Complementary vertices and adjacency testing in polytopes

Benjamin Burton

Computational Geometry & Topology Group
The University of Queensland, Australia

COCOON 2012

Overview

- A theorem:

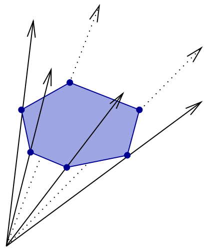
  For a simple polytope of dimension > 1,
  some pair of complementary vertices
  $\Longrightarrow$ at least two pairs of complementary vertices

- An algorithm:

  All-pairs adjacency testing for vertices of a polytope in $\mathbb{R}^n$

Motivations:

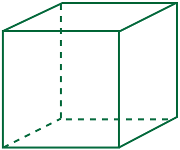
- Computational geometry (vertex enumeration)
- Computational topology (unknot recognition).



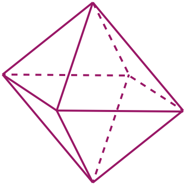
Definitions

Polytopes are always bounded.

The facets of a d-dimensional polytope are its $(d-1)$-dimensional faces.



Simple



Simplicial

In a simple polytope of dimension d, every vertex belongs to precisely d facets.

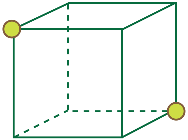
In a simplicial polytope, every facet is a simplex (and so contains precisely d vertices).

A theorem

Complementary vertices do not lie on a common facet.

Theorem

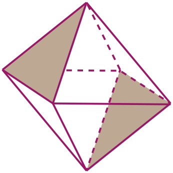
In a simple polytope of dimension > 1,
if there is a pair of complementary vertices
$\Longrightarrow$ *there are at least two such pairs.*



Disjoint facets do not contain any common vertices.

Corollary (Dual statement)

In a simplicial polytope of dimension > 1,
if there is a pair of disjoint facets
$\Longrightarrow$ *there are at least two such pairs.*

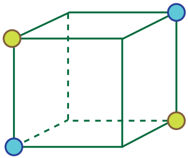


A theorem

Complementary vertices do not lie on a common facet.

Theorem

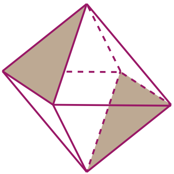
In a simple polytope of dimension > 1,
if there is a pair of complementary vertices
$\Longrightarrow$ *there are at least two such pairs.*



Disjoint facets do not contain any common vertices.

Corollary (Dual statement)

In a simplicial polytope of dimension > 1,
if there is a pair of disjoint facets
$\Longrightarrow$ *there are at least two such pairs.*

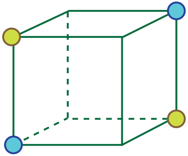


A theorem

Complementary vertices do not lie on a common facet.

Theorem

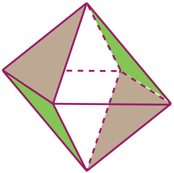
In a simple polytope of dimension > 1,
if there is a pair of complementary vertices
$\Longrightarrow$ *there are at least two such pairs.*



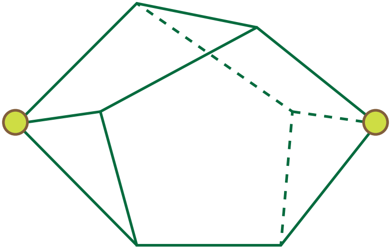
Disjoint facets do not contain any common vertices.

Corollary (Dual statement)

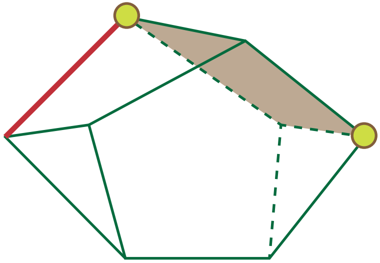
In a simplicial polytope of dimension > 1,
if there is a pair of disjoint facets
$\Longrightarrow$ *there are at least two such pairs.*



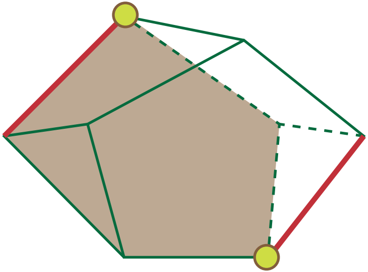
The proof



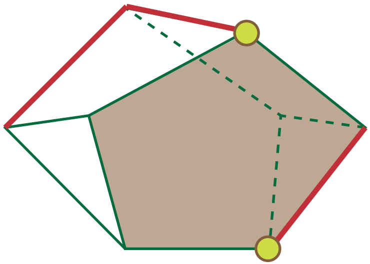
The proof



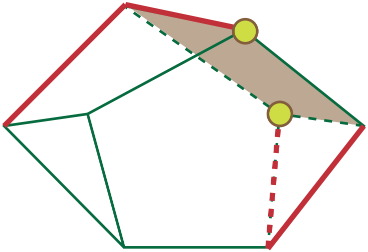
The proof



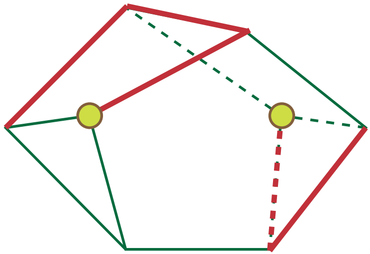
The proof



The proof



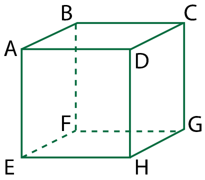
The proof

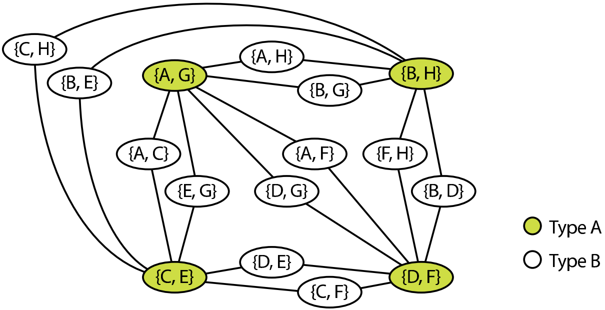


The details

Build an auxiliary graph:

- Type A nodes are pairs of complementary vertices;
- Type B nodes are pairs of vertices on exactly one common facet;
- Arcs join $\{u, x\} \leftrightarrow \{u, y\}$, where x, y are adjacent vertices, and no facet contains all of u, x, y.



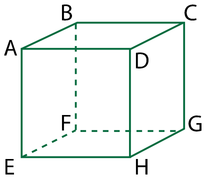


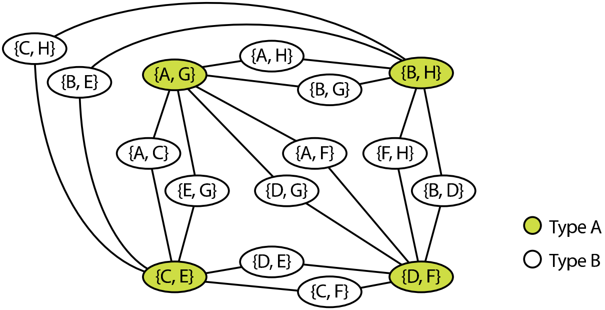
The details

Observations:

- Each type A node has $2d$ outgoing arcs;
- Each type B node has two outgoing arcs;
- There are no loops or multiple edges.

Our strategy is to follow a path from a type A vertex, and hope to arrive at a different type A vertex.



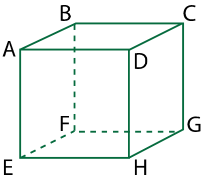


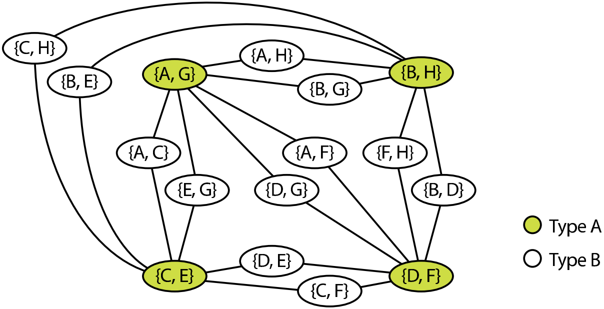
The details

More observations:

- On any path passing through only type B nodes, every vertex pair $\{x, y\}$ meets the same $2d - 1$ facets.
- All $2d$ outgoing arcs from any type A node lead to vertex pairs $\{x, y\}$ that meet different sets of facets.

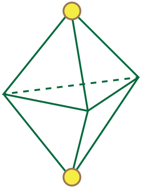
$\Longrightarrow$ A path from a type A vertex cannot return to the same vertex.

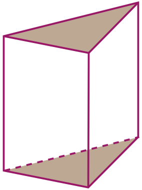




Observations

- The “simple” condition is necessary.





- The proof is reminiscent of the Lemke-Howson algorithm for constructing Nash equilibria in game theory.
  - Lemke-Howson operates on a tightly-structured pair of “best response polytopes”, and also yields a parity theorem (the number of Nash equilibria is odd).
  - Our setting is much less controlled, and simple examples show that such a parity result is not possible.

An algorithm

We work with a polytope $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x} \geq 0\}$.

This is a standard presentation in mathematical programming.

- Caveat: The matrix A does not immediately tell us the dimension, or the facets, or whether P is simple.

All-pairs adjacency testing

Input: A polytope P described by n, A and $\mathbf{b}$ as above, plus the list of all vertices of P.

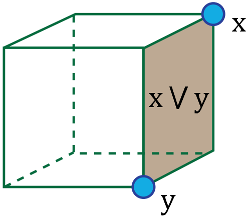
Output: The set of all adjacent pairs of vertices of P.

Applications and motivations:

- Studying the graph of a polytope
- The double description method for vertex enumeration (used in multiobjective optimisation and computational topology)

The setting

For vertices x, y of P, the join $x \vee y$ is the smallest face containing both x and y.



Observation: x and y are adjacent if and only if $x \vee y$ is an edge.

Well-known tests for whether vertices x, y are adjacent:

- a $O(nV)$ combinatorial test (search for a third vertex on $x \vee y$);
- a $O(n^3)$ algebraic test (compute the dimension of $x \vee y$).

Our results

An $O(n)$ test for simple polytopes:

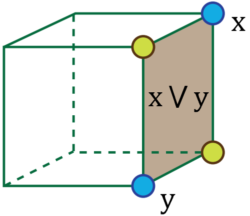
- Requires $O(n^2 V + nV^2)$ precomputation;
- Also identifies whether P is simple.

For all-pairs adjacency testing, this outperforms the $O(nV^3)$ and $O(n^3 V^2)$ combinatorial and algebraic methods.

For non-simple polytopes, we still obtain a fast filter for eliminating non-adjacent pairs.

How it works

If x and y are non-adjacent and P is simple, then by our theorem there is some other pair of vertices x', y' with $x \vee y = x' \vee y'$.



- Precomputation:
  Compute $x \vee y$ for all vertex pairs $\{x, y\}$.
- Fast adjacency test:
  Given vertices x, y, look up whether $x \vee y$ was computed more than once.

Implementation details

Store vertices and faces using zero sets:
bitmasks of length n indicating which coordinates are set to zero.

Store joins $x \vee y$ using a trie for $O(n)$ insertion and lookup.

Precomputation takes $O(n^2 V + nV^2)$ time.
The $O(n^2 V)$ term comes from testing whether P is simple.

Implementation details

Store vertices and faces using zero sets:
bitmasks of length n indicating which coordinates are set to zero.

Store joins $x \vee y$ using a trie for $O(n)$ insertion and lookup.

Precomputation takes $O(n^2 V + nV^2)$ time.
The $O(n^2 V)$ term comes from testing whether P is simple.

Questions?