Complementary vertices and adjacency testing in polytopes

Benjamin A. Burton

Abstract

Our main theoretical result is that, if a simple polytope has a pair of complementary vertices (i.e., two vertices with no facets in common), then it has a second such pair. Using this result, we improve adjacency testing for vertices in both simple and non-simple polytopes: given a polytope in the standard form $\{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x} \geq 0\}$ and a list of its V vertices, we describe an $O(n)$ test to identify whether any two given vertices are adjacent. For simple polytopes this test is perfect; for non-simple polytopes it may be indeterminate, and instead acts as a filter to identify non-adjacent pairs. Our test requires an $O(n^2V + nV^2)$ precomputation, which is acceptable in settings such as all-pairs adjacency testing. These results improve upon the more general $O(nV)$ combinatorial and $O(n^3)$ algebraic adjacency tests from the literature.

AMS Classification Primary 52B05, 52B55

Keywords Polytopes, complementary vertices, disjoint facets, adjacent vertices, vertex enumeration, double description method

1 Introduction

Two vertices of a polytope are *complementary* if they do not belong to a common facet. Complementary vertices play an important role in the theory of polytopes; for instance, they provide the setting for the d-step conjecture [9, 10] (now recently disproved [15]), and in the dual setting of disjoint facets they play a role in the classification of compact hyperbolic Coxeter polytopes [6]. In game theory, Nash equilibria of bimatrix games are described by an analogous concept of complementary vertices in *pairs* of polytopes [8, 16].

Our first main contribution, presented in Section 2, relates to the minimal number of complementary vertex pairs. Many polytopes have no pairs of complementary vertices at all (for instance, any neighbourly polytope). However, we prove here that if a simple polytope P of dimension $d > 1$ has at least one pair of complementary vertices, then it must have at least *two* such pairs.

The proof involves the construction of paths through a graph whose nodes represent pairs of complementary or "almost complementary" vertices of P. In this sense it is reminiscent of the Lemke-Howson algorithm for constructing Nash equilibria in bimatrix games [12], although our proof operates in a less well-controlled setting. We discuss this relationship further in Section 4.

Our second main contribution, presented in Section 3, is algorithmic: we use our first theorem to build a fast adjacency test. Specifically, given a polytope in the standard form $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x} \geq 0\}$ with V vertices, we begin with an $O(n^2V + nV^2)$ time precomputation step, after which we can test any two vertices for adjacency in $O(n)$ time. If P is simple (which the algorithm can also identify) then this test always gives a precise response; otherwise it may be indeterminate but it can still assist in identifying non-adjacent pairs.

The key idea is, for each pair of vertices $u, v \in P$, to compute the join $u \vee v$; that is, the minimal face containing both u and v. If P is simple then our theorem on complementary vertices shows that $u \vee v = u' \vee v'$ for a second pair of vertices u', v'. Our algorithm then identifies such "duplicate" joins.

Although the precomputation is significant, if we are testing all $\binom{V}{2}$ pairs of vertices for adjacency then it does not increase the overall time complexity. Our $O(n)$ test then becomes extremely fast, outperforming the standard $O(nV)$ combinatorial and $O(n^3)$ algebraic tests from the literature [7]. Even in the non-simple setting, our test can be used as a fast pre-filter to identify non-adjacent pairs of vertices, before running the more expensive standard tests on those pairs that remain.

In Section 4 we discuss these performance issues further, as well as the application of these ideas to the key problem of polytope vertex enumeration.

All time complexities are measured using the arithmetic model of computation, where we treat each arithmetical operation as constant-time.

We briefly remind the reader of the necessary terminology. Following Ziegler [17], we insist that all polytopes be bounded. A *facet* of a d-dimensional polytope P is a $(d-1)$-dimensional face, and two vertices of P are *adjacent* if they are joined by an edge. P is *simple* if every vertex belongs to precisely d facets (i.e., every vertex figure is a $(d-1)$-simplex), and P is *simplicial* if every facet contains precisely d vertices (i.e., every facet is a $(d-1)$-simplex). As before, two vertices of P are *complementary* if they do not belong to a common facet; similarly, two facets of P are *disjoint* if they do not contain a common vertex.

2 Complementary Vertices

In this section we prove the main theoretical result of this paper:

Theorem 1. *Let P be a simple polytope of dimension $d > 1$. If P has a pair of complementary vertices, then P has at least two distinct pairs of complementary vertices.*

Note that these pairs of vertices are distinct, but need not be disjoint: they might be of the form $\{u, v\}$ and $\{u, w\}$ for some vertices $u, v, w \in P$.

The proof of Theorem 1 involves an auxiliary graph, which we now describe. To avoid confusion with vertices and edges of polytopes, we describe graphs in terms of *nodes* and *arcs*. We do not allow graphs to have loops or multiple edges.

Definition 2 (Auxiliary graph)**.** Let P be a simple polytope of dimension $d > 1$. We construct the *auxiliary graph* $\Gamma(P)$ as follows:

- The *nodes* of $\Gamma(P)$ are unordered pairs of vertices $\{u, v\}$ of P where u, v have at most one facet in common. We say the node $\{u, v\}$ is of *type A* if the vertices $u, v \in P$ are complementary (they have no facets in common), or of *type B* otherwise (they have precisely one facet in common).

- The *arcs* of $\Gamma(P)$ join nodes of the form $\{u,x\}$ and $\{u,y\}$, where x and y are adjacent vertices of P, and where no single facet of P contains all three vertices u,x,y.

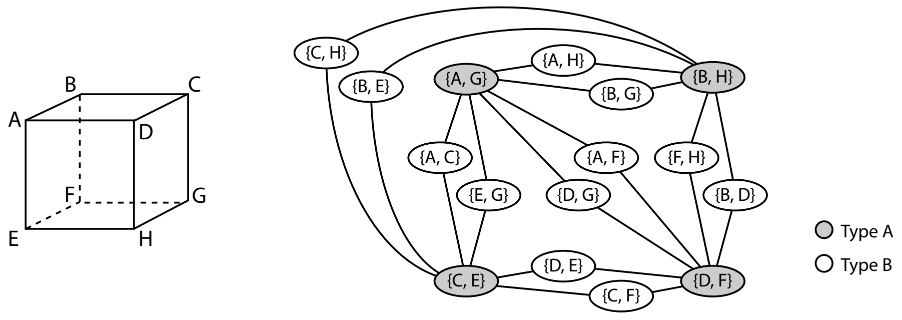

Figure 1: A polytope P and the corresponding auxiliary graph $\Gamma(P)$

Figure 1 illustrates this graph for the case where P is a cube. Informally, each arc of $\Gamma(P)$ modifies a node by "moving" one of its two vertices along an edge of P, so that if the two vertices lie on a common facet F then this movement is away from F. More formally, we can characterise the arcs of $\Gamma(P)$ as follows:

Lemma 3. *Let $\nu = \{u,v\}$ be a node of $\Gamma(P)$ as outlined above.*

(i) If ν is of type A, there are precisely $2d$ arcs meeting ν. These include d arcs that connect ν with $\{u,x\}$ for every vertex x adjacent to v in P, and d arcs that connect ν with $\{y,v\}$ for every vertex y adjacent to u in P.

(ii) If ν is of type B, there are precisely two arcs meeting ν. Let F be the unique facet of P containing both u and v. Then these two arcs join ν with $\{u,x\}$ and $\{y,v\}$, where x is the unique vertex of P adjacent to v for which $x \notin F$, and y is the unique vertex of P adjacent to u for which $y \notin F$.

Proof. We consider the type A and B cases in turn.

(i) Let $\nu = \{u,v\}$ be of type A, and let x be any vertex of P adjacent to v. Since ν is of type A, u and v have no facets in common. Since P is simple, at most one facet of P contains x but not v. Therefore u and x have at most one facet in common, and so $\{u,x\}$ is a node of $\Gamma(P)$.

Since $\{u,x\}$ is a node of $\Gamma(P)$ and no facet contains both u and v, it follows that $\nu = \{u,v\}$ and $\{u,x\}$ are joined by an arc. A similar argument applies to nodes $\{y,v\}$ where y is any vertex of P adjacent to u. Because P is simple there are precisely d vertices adjacent to v and d vertices adjacent to u, yielding precisely $2d$ arcs of this type.

(ii) Now let $\nu = \{u,v\}$ be of type B, and let F be the unique facet of P containing both u and v. If there is an arc from ν to any node of the form $\{u,x\}$, it is clear from Definition 2 that we must have x adjacent to v and $x \notin F$. Because P is simple, there is precisely one x with these properties.

We now show that an arc from ν to $\{u, x\}$ does indeed exist. Because P is simple, at most one facet of P contains x but not v. The vertex v in turn has only the facet F in common with u; since $x \notin F$ it follows that u and x have at most one facet in common. Therefore $\{u, x\}$ is a node of $\Gamma(P)$. Because $x \notin F$ there is no facet containing all of u, v and x, and so the arc from ν to $\{u, x\}$ exists.

A similar argument applies to the arc from ν to $\{y, v\}$, yielding precisely two arcs that meet ν as described in the lemma statement.

To finish, we note that every vertex belongs to $d > 1$ facets, and so every node $\{u, v\}$ of $\Gamma(P)$ has $u \neq v$. This ensures that we do not double-count arcs in our argument; that is, the $2d$ arcs in case (i) join ν to $2d$ distinct nodes of $\Gamma(P)$, and likewise the two arcs in case (ii) join ν to two distinct nodes of $\Gamma(P)$. □

Our strategy for proving Theorem 1 is to show that, if we follow any path from a type A node of $\Gamma(P)$, we must arrive at some *different* type A node; that is, we obtain a new pair of complementary vertices. The details are as follows.

Proof of Theorem 1. To establish Theorem 1, we must prove that if $\Gamma(P)$ contains at least one type A node, then it contains at least two type A nodes.

Let ν be a type A node, and let α be any arc meeting ν. Since every type B node has degree two (by Lemma 3), this arc α begins a well-defined path through $\Gamma(P)$ that passes through zero or more type B nodes in sequence, until either (a) it arrives at a new type A node ν', or (b) it returns to the original type A node ν and becomes a cycle (see Figure 2). Case (a) gives us the desired result; our task is to prove that case (b) is impossible.

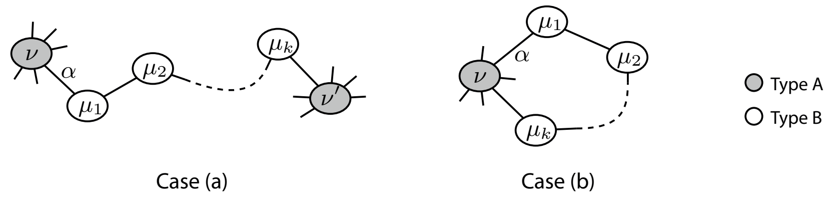

Figure 2: Following a path from the type A node ν

Suppose then that case (b) occurs, and there is a cycle that passes through nodes $\nu, \mu_1, \mu_2, \ldots, \mu_k, \nu$ in turn, where ν is type A and $\mu_1, \ldots, \mu_k$ are type B. Since $\Gamma(P)$ is a graph with no loops or multiple edges, we have $k \geq 2$. Since $\deg(\mu_i) = 2$, the nodes $\mu_1, \ldots, \mu_k$ are all distinct.

For any node $\eta = \{u, v\}$ of $\Gamma(G)$, let $\Phi(\eta)$ denote the set of all facets of P that contain either u or v. Because P is simple, Definition 2 gives $|\Phi(\eta)| = 2d$ if η is type A, or $|\Phi(\eta)| = 2d - 1$ if η is type B.

We now show that $\Phi(\mu_1) = \Phi(\mu_2) = \ldots = \Phi(\mu_k)$. Consider two adjacent nodes $\mu_i = \{x, y\}$ and $\mu_{i+1} = \{x, z\}$ along our cycle. Because y and z are adjacent vertices of the simple polytope P, there is only one facet F for which $y \in F$ but $z \notin F$. By Lemma 3, this facet F must be the unique facet containing both x and y. Therefore every facet that contains *either* x or y must also contain either x or z, and we have $\Phi(\mu_i) \subseteq \Phi(\mu_{i+1})$. Because $|\Phi(\mu_i)| = |\Phi(\mu_{i+1})| = 2d - 1$ we have $\Phi(\mu_i) = \Phi(\mu_{i+1})$, and by induction $\Phi(\mu_1) = \Phi(\mu_2) = \ldots = \Phi(\mu_k)$.

Consider now the type A node ν that begins and ends the cycle. Let $\nu = \{u, v\}$, and without loss of generality let $\mu_1 = \{u, x\}$ for some vertex $x \in P$. Because $|\Phi(\mu_1)| = 2d-1 < 2d = |\Phi(\nu)|$, there must be some facet F for which $F \in \Phi(\nu)$ but $F \notin \Phi(\mu_1)$; it follows then that $v \in F$ but $x \notin F$.

Moving to the other end of the cycle: since μ_k is adjacent to ν in $\Gamma(P)$, we have either $\mu_k = \{u, x'\}$ where x' is adjacent to v in P, or else $\mu_k = \{x', v\}$ where x' is adjacent to u in P. The second option is not possible because $v \in F \notin \Phi(\mu_1) = \Phi(\mu_k)$; therefore $\mu_k = \{u, x'\}$ for some x' adjacent to v.

We now know that both vertices x and x' are adjacent to v; moreover, since $F \notin \Phi(\mu_1) = \Phi(\mu_k)$ we have $x, x' \notin F$. However, P is a simple polytope with $v \in F$, and so there is only one vertex adjacent to v that is not in F. Therefore $x = x'$ and $\mu_1 = \mu_k$, contradicting the earlier observations that $k \geq 2$ and the nodes $\mu_1, \ldots, \mu_k$ are all distinct. This completes the proof of Theorem 1. □

Remark. The proof of Theorem 1 is algorithmic: given a simple polytope P of dimension $d > 1$ and a pair of complementary vertices $u, v \in P$, it gives an explicit algorithm for locating a second pair of complementary vertices.

In essence, we arbitrarily replace one of the vertices u with an adjacent vertex u', and then repeatedly adjust this pair of vertices according to Lemma 3 part (ii) until we once again reach a pair of complementary vertices. For each adjustment, Lemma 3 part (ii) gives two options (corresponding to the two arcs that meet a type B node); we always choose the option that leads us "forwards" to a new pair, and not "backwards" to the pair we had immediately before.

The proof above ensures that we will eventually reach a complementary pair of vertices again, and that these will not be the same as the original pair u, v.

Passing to the dual polytope, Theorem 1 gives us an immediate corollary:

Corollary 4. *Let P be a simplicial polytope of dimension $d > 1$. If P has a pair of disjoint facets, then P has at least two distinct pairs of disjoint facets.*

We finishing by showing that the "simple" and "simplicial" conditions are necessary in Theorem 1 and Corollary 4.

Observation 5. *The triangular bipyramid (Figure 3, left) is a non-simple polytope of dimension $d = 3$ with precisely one pair of complementary vertices (the apexes at the top and bottom, shaded in the diagram).*

Its dual is the triangular prism (Figure 3, right), which is a non-simplicial polytope of dimension $d = 3$ with precisely one pair of disjoint facets (the triangles at the top and bottom, shaded in the diagram).

Figure 3: A triangular bipyramid (left) and a triangular prism (right)

These constructions are easily generalised. For instance, we can build a non-simple bipyramid over a neighbourly polytope: the two apexes form the unique pair of complementary vertices, and all other pairs of vertices are adjacent. Felikson and Tumarkin provide further examples in the dual setting [6], involving non-simplicial Coxeter polytopes with precisely one pair of disjoint facets.

3 Adjacency Testing

In this section we prove our main algorithmic result, which uses Theorem 1 to identify simple polytopes and test for adjacent vertices. Throughout this section we work with polytopes of the form

$$P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x} \geq 0\}, \tag{1}$$

where A is some n-column matrix. This form is standard in mathematical programming, and appears in key applications of vertex enumeration [4, 5]. Note that the dimension of P is not immediately clear from (1), although $n - \operatorname{rank} A$ gives an upper bound; likewise, it is not immediately clear whether P is simple.

We recall some standard terminology from polytope theory: if F and G are faces of the polytope P, then the *join* $F \vee G$ is the unique smallest-dimensional face that contains both F and G as subfaces. For example, recall the cube from Figure 1, and consider the edges $\overline{AD}$ and $\overline{DC}$. Their meet $\overline{AD} \wedge \overline{DC}$ is the vertex D (where they intersect), and their join $\overline{AD} \vee \overline{DC}$ is the square facet $ABCD$ (the smallest face containing both edges).

Note that the join may be the entire polytope P (for instance, this happens if the faces F and G are complementary vertices).

Our main algorithmic result is the following:

Theorem 6. *Consider any polytope $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x} \geq 0\}$, and suppose we have a list of all vertices of P, with V vertices in total. Then, after a precomputation step that requires $O(n^2V + nV^2)$ time and $O(nV^2)$ space:*

- *we know immediately the dimension of P and whether or not P is simple;*
- *if P is simple then, given any two vertices $u, v \in P$, we can test whether or not u and v are adjacent in $O(n)$ time;*
- *if P is not simple then, given any two vertices $u, v \in P$, we may still be able to identify that u and v are non-adjacent in $O(n)$ time.*

We discuss the importance and implications of this result in Section 4; in the meantime, we devote the remainder of this section to proving Theorem 6.

Our overall strategy is to use our main theoretical result (Theorem 1) to characterise and identify adjacent vertices. As a first step, we describe complementary vertices in terms of joins:

Lemma 7. *Let u and v be distinct vertices of a polytope P. Then u and v are complementary vertices if and only if $u \vee v = P$.*

Proof. If $u \vee v \neq P$ then there is some facet F for which $u \vee v \subseteq F$; therefore $u, v \in F$, and u and v cannot be complementary.

On the other hand, if $u \vee v = P$ then there is no facet F for which $u, v \in F$ (otherwise we would have $u \vee v \subseteq F$). Therefore u and v are complementary. □

Using this lemma, we can now make a direct link between Theorem 1 and adjacency testing in polytopes:

Theorem 8. *Let P be a simple polytope, and let F be a face of P. Then F is an edge if and only if there is precisely one pair of distinct vertices $u, v \in P$ for which $F = u \vee v$.*

If P is any polytope (not necessarily simple), then the forward direction still holds: if F is an edge then there must be precisely one pair of distinct vertices $u, v \in P$ for which $F = u \vee v$.

Proof. The forward direction is straightforward: let F be an edge of any (simple or non-simple) polytope P, and let the endpoints of F be the vertices u and v. It is clear that $F = u \vee v$, and because F contains no other vertices it cannot be expressed as the join $x \vee y$ for any other pair of distinct vertices x, y.

We now prove the reverse direction in the case where P is a simple polytope. Let F be a face of P, and suppose that F is not an edge. If $\dim F < 1$ then F contains at most one vertex, and so there can be no pairs of distinct vertices u, v for which $F = u \vee v$.

Otherwise $\dim F > 1$; moreover, since P is a simple polytope then the face F is likewise simple when considered as a polytope of its own. Hence we can invoke Theorem 1 to finish the proof. If $F = u \vee v$ for distinct vertices u, v, then Lemma 7 shows that u, v are complementary in the "sub-polytope" F. By Theorem 1 there must be another pair of complementary vertices $\{u', v'\} \neq \{u, v\}$ in F, and by Lemma 7 we have $F = u' \vee v'$ as well. $\square$

To make the join operation accessible to algorithms, we describe faces of polytopes using *zero sets*. Fukuda and Prodon [7] define zero sets for points in P; here we extend this concept to arbitrary faces.

Definition 9 (Zero set)**.** Consider any polytope of the form $P = \{\mathbf{x} \in \mathbb{R}^n \,|\, A\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x} \geq 0\}$, and let F be any face of P. Then the *zero set* of F, denoted $Z(F)$, is the set of coordinate positions that take the value zero throughout F. That is, $Z(F) = \{i \,|\, x_i = 0 \text{ for all } \mathbf{x} \in F\} \subseteq \{1, 2, \ldots, n\}$.

Zero sets are efficient for computation: each can be stored and manipulated using a bitmask of size n, which for moderate problems ($n \leq 64$) involves just a single machine-native integer and fast bitwise CPU instructions. Joins, equality and subface testing all have natural representations using zero sets:

Lemma 10. *Let F and G be non-empty faces of the polytope $P = \{\mathbf{x} \in \mathbb{R}^n \,|\, A\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x} \geq 0\}$. Then:*

- *$F \subseteq G$ if and only if $Z(F) \supseteq Z(G)$;*
- *$F = G$ if and only if $Z(F) = Z(G)$;*
- *the join has zero set $Z(F \vee G) = Z(F) \cap Z(G)$.*

Proof. These are all consequences of the fact that every non-empty face $F \subseteq P$ is the intersection of P with all hyperplanes of the form $x_i = 0$ where $i \in Z(F)$. The details are as follows.

For each $i = 1, \ldots, n$, let H_i be the hyperplane $H_i = \{\mathbf{x} \in \mathbb{R}^n \,|\, x_i = 0\}$. It is a standard result that every non-empty face of P is the intersection of P with some set of hyperplanes H_i, and conversely that any intersection of P with some set of hyperplanes H_i yields a (possibly empty) face of P.

We claim that every non-empty face $F \subseteq P$ satisfies $F = P \cap \left(\bigcap_{i \in Z(F)} H_i\right)$; that is, F is obtained by intersecting P with every H_i for which $i \in Z(F)$.

To prove this claim, consider the face $F' = P \cap \left(\bigcap_{i \in Z(F)} H_i\right)$. By definition of $Z(F)$ it is clear that $F \subseteq F'$. If $F \neq F'$ then it follows that F is a strict subface of F', and so there must be some additional hyperplane H_j with $j \notin Z(F)$ for which $F \subseteq F' \cap H_j$. This in turn would imply that $j \in Z(F)$, a contradiction. Therefore $F = F'$.

We now prove the individual statements of the lemma. If $F \subseteq G$ then it is clear by definition of the zero set that $Z(F) \supseteq Z(G)$. Conversely, if $Z(F) \supseteq Z(G)$ then it follows from the claim above that F is the intersection of G with zero or more additional hyperplanes H_i, and so $F \subseteq G$. This proves the first statement, and the second statement of the lemma now follows immediately.

We finish with the third statement. Since $F, G \subseteq F \vee G$, it follows from the first statement that $Z(F \vee G) \subseteq Z(F) \cap Z(G)$. Suppose now that $Z(F \vee G) \neq Z(F) \cap Z(G)$; that is, there is some $i \in Z(F) \cap Z(G)$ for which $i \notin Z(F \vee G)$. Denote $F' = H_i \cap (F \vee G)$. Since $i \notin Z(F \vee G)$ we see that F' is a strict subface of $F \vee G$; moreover, since $i \in Z(F) \cap Z(G)$ we have $F, G \subseteq H_i$ and so $F, G \subseteq F'$. Therefore $F \vee G$ is not the smallest-dimensional face containing both F and G, contradicting the definition of join. □

We now define the main data structure that we build during our precomputation step in Theorem 6:

Definition 11 (Join map)**.** Consider any polytope of the form $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x} \geq 0\}$. The *join map* of P, denoted $\mathcal{J}_P$, is a map of the form $\mathcal{J}_P \colon 2^{\{1,\ldots,n\}} \to \mathbb{Z}_{\geq 0}$; that is, $\mathcal{J}_P$ maps subsets of $\{1, \ldots, n\}$ to non-negative integers. For each subset $S \subseteq \{1, \ldots, n\}$, we define the image $\mathcal{J}_P(S)$ to be the number of pairs of distinct vertices $\{u, v\} \in P$ for which $Z(u \vee v) = S$.

The following result reformulates Theorem 8 in terms of the join map, and follows immediately from Theorem 8 and Lemma 10.

Corollary 12. *Let P be a simple polytope, and let u, v be vertices of P. Then u and v are adjacent if and only if $\mathcal{J}_P(Z(u) \cap Z(v)) = 1$.*

If P is any polytope (not necessarily simple), then the forward direction still holds: if u and v are adjacent then we must have $\mathcal{J}_P(Z(u) \cap Z(v)) = 1$.

As a further corollary, the join map can be used to identify precisely whether or not a polytope is simple:

Corollary 13. *Let P be any polytope of dimension d. Then P is simple if and only if, for every vertex $u \in P$, there are precisely d other vertices $u' \in P$ for which $\mathcal{J}_P(Z(u) \cap Z(u')) = 1$.*

Proof. If P is simple then, for each vertex $u \in P$, there are precisely d other vertices $u' \in P$ adjacent to u. By Corollary 12 it follows that there are precisely d other vertices $u' \in P$ for which $\mathcal{J}_P(Z(u) \cap Z(u')) = 1$.

If P is non-simple then there is some vertex $u \in P$ that belongs to $> d$ edges, and so there are $> d$ other vertices $u' \in P$ adjacent to u. By Corollary 12, we have $\mathcal{J}_P(Z(u) \cap Z(u')) = 1$ for each of these adjacent vertices. □

We now show that the join map enjoys many of the properties required for the complexity bounds in Theorem 6:

Lemma 14. *Consider any polytope $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x} \geq 0\}$, and suppose we have a list of all vertices of P, with V vertices in total. Then we can construct the join map $\mathcal{J}_P$ in $O(nV^2)$ time and $O(nV^2)$ space. Once it has been constructed, we can compute $\mathcal{J}_P(S)$ for any set $S \subseteq \{1, \ldots, n\}$ in $O(n)$ time.*

Proof. We store the join map $\mathcal{J}_P$ using a *trie* (also known as a *prefix tree*). This is a binary tree of height n. Each leaf node (at depth n) represents some set $S \subseteq \{1, \ldots, n\}$, and stores the corresponding image $\mathcal{J}_P(S)$. Each intermediate node at depth $k < n$ supports two children: a "left child" beneath which every set S has $k+1 \notin S$, and a "right child" beneath which every set S has $k+1 \in S$.

We optimise our trie by only storing leaf nodes for sets S with $\mathcal{J}_P(S) \geq 1$, and only storing intermediate nodes that have such leaves beneath them. Figure 4 illustrates the complete trie for an example map with $n = 3$.

Join map:

S	$\{\}$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$
$\mathcal{J}_P(S)$	0	0	7	5	4	0	1	1

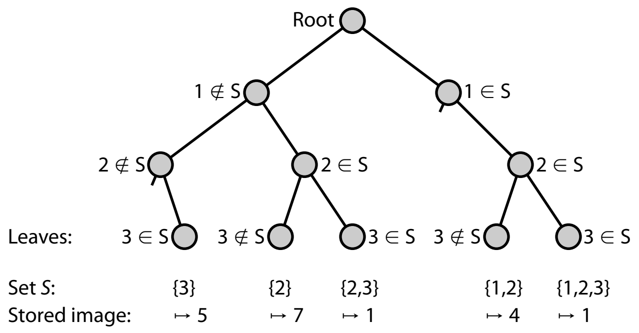

Figure 4: An example of a trie representing a join map for $n = 3$

Tries are fast to use: for any set $S \subseteq \{1, \ldots, n\}$, the operations of looking up the value of $\mathcal{J}_P(S)$, inserting a new value of $\mathcal{J}_P(S)$, or updating an existing value of $\mathcal{J}_P(S)$ are all $O(n)$, since each operation requires us to follow a single path from the root down to level n (possibly inserting new nodes as we go). For further information on tries in general, see a standard text such as [11].

It is clear now that we can construct $\mathcal{J}_P$ in $O(nV^2)$ time: for each of the $\binom{V}{2}$ pairs of vertices $\{u, v\}$, we construct the set $S = Z(u \vee v) = Z(u) \cap Z(v)$ in $O(n)$ time (just test which coordinate positions are zero in both u and v), and then perform an $O(n)$ lookup for S in the trie. If S is present then we increment $\mathcal{J}_P(S)$; otherwise we perform an $O(n)$ insertion to store the new value $\mathcal{J}_P(S) = 1$.

It is also clear that the trie consumes at most $O(nV^2)$ space: because the construction involves $O(V^2)$ insertions we have $O(V^2)$ leaf nodes, and since the trie has depth n this gives $O(nV^2)$ nodes in total.

Finally, for any set $S \subseteq \{1, \ldots, n\}$, computing $\mathcal{J}_P(S)$ involves a simple lookup operation in the trie, which again requires $O(n)$ time. $\square$

We are now ready to complete the proof of our main result, Theorem 6:

Proof of Theorem 6. The precomputation involves three stages:

(i) building the join map $\mathcal{J}_P$;

(ii) computing $\dim P$ using a rank computation;

(iii) iterating through the join map to determine whether P is simple.

By Lemma 14, stage (i) requires $O(nV^2)$ time and $O(nV^2)$ space.

To compute the dimension in stage (ii) we select an arbitrary vertex $v_0 \in P$, and build a $(V-1) \times n$ matrix M whose rows are of the form $v - v_0$ for each vertex $v \neq v_0$. It follows that $\dim P = \operatorname{rank} M$, and using standard Gaussian elimination we can compute this rank in $O(n^2V)$ time and $O(nV)$ space.

For stage (iii) we once again scan through all $\binom{V}{2}$ pairs of vertices u, v, compute $Z(u) \cap Z(v)$ for each in $O(n)$ time, and use an $O(n)$ lookup in $\mathcal{J}_P$ to test whether $\mathcal{J}_P(Z(u) \cap Z(v)) = 1$. We can thereby identify whether or not, for each vertex $u \in P$, there are precisely $\dim P$ other vertices $u' \in P$ for which $\mathcal{J}_P(Z(u) \cap Z(u')) = 1$; by Corollary 13 this tells us whether or not P is simple. The total running time for this stage is $O(nV^2)$.

Summing the three stages together, we find that our precomputation step requires a total of $O(n^2V + nV^2)$ time and $O(nV^2)$ space.

Once this precomputation is complete, it is clear from stages (ii) and (iii) that we know immediately the dimension of P and whether or not P is simple.

Consider now any two vertices $u, v \in P$. As before, we can compute the set $Z(u) \cap Z(v)$ in $O(n)$ time, and using Lemma 14 we can evaluate $\mathcal{J}_P(Z(u) \cap Z(v))$ in $O(n)$ time. Now Corollary 12 tells us what we need to know: if P is simple then u and v are adjacent if and only if $\mathcal{J}_P(Z(u) \cap Z(v)) = 1$, and if P is non-simple but $\mathcal{J}_P(Z(u) \cap Z(v)) \neq 1$ then we still identify that u and v are non-adjacent. $\square$

4 Discussion

As noted in the introduction, the proof of Theorem 1 is reminiscent of the Lemke-Howson algorithm for constructing Nash equilibria [12]. The Lemke-Howson algorithm operates on a pair of simple polytopes P and Q (best response polytopes for a bimatrix game), each with precisely $f = \dim P + \dim Q$ facets labelled $1, \ldots, f$, and locates vertices $u \in P$ and $v \in Q$ whose incident facet labels combine to give the full set $\{1, \ldots, f\}$. See [3, 12] for details.

The Lemke-Howson algorithm can also be framed in terms of paths through a graph Γ, where it can be shown that these paths yield a 1-factorisation of Γ. One then obtains the corollary that the number of fully-labelled vertex pairs is even; in particular, because there is always a "trivial" pair $(\mathbf{0}, \mathbf{0})$, there must be a second pair (which gives to a Nash equilibrium).

In this paper our setting is less well controlled. We work with a single polytope P, which means we must avoid transforming the pair of vertices $\{u, v\}$ into the identical pair $\{v, u\}$. Moreover, P may have arbitrarily many facets, which makes the arcs of $\Gamma(P)$ more difficult to categorise. In particular, we do not obtain any such 1-factorisation or parity results. To illustrate this, Figure 5 shows that both parities can occur: the cube (on the left) has four complementary pairs of vertices, whereas the cube with one truncated vertex (on the right) has nine.

Figure 5: Simple polytopes with (i) four, and (ii) nine complementary vertex pairs

Moving to Theorem 6: our $O(n)$ time adjacency test is the fastest we can hope for, since vertices require $\Omega(n)$ space to store (a consequence of the fact that there may be exponentially many vertices [13]). In contrast, standard approaches to adjacency testing use either an $O(nV)$ "combinatorial test" (where we search for a third vertex $w \in u \vee v$), or an $O(n^3)$ "algebraic test" (where we use a rank computation to determine $\dim(u \vee v)$). See [7] for details of these standard tests.

If we are testing adjacency for all pairs of vertices, our total running time including precomputation comes to $O(n^2V + nV^2)$, as opposed to $O(nV^3)$ or $O(n^3V^2)$ for the combinatorial and algebraic tests respectively. This is a significant improvement, given that V may be exponential in n.

The main drawback of our test is that it only guarantees conclusive results for simple polytopes. Nevertheless, it remains useful in the general case: it can detect when a polytope is simple, even if this is not clear from the initial representation $\{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x} \geq 0\}$, and even in the non-simple setting it gives a fast filter for eliminating non-adjacent pairs of vertices.

One application of all-pairs adjacency testing is in studying the *graph* of a polytope; that is, the graph formed from its vertices and edges. Another key application is in the *vertex enumeration* problem: given a polytope in the form $\{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x} \geq 0\}$, identify all of its vertices. This is a difficult problem, and it is still unknown whether there exists an algorithm polynomial in the combined input and output size.

The two best-known algorithms for vertex enumeration are reverse search [1, 2] and the double description method [7, 14], each with their own advantages and drawbacks. The double description method, which features in several application areas such as multiobjective optimisation [4] and low-dimensional topology [5], inductively constructs a sequence of polytopes by adding constraints one at a time. Its major bottleneck is in identifying pairs of adjacent vertices in each intermediate polytope, and in this setting our fast all-pairs adjacency test can be of significant practical use.

Acknowledgements

The author is grateful to the Australian Research Council for their support under the Discovery Projects funding scheme (project DP1094516).

References

[1] David Avis, *A revised implementation of the reverse search vertex enumeration algorithm*, Polytopes—Combinatorics and Computation (Oberwolfach, 1997), DMV Sem., vol. 29, Birkhäuser, Basel, 2000, pp. 177–198.

[2] David Avis and Komei Fukuda, *A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra*, Discrete Comput. Geom. **8** (1992), no. 3, 295–313.

[3] David Avis, Gabriel D. Rosenberg, Rahul Savani, and Bernhard von Stengel, *Enumeration of Nash equilibria for two-player games*, Econom. Theory **42** (2010), no. 1, 9–37.

[4] Harold P. Benson, *An outer approximation algorithm for generating all efficient extreme points in the outcome set of a multiple objective linear programming problem*, J. Global Optim. **13** (1998), no. 1, 1–24.

[5] Benjamin A. Burton, *Optimizing the double description method for normal surface enumeration*, Math. Comp. **79** (2010), no. 269, 453–484.

[6] Anna Felikson and Pavel Tumarkin, *Coxeter polytopes with a unique pair of non-intersecting facets*, J. Combin. Theory Ser. A **116** (2009), no. 4, 875–902.

[7] Komei Fukuda and Alain Prodon, *Double description method revisited*, Combinatorics and Computer Science (Brest, 1995), Lecture Notes in Comput. Sci., vol. 1120, Springer, Berlin, 1996, pp. 91–111.

[8] M. J. M. Jansen, *Maximal Nash subsets for bimatrix games*, Naval Res. Logist. Quart. **28** (1981), no. 1, 147–152.

[9] Edward D. Kim and Francisco Santos, *An update on the Hirsch conjecture*, Jahresber. Dtsch. Math.-Ver. **112** (2010), no. 2, 73–98.

[10] Victor Klee and David W. Walkup, *The d-step conjecture for polyhedra of dimension* $d < 6$, Acta Math. **117** (1967), 53–78.

[11] Donald E. Knuth, *The art of computer programming, Vol. 3: Sorting and searching*, 2nd ed., Addison-Wesley, Reading, MA, 1998.

[12] C. E. Lemke and Jr. J. T. Howson, *Equilibrium points of bimatrix games*, J. Soc. Indust. Appl. Math. **12** (1964), no. 2, 413–423.

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

[14] T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall, *The double description method*, Contributions to the Theory of Games, Vol. II (H. W. Kuhn and A. W. Tucker, eds.), Annals of Mathematics Studies, no. 28, Princeton University Press, Princeton, NJ, 1953, pp. 51–73.

[15] Francisco Santos, *A counterexample to the Hirsch conjecture*, To appear in Ann. of Math. (2), `arXiv:1006.2814`, 2010.

[16] H.-M. Winkels, *An algorithm to determine all equilibrium points of a bimatrix game*, Game Theory and Related Topics (Proc. Sem., Bonn and Hagen, 1978), North-Holland, Amsterdam, 1979, pp. 137–148.

[17] Günter M. Ziegler, *Lectures on polytopes*, Graduate Texts in Mathematics, no. 152, Springer-Verlag, New York, 1995.

Benjamin A. Burton
School of Mathematics and Physics, The University of Queensland
Brisbane QLD 4072, Australia
(bab@maths.uq.edu.au)