $S \subseteq 1, \ldots, 2d$, this permutation has the property

$$\lambda(\{1, \ldots, d\}) = \{1, \ldots, d\} \tag{A.5}$$

(and thus $\lambda(\{d+1, \ldots, 2d\}) = \{d+1, \ldots, 2d\}$). This condition implies that the pair $(u, v) = e_0$ in (3.4) is complementary. The corresponding vertex pair of $P \times Q$ is the artificial equilibrium $(\mathbf{0}, \mathbf{0})$. (Property (A.5) also implies that e_1 in Lemma 3.3 is complementary, which defines the completely mixed equilibrium.)

The following proposition describes the construction of a bimatrix game (A, B) using P in (A.3), and Q defined by $\overline{Q}$ in (A.4) with labels given by a permutation λ fulfilling (A.5). The proposition shows how to obtain A from $B^\top$ by permuting rows and columns suitably.

Proposition A.1 *Consider a pair of dual cyclic polytopes in dimension d with $2d$ facets, with each vertex set represented by the set of Gale evenness strings $G(d)$. Let λ be a permutation of $\{1, \ldots, 2d\}$ that fulfills* (A.5)*. For $k = 1, \ldots, 2d$, a vertex u in $G(d)$ of the first polytope has the labels k where $u_k = 1$, a vertex v in $G(d)$ of the second polytope has the labels $\lambda(k)$ where $v_k = 1$. A vertex pair (u, v) is complementary if it has all labels. Then a $d \times d$ bimatrix game (A, B) with Nash equilibria corresponding to complementary vertex pairs, where the artificial equilibrium corresponds to e_0 in* (3.4)*, is given by $B^\top = -SDC^{-1}$ as in* (A.3)*, using a representation* (A.1) *of a dual cyclic polytope consistent with the Gale evenness ordering. The matrix entries $a(i, j)$ of A are obtained from the*

matrix entries $b(i,j)$ of B by

$$a(\lambda(i),\lambda(j+d)-d) = b(j,i) \qquad (1 \le i,j \le d). \tag{A.6}$$

Proof: For the characterization of equilibria, the combinatorial structure of the dual cyclic polytopes suffices, as given by the Gale evenness strings $G(d)$. Both P in (A.3) and $\overline{Q}$ in (A.4) are representations of such polytopes. By assumption, for $k = 1,\ldots,2d$, the kth inequality of (A.3) has label k, and the kth inequality of (A.4) has label $\lambda(k)$.

Let $B^\top = -SDC^{-1}$, define the matrix A with entries $a(i,j)$ by (A.6), and let

$$Q = \{\, y \in \mathbb{R}^d \mid Ay \le \mathbf{1},\ \ y \ge \mathbf{0} \,\}\ . \tag{A.7}$$

The polytopes P and Q in (A.3), (A.7) correspond to the bimatrix game (A,B), as in (2.4). The facets of Q have labels in the order of the inequalities in (A.7). It suffices to show that these labels k correspond to the labels $\lambda(k)$ of $\overline{Q}$ stated above.

In detail, the inequalities in (A.4) are

$$\begin{aligned} \overline{Q} = \{\, \overline{y} \in \mathbb{R}^d \mid \sum_{j=1}^{d} b(j,i)\, \overline{y}_j \le 1 \qquad & (1 \le i \le d), \\ \overline{y}_j \ge 0 \qquad & (1 \le j \le d)\ \}\ . \end{aligned} \tag{A.8}$$

For $1 \le j \le d$, the $(d+j)$th inequality in (A.8) has label $\lambda(d+j)$. Hence, it should appear as the $\lambda(d+j)$th inequality in (A.7), which by (A.5) is the inequality $y_{\lambda(d+j)-d} \ge 0$. This is achieved by the correspondence between y in (A.7) and $\overline{y}$ in (A.8) given by $y_{\lambda(d+j)-d} = \overline{y}_j$.

The ith of the first d inequalities in (A.8), for $1 \le i \le d$, has label $\lambda(i)$. It should appear as the $\lambda(i)$th inequality in (A.7). That inequality has the form

$$\sum_{l=1}^{d} a(\lambda(i),l)\ y_l \le 1\,,$$

which by (A.5) can be rewritten as

$$\sum_{j=1}^{d} a(\lambda(i),\lambda(d+j)-d)\ y_{\lambda(d+j)-d} \le 1\,,$$

which by (A.6) is the ith inequality of (A.8) as claimed. □

Consider the following 6×6 bimatrix game (A,B) with

$$A = \begin{bmatrix} -180 & 72 & -333 & 297 & -153 & 270 \\ -30 & 17 & -33 & 42 & -3 & 20 \\ -81 & 36 & -126 & 126 & -36 & 90 \\ 90 & -36 & 126 & -126 & 36 & -81 \\ 20 & -3 & 42 & -33 & 17 & -30 \\ 270 & -153 & 297 & -333 & 72 & -180 \end{bmatrix}, \quad B = \begin{bmatrix} 72 & 36 & 17 & -3 & -36 & -153 \\ -180 & -81 & -30 & 20 & 90 & 270 \\ 297 & 126 & 42 & -33 & -126 & -333 \\ -333 & -126 & -33 & 42 & 126 & 297 \\ 270 & 90 & 20 & -30 & -81 & -180 \\ -153 & -36 & -3 & 17 & 36 & 72 \end{bmatrix}.$$

The matrix A is obtained from B via (A.6) with $\lambda = l'$ in (3.3). The matrix B is obtained as in Proposition A.1. The underlying representation (A.1), however, is not based on points (t,t^2,t^3,t^4,t^5,t^6) of the moment curve, but on points $\nu(t)$ of the *trigonometric* moment curve, $\nu(t) = (\cos t, \sin t, \cos 2t, \sin 2t, \cos 3t, \sin 3t)$. These points also give rise to cyclic polytopes (see Ziegler (1995, p. 75f) or Grünbaum (2003, p. 67)). For $t = i\pi/6$ for $i = 1,\ldots,12$, the first pair of coordinates of $\nu(t)$ denote the vertices of a regular 12-gon, the second pair those of a regular hexagon, used twice, and the third pair those of a square, used three times. The origin is in the interior of the convex hull of these vertices, so the polytope does not have to be translated to obtain its polar. The combinatorial structure is preserved by choosing suitable integer coordinates near the points on the circle, which are shown in Figure A.1; payoffs have been multiplied by 18 to obtain integers. (The square is represented perfectly; choosing as its vertices instead the points $(1,0)$, $(0,1)$, $(-1,0)$, $(0,-1)$, say, would not change B as the affine transformation that produces (A.2) always gives the unit vectors as the normal vectors of the first d facets of P.) It is an open problem to find suitable approximations with small integers in higher dimensions that preserve the combinatorial structure.

The bimatrix game (A', B) with

$$A' = \begin{bmatrix} -81 & 36 & -126 & 126 & -36 & 90 \\ -180 & 72 & -333 & 297 & -153 & 270 \\ 20 & -3 & 42 & -33 & 17 & -30 \\ -30 & 17 & -33 & 42 & -3 & 20 \\ 270 & -153 & 297 & -333 & 72 & -180 \\ 90 & -36 & 126 & -126 & 36 & -81 \end{bmatrix}$$

is obtained from the permutation $\lambda(k) = k - (-1)^k$ for $1 \le k \le 12$ in (A.6). This permutation is used in von Stengel (1999), and the game (A', B) has 75 equilibria.

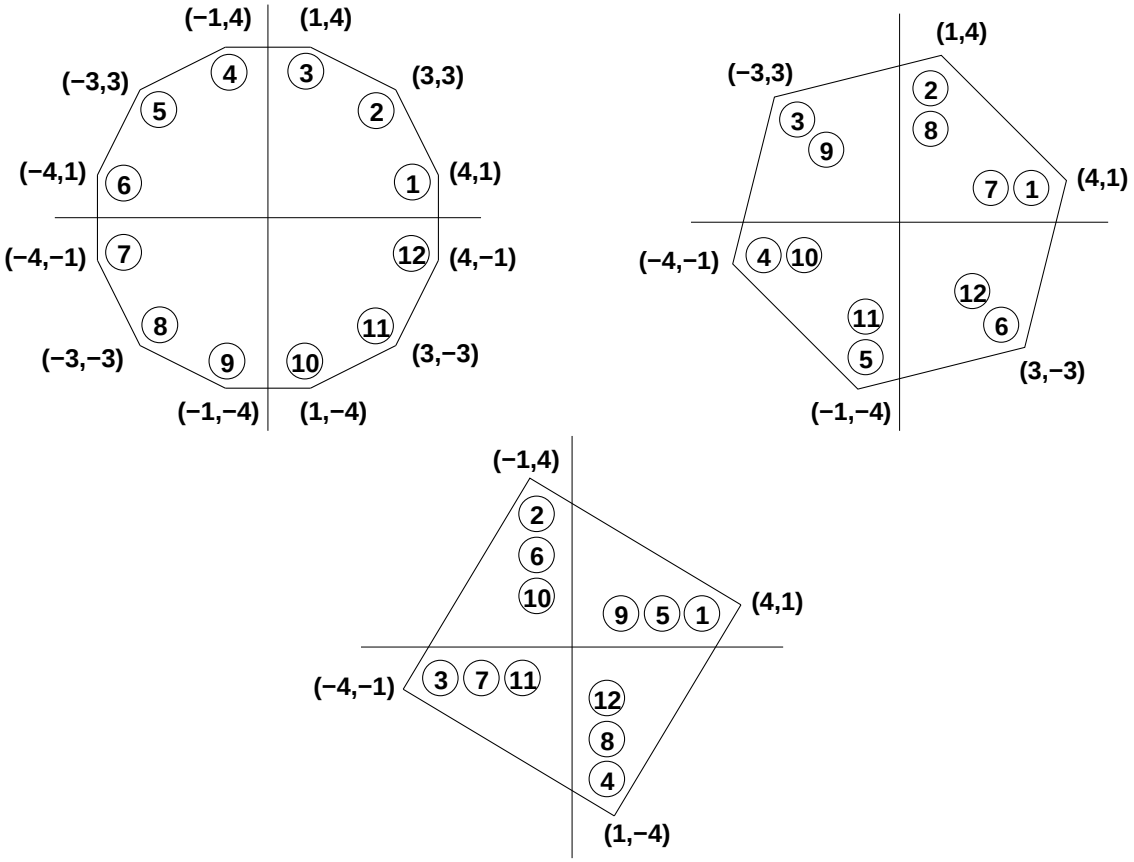

Figure A.1 Approximation of points on the trigonometric moment curve by small integers. The circled numbers refer to the labels $i = 1, \ldots, 12$ of the vertices, which become facets in the dual cyclic polytope P.

A.2 Examples of path lengths and paths

The following figures show the empirical evidence leading to Theorems 3.9, 3.10, and 3.15. Figure A.2 shows the path lengths and their exponential growth, and that the lengths of the short paths $\pi(d, 3d/2)$ for $d = 2, 4, 6, \ldots$ are given by the Fibonacci numbers times two, with every third Fibonacci number omitted. Figures A.5, A.9, A.8, (which appeared in a different format in Section 3 as Figures 3.3, 3.5, and 3.4 respectively) and A.10 illustrate Theorem 3.9, and Figures 8, 13, and 14 show cases of Theorem 3.10.

A.2.1 LH path lengths for $\Gamma(d,d)$ and $\Gamma(d,2d)$

	$\Gamma(d,d)$				
label	d				
	2	4	6	8	10
1	4	20	88	376	1596
2	4	8	24	92	380
3	4	8	24	92	380
4	4	20	24	40	108
5		10	24	40	108
6		10	88	92	108
7		10	36	92	108
8		10	36	376	380
9			16	146	380
10			16	146	1596
11			36	42	612
12			36	42	612
13				42	152
14				42	152
15				146	68
16				146	68
17					152
18					152
19					612
20					612
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					

	$\Gamma(d,2d)$				
label	d				
	2	4	6	8	10
1	4	20	88	376	1596
2	4	8	24	92	380
3	5	8	24	92	380
4	4	20	24	40	108
5	4	21	24	40	108
6	5	10	88	92	108
7		21	89	92	108
8		10	36	376	380
9		10	89	377	380
10		21	16	146	1596
11		10	89	377	1597
12		21	36	42	612
13			36	377	1597
14			89	42	152
15			16	377	1597
16			89	146	68
17			36	146	1597
18			89	377	152
19				42	1597
20				377	612
21				42	612
22				377	1597
23				146	152
24				377	1597
25					68
26					1597
27					152
28					1597
29					612
30					1597

Figure A.2 LH path lengths of $\Gamma(d,d)$ and $\Gamma(d,2d)$ for different missing labels.

A.2.2 Sample LH paths for $\Gamma(d,d)$

In the following figures, each row displays two pivoting steps, one in P and one in Q, so the number of the last row is to be multiplied by two to obtain the path length.

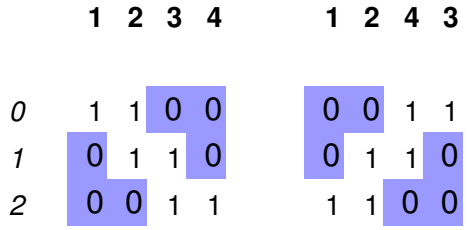

Figure A.3 The path $\pi(2,1)$

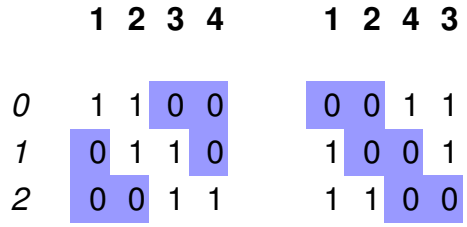

Figure A.4 The path $\pi(2,4)$

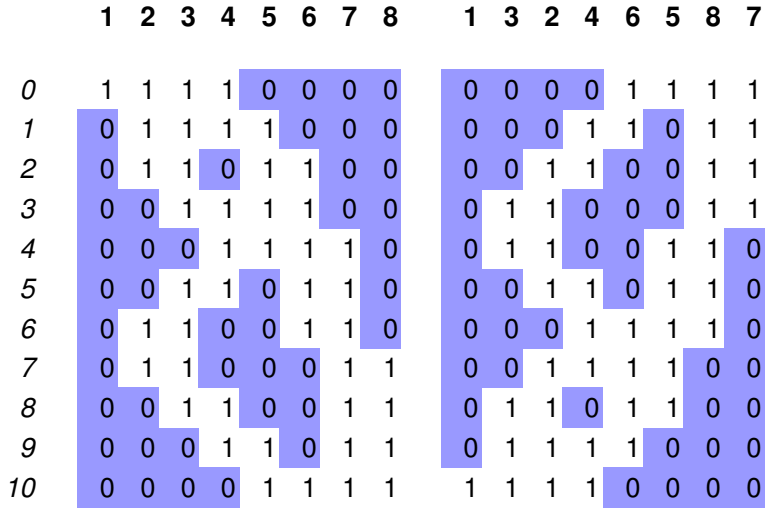

Figure A.5 The path $\pi(4,1)$

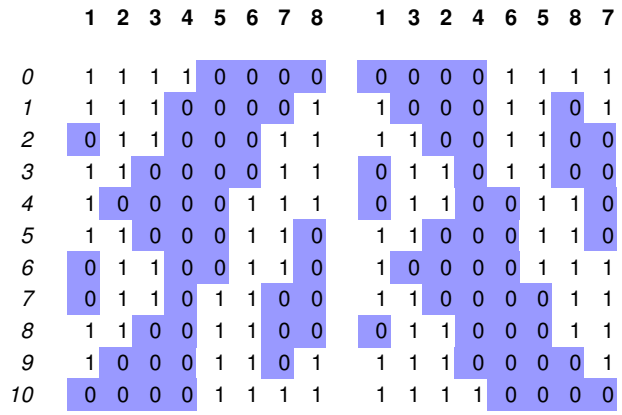

Figure A.6 The path $\pi(4,4)$

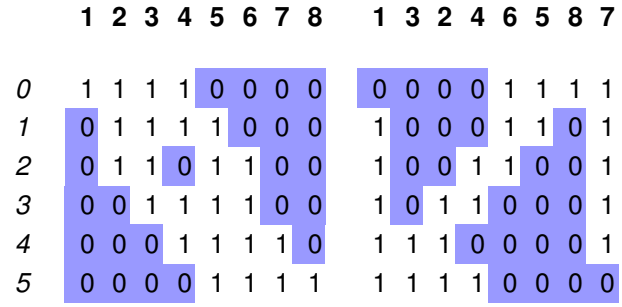

Figure A.7 The path $\pi(4,8)$

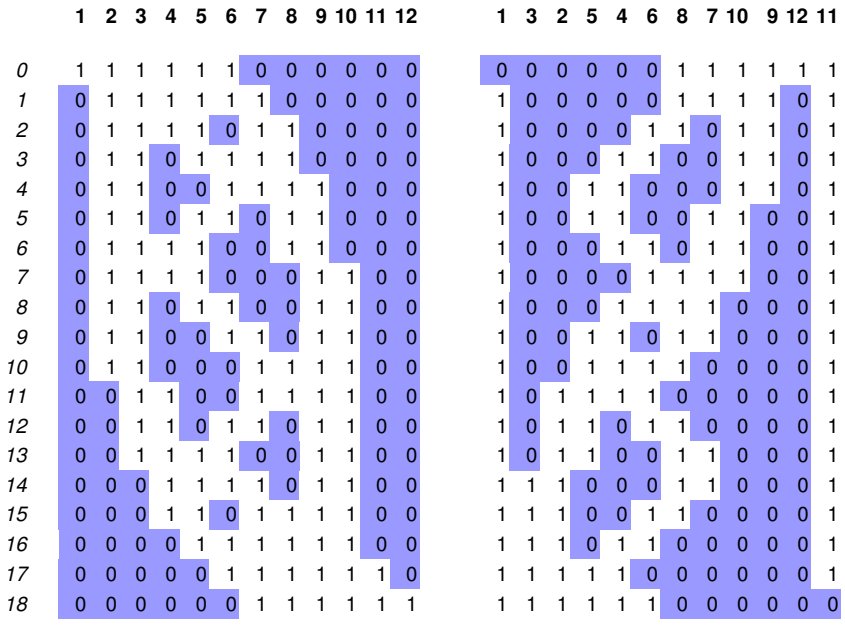

Figure A.8 The path $\pi(6,12)$

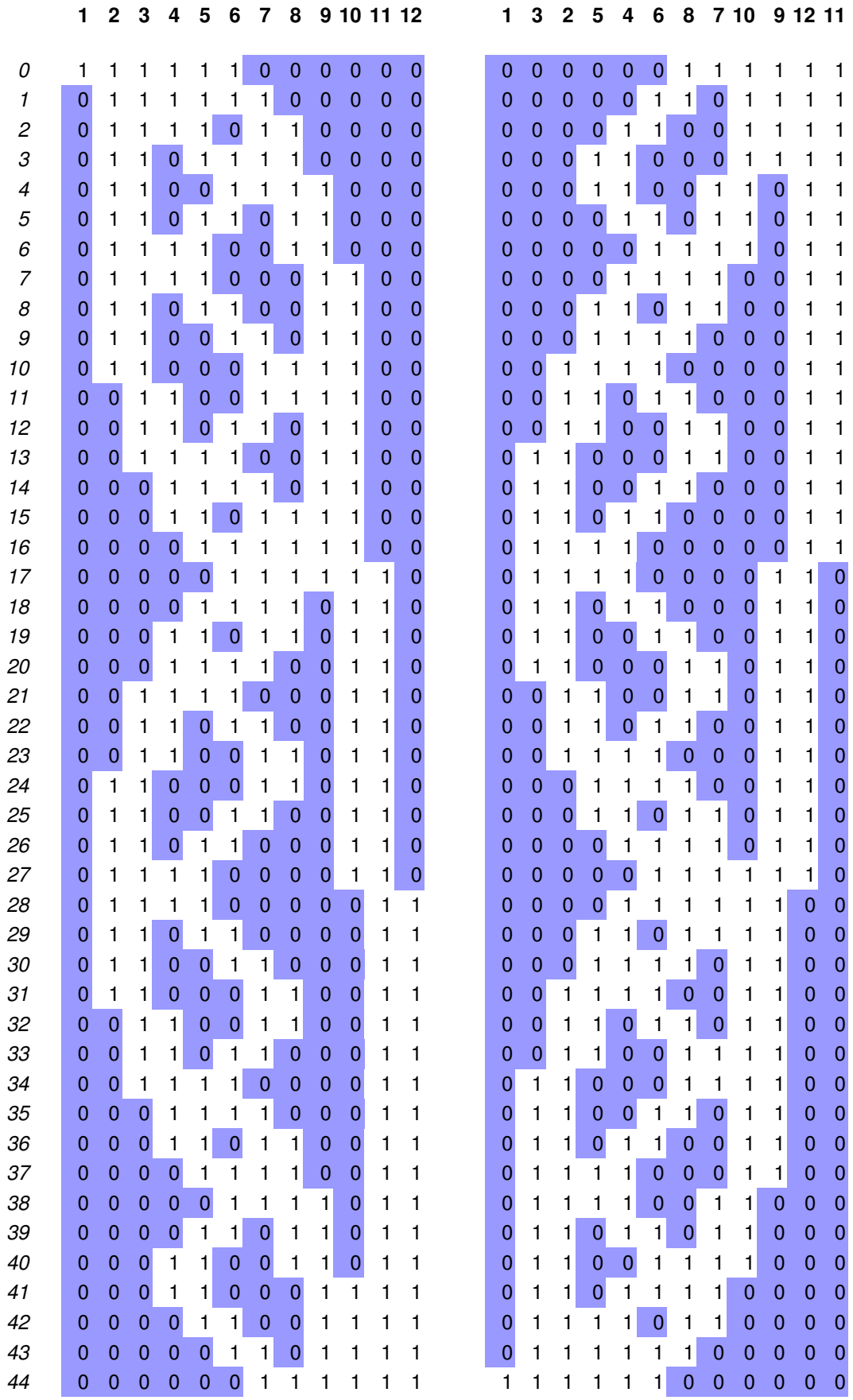

Figure A.9 The path $\pi(6,1)$

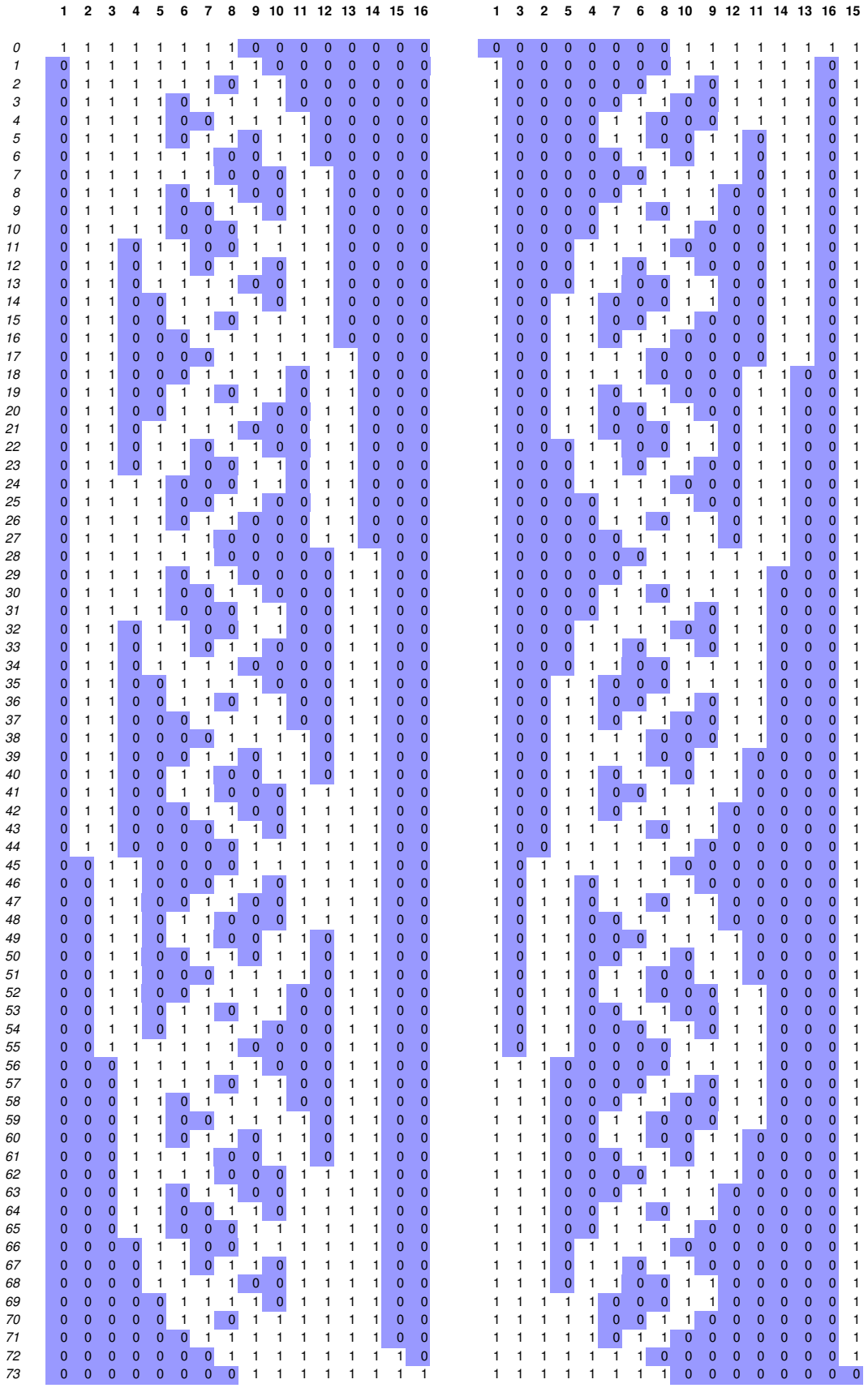

Figure A.10 The path $\pi(8, 16)$

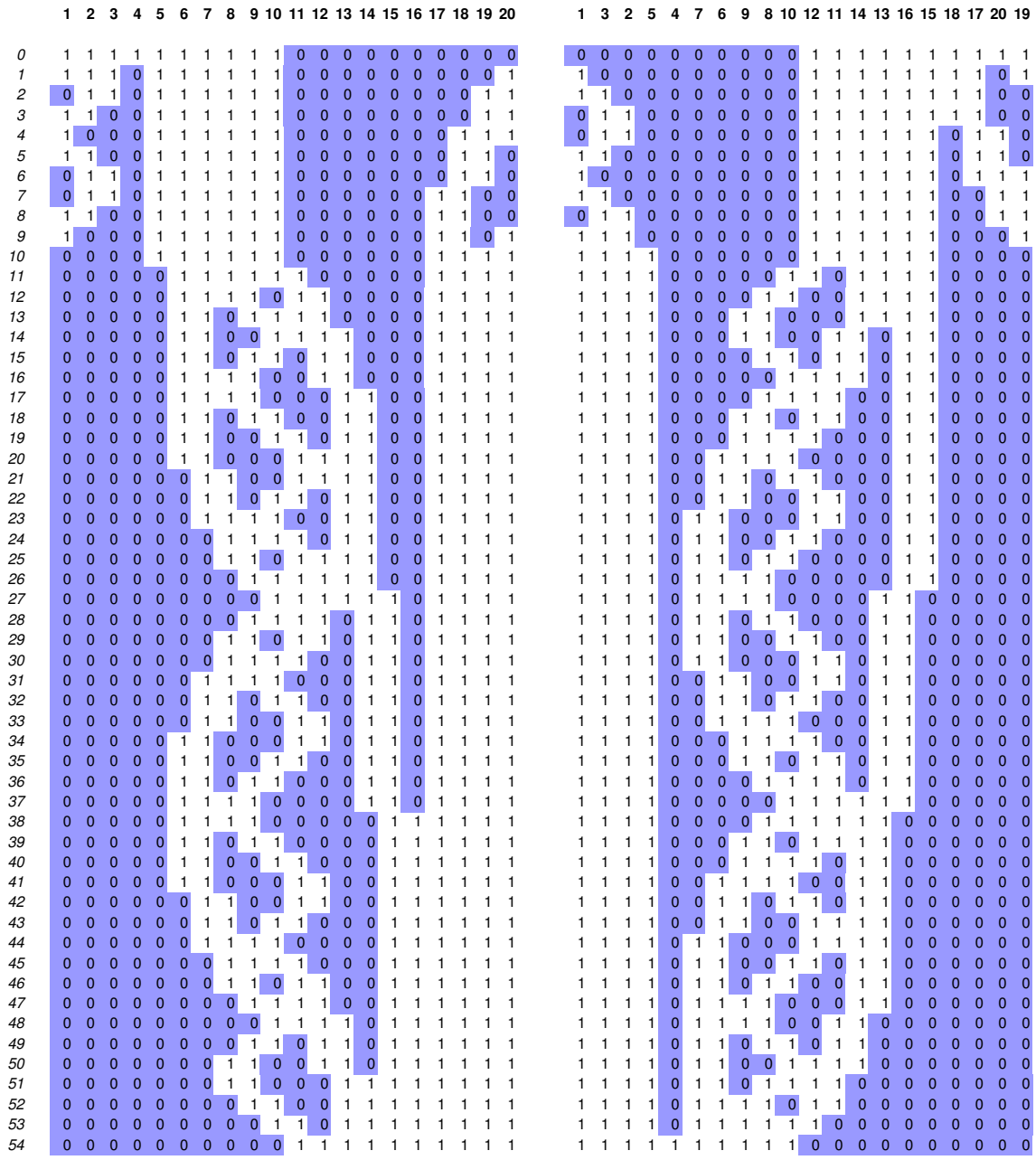

Figure A.11 The path $\pi(10,4)$

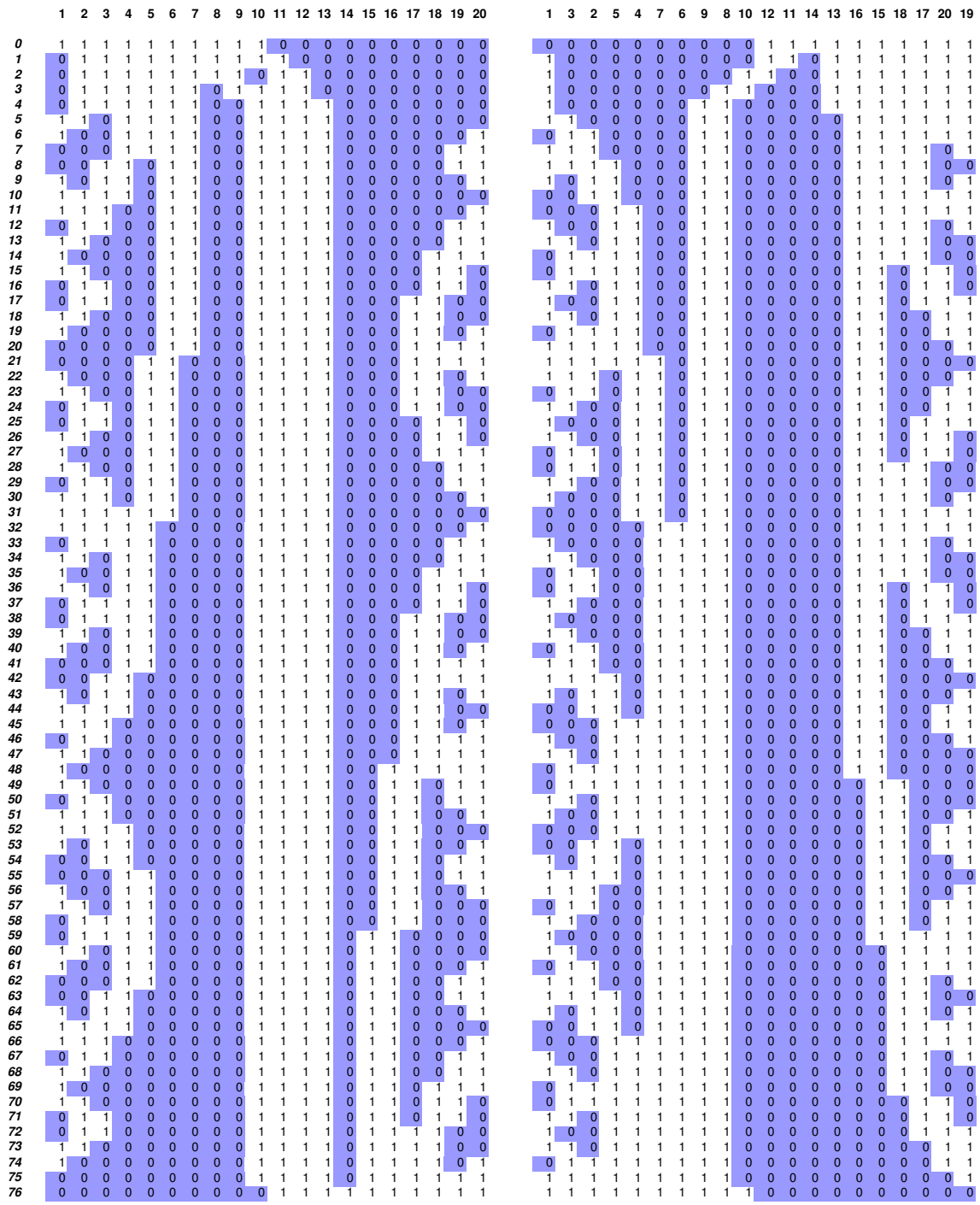

Figure A.12 The path $\pi(10, 14)$

A.2.3 Sample LH paths for $\Gamma(d, 2d)$

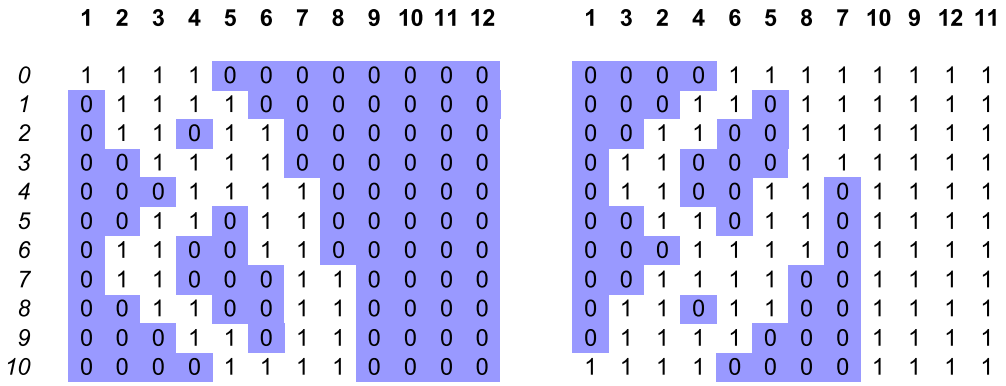

Figure A.13 The path $\rho(4,1)$

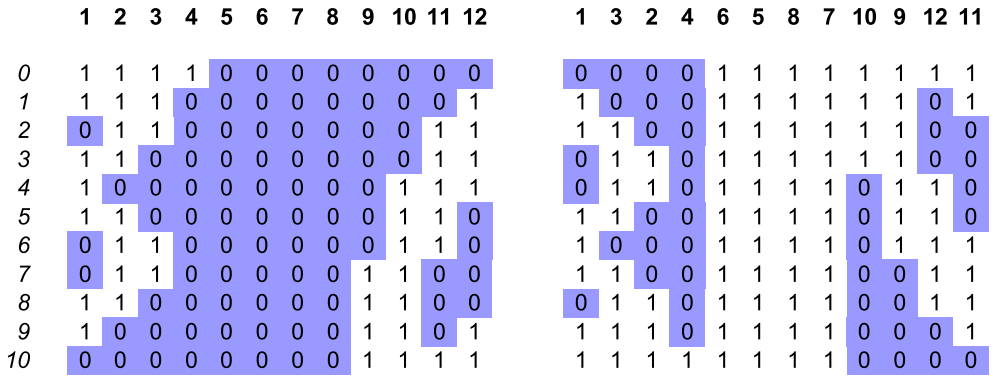

Figure A.14 The path $\rho(4,4)$

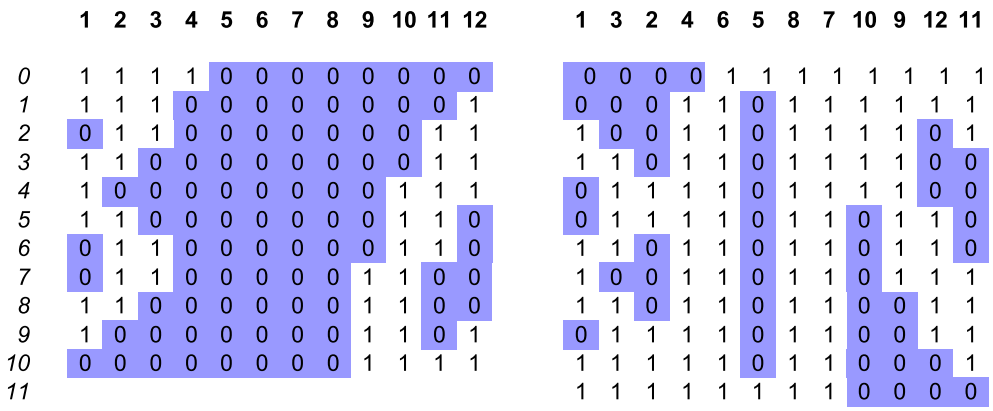

Figure A.15 The path $\rho(4,5)$

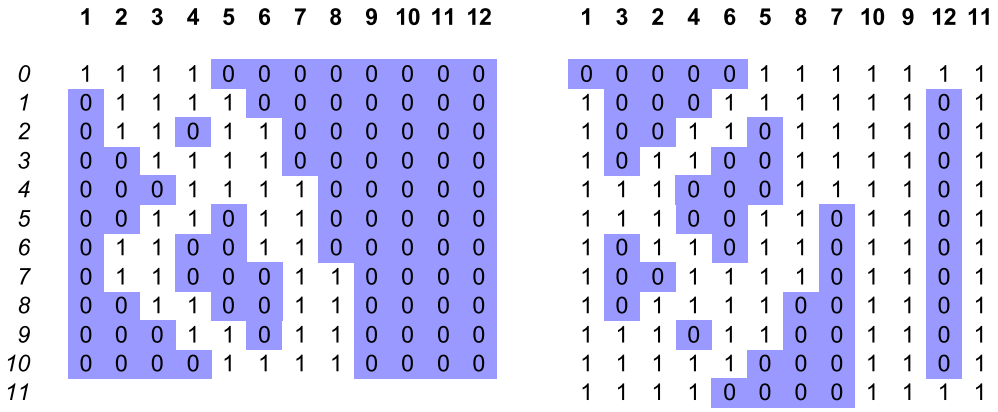

Figure A.16 The path $\rho(4, 12)$

A.2.4 Lemke path lengths for Morris's construction

Morris's construction was for both even and odd dimension; here we only give path lengths for even dimension.

label	dimension				
	2	4	6	8	10
1	2	6	16	40	98
2	2	4	8	18	42
3		4	8	18	42
4		6	8	12	22
5			8	12	22
6			16	18	22
7				18	22
8				40	42
9					42
10					98

Figure A.17 Lemke path lengths for Morris's construction in even dimension.

A.2.5 Sample Lemke paths for Morris's construction

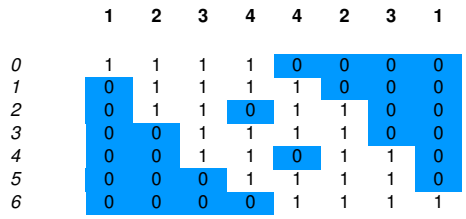

Figure A.18 The Lemke path of Morris in dimension 4 with missing label 1.

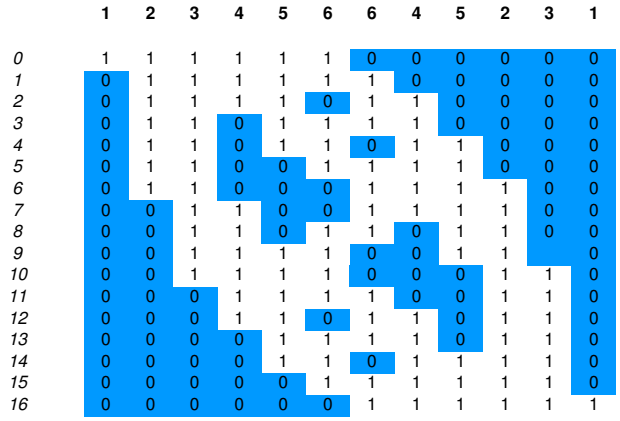

Figure A.19 The Lemke path of Morris in dimension 6 with missing label 1.

A.3 Number of equilibria of games $\Gamma(d,2d)$

The figure and formulas in this section are from von Stengel (1999), and are given here for convenience; see Lemma 3.14. In Figure A.20 we give the values of σ and $\tilde{\sigma}(l)$ for small values of $d = 2l$, which correspond to the number of equilibria in the games $\Gamma(d,2d)$.

d	0	2	4	6	8	10	12	14	16
$\sigma(l)$	1	3	13	63	321	1683	8989	48639	265729
$\tilde{\sigma}(l)$		3.4	13.8	65.5	330.4	1722.6	9165.3	49456.6	269636.8

Figure A.20 The first values of $\sigma(l)$ and $\tilde{\sigma}(l)$, where $\sigma(l)$ is the number of equilibria in a game $\Gamma(d,2d)$, $l = d/2$, and $\tilde{\sigma}(l)$ an asymptotic approximation of $\sigma(l)$ from von Stengel (1999).

Now we reproduce the corresponding equations for $\sigma(l)$ and $\tilde{\sigma}(l)$. The first is (3.6) from von Stengel (1999), and the second appears on page 566 of the same paper, without an equation number.

$$\sigma(l) = \sum_{k=0}^{l} \frac{(l+k)!}{k!\,k!\,(l-k)!} = \sum_{k=0}^{l} \binom{l+k}{k}\binom{l}{k},$$

$$\sigma(n) \sim \tilde{\sigma}(n) := \frac{1+\sqrt{2}}{2^{5/4}\sqrt{\pi}}\,\frac{(1+\sqrt{2})^{2n}}{\sqrt{n}}.$$

Index of Symbols

Symbol	Description	Page
$\mathbf{0}$	column vector of all 0s	19
$\mathbf{1}$	column vector of all 1s	19
$A(d)$	subpath of LH path $\pi(d,1)$	39
A, B	payoff matrices for player 1 and 2	17
$B(d)$	subpath of LH path $\pi(d,2d)$	35
C	payoff or cost matrix for player 1 in a symmetric game	18
$C(d)$	subpath of LH path $\pi(d,1)$	39
d	dimension (also see next entry)	18
d	covering vector for Lemke's algorithm (also see previous entry)	71
e_k	unit vector, with component k equal to 1, and all others 0	68
F	LCP map	79
$G(d,f)$	set of Gale even bitstrings of length f with d ones	31
$G(d)$	abbreviates $G(d,2d)$	33
$\Gamma(m,n)$	$m \times n$ double cyclic polytope game	32
$\Gamma_M(d)$	$d \times d$ symmetric game derived from Morris's construction	57
I	identity matrix	71
λ	permutation of labels	95
$l(k)$	facet labelling function	32
$l'(k)$	facet labelling function	32
$l''(k)$	facet labelling function	61
$L(d,k)$	length of LH path for $\Gamma(d,d)$ with missing label k	33
LCP (M,q)	linear complementarity problem with rhs q and matrix M	20
M	LCP matrix	20
M_i	ith column of matrix M	79
$M(d,k)$	length of LH path for $\Gamma(d,2d)$ with missing label k	50
n	LCP dimension	20

P''	dual cyclic polytope	30
P, Q	best response polytopes P and Q	22
$\pi(d,k)$	the LH path for $\Gamma(d,d)$ with missing label k	33
q	LCP rhs	20
$\rho(d,j)$	the LH path for $\Gamma(d,2d)$ with missing label j	51
$\mathrm{T}(d)$	triple imitation game in dimension d	60
S	best response polytope for symmetric game	18
u,v	bitstrings	32
w	LCP vector of slack variables	20
W	best response polyhedron for symmetric game defined by costs	68
z	LCP vector of variables	20

References

Adler, I., and N. Megiddo (1985), A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimension. *Journal of the Association for Computing Machinery* **32**, 871–895.

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.

Avis, D., and K. Fukuda (1992), A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete and Computational Geometry* **8**, 295–313.

Beame, P., S. Cook, J. Edmonds, R. Impagliazzo, and T. Pitassi (1998), The relative complexity of NP search problems. *Journal of Computer and System Sciences* **57**, 13–19.

Chen, X., and X. Deng (2005a), 3-NASH is PPAD-complete. *Electronic Colloquium on Computational Complexity*, Report TR05-134.

——— (2005b), Settling the complexity of 2-player Nash equilibrium. *Electronic Colloquium on Computational Complexity*, Report TR05-140.

Chvátal, V. (1983), Linear Programming. Freeman, New York.

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

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

Coxson, G. E. (1994), The P-matrix problem is co-NP-complete. *Mathematical Programming* **64**, 173–178.

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

Daskalakis, C., P. W. Goldberg, and C. H. Papadimitriou (2005), The complexity of Nash equilibria. *Electronic Colloquium on Computational Complexity*, Report TR05-115.

Daskalakis, C., and C. H. Papadimitriou (2005), Three-player games are hard. *Electronic Colloquium on Computational Complexity*, Report TR05-139.

Eaves, B. C. (1971), The Linear Complementarity Problem. *Management Science* **17**, 612–634.
Eaves, B. C., and H. Scarf (1976), The solution of systems of piecewise linear equations. *Mathematics of Operations Research* **1**, 1–27.

Gale, D. (1963), Neighborly and cyclic polytopes. In: *Convexity*, Proc. Symposia in Pure Mathematics, Vol. 7, ed. V. Klee, American Mathematical Society, Providence, Rhode Island, 225–232.

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

Garey, M. R., and D. S. Johnson (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA.

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

Goldberg, P. W., and C. H. Papadimitriou (2005), Reducibility among equilibrium problems. *Electronic Colloquium on Computational Complexity*, Report TR05-090.

Grünbaum, B. (2003), *Convex Polytopes, 2nd edition.* Springer, New York.

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

Goldfarb, D. (1983), Worst-case complexity of the shadow simplex algorithm. Report, Department of Industrial Engineering and Operations Research, Columbia University, New York.

——— (1994), On the complexity of the simplex method. In: *Advances in Optimization and Numerical Analysis (Oaxaca, 1992)*, Mathematics and its Applications, 275. Dordrecht: Kluwer, 25–38.

Hirsch, M., C. H. Papadimitriou, and S. Vavasis (1989), Exponential lower bounds for finding Brouwer fixpoints. *Journal of Complexity* **5**, 379–416.

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

Kalai, G., and D. J. Kleitman (1992), A quasi-polynomial bound for the diameter of graphs of polyhedra. *Bulletin of the American Mathematical Society* **26**, 315–316.

Keiding H. (1997), On the maximal number of Nash equilibria in an $n \times n$ bimatrix game. *Games and Economic Behavior* **21**, 148–160.

Klee, V., and P. Kleinschmidt (1987), The d-step conjecture and its relatives. *Mathematics of Operations Research* **12**, 718–755.

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.

Koller, D., N. Megiddo, and B. von Stengel (1994), Fast algorithms for finding randomized strategies in game trees. In: *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC)*, 750–759.

_____ (1996), Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior* **14**, 247–259.

Kuhn, H. W. (1961), An algorithm for equilibrium points in bimatrix games. In: *Proc. National Academy of Sciences of the U.S.A.* **47**, 1657–1662.

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.

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

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.

McLennan A., and I.-U. Park (1999), Generic 4×4 two person games have at most 15 Nash equilibria. *Games and Economic Behavior* **26**, 111–130.

McLennan, A., and R. Tourky (2004), From Lemke-Howson to Kakutani. Working paper, Department of Economics, University of Minnesota.

McMullen, P. (1970), The maximum number of faces of a convex polytope. *Mathematika* **17**, 179–184.

Megiddo, N. (1985), A note on the generality of the self-dual algorithm with various starting points. *Methods of Operations Research* **49**, 271–275.

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

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

Morris, W. D., Jr. (1994), Lemke paths on simple polytopes. *Mathematics of Operations Resesearch* **19**, 780–789.

Murty, K. G. (1978), Computational complexity of complementary pivot methods. *Mathematical Programming Study* **7***: Complementary and Fixed Point Problems*, 61–73.

_____ (1980), Computational complexity of parametric linear programming. *Mathematical Programming* **19**, 213–219.

_____ (1988), *Linear Complementarity, Linear and Nonlinear Programming.* Heldermann Verlag, Berlin.

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

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

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

_____ (2001), Algorithms, games, and the internet. In: *Proc. 33rd Annual ACM Symposium on the Theory of Computing (STOC)*, 749–753.

Porter, R. W., E. Nudelman, and Y. Shoham (2004), Simple search methods for finding a Nash equilibrium. In: *Proc. 19th National Conference on Artificial Intelligence (AAAI)*, 664–669.

Quint, T., and M. Shubik (1997), A theorem on the number of Nash equilibria in a bimatrix game. *International Journal of Game Theory* **26**, 353–359.

Rosenmüller, J. (1971), On a generalization of the Lemke–Howson algorithm to noncooperative N-person games. *SIAM Journal on Applied Mathematics* **21**, 73–79.

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

Savani, R., and B. von Stengel (2004), Exponentially many steps for finding a Nash equilibrium in a bimatrix game. *CDAM Research Report LSE-CDAM-2004-03*, London School of Economics. [Extended Abstract in *Proc. 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.]

_____ (2006), Hard-to-solve bimatrix games. *Econometrica* **74**, 397–429.

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

Smale, S. (1983), On the average number of steps of the simplex method of linear programming. *Mathematical Programming* **27**, 241–262.

Spielman, D. A., and S.-H. Teng (2004), *Journal of the ACM* **51**, 385–463.

Todd, M. J. (2001), The many facets of linear programming. *Mathematical Programming Series B* **91**, 417–436.

van der Heyden, L., (1980) A variable dimension algorithm for the linear complementarity problem. *Mathematical Programming* **19**, 328–346.

von Schemde, A. (2005), *Index and Stability in Bimatrix Games (A Geometric-Combinatorial Approach).* Series: Lecture Notes in Economics and Mathematical Systems, Vol. 560. Springer, New York.

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

_____ (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.

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.

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

Wilson, R. (1971), Computing equilibria of N-person games. *SIAM Journal on Applied Mathematics* **21**, 80–87.

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.

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