On the complexity of the Euler characteristic of abstract simplicial complexes. Algorithms to compute it

Bjarke Hammersholt Roune Eduardo Sáenz-de-Cabezón

Datalogisk Institut
Aarhus University
bjarke@cs.au.dk
Departamento de Matemáticas y Computación
Universidad de La Rioja
eduardo.saenz-de-cabezon@unirioja.es

November - 11th - 2010

AARHUS UNIVERSITY

Euler characteristic

- Important invariant of a topological space.
- Appears in very different contexts: category theory, algebraic and differential geometry, ...
- In combinatorics: related to Möbius function, inclusion-exclusion, ...

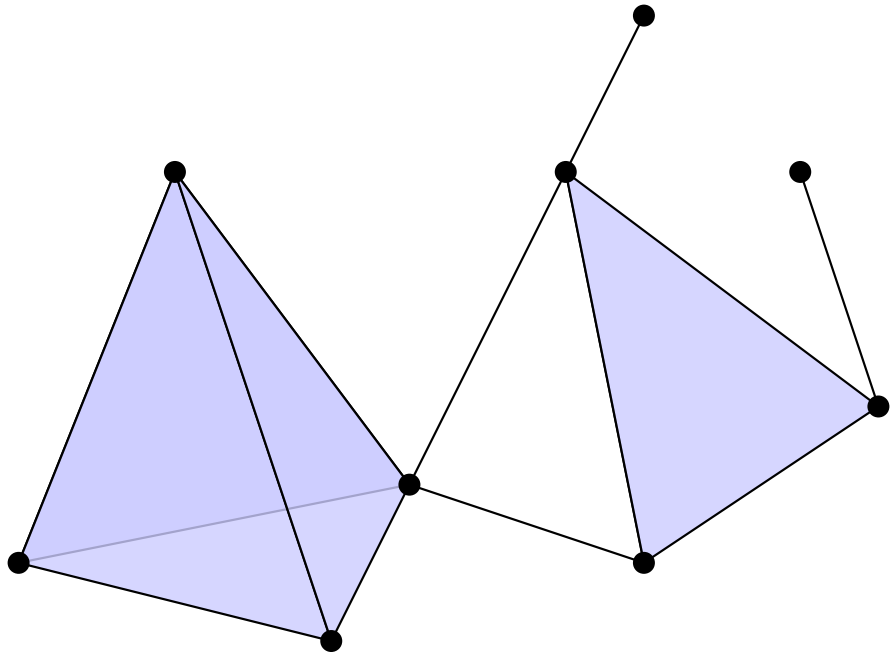
Euler characteristic

- Important invariant of a topological space.
- Appears in very different contexts: category theory, algebraic and differential geometry, ...
- In combinatorics: related to Möbius function, inclusion-exclusion, ...

Euler characteristic

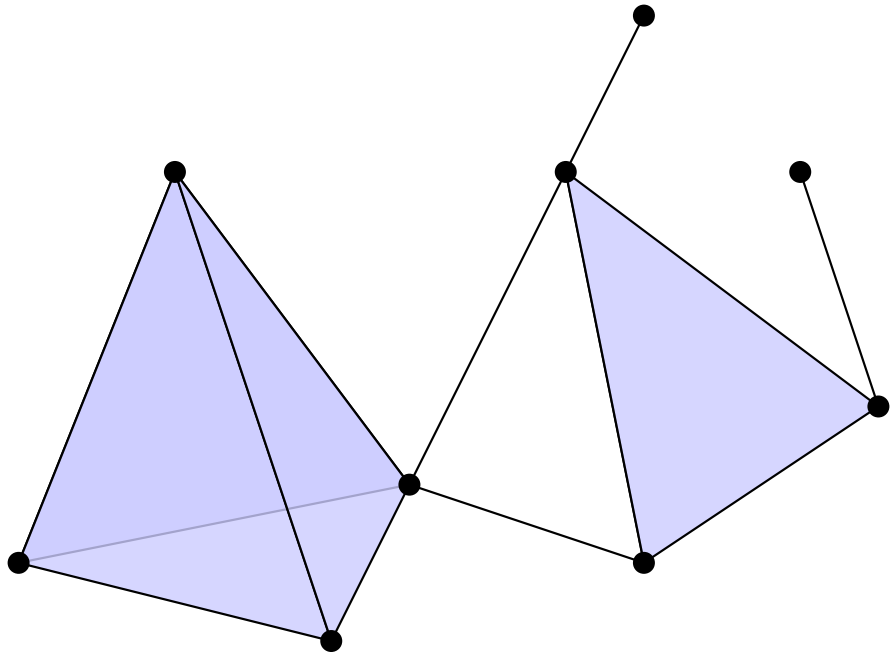
- Important invariant of a topological space.
- Appears in very different contexts: category theory, algebraic and differential geometry, ...
- In combinatorics: related to Möbius function, inclusion-exclusion, ...

Euler characteristic of an abstract simplicial complex



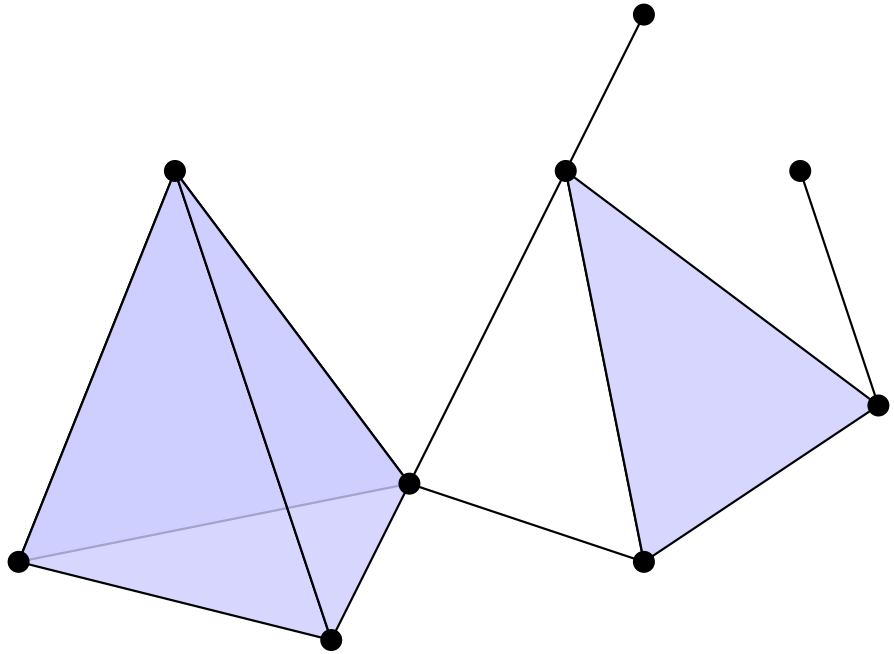
$$\chi(\Delta) = f_0 - f_1 + f_2 - f_3 + \cdots$$

Euler characteristic of an abstract simplicial complex



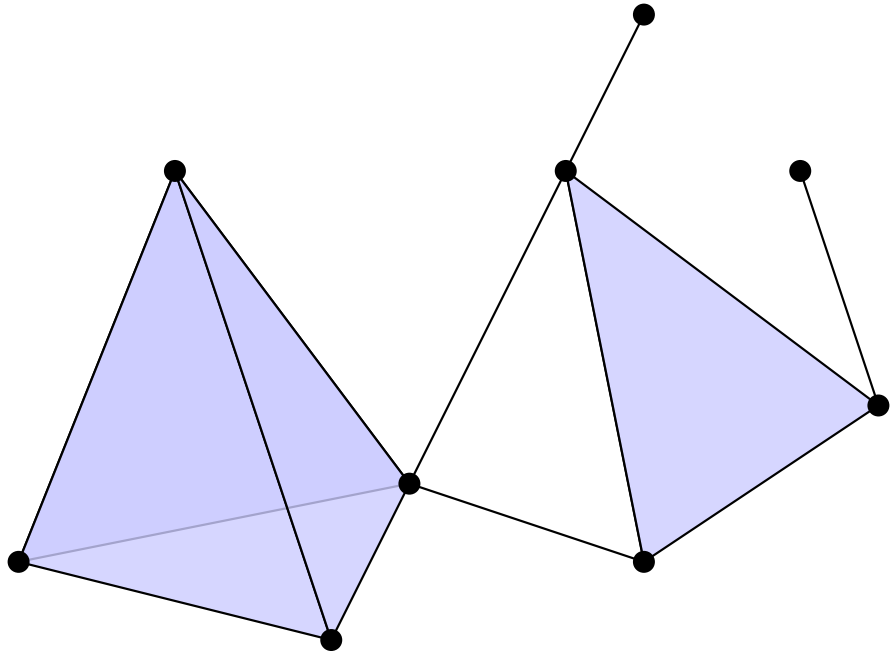
$$\chi(\Delta) = f_0 - f_1 + f_2 - f_3 + \cdots$$
$$= 9 - 13 + 5 = 1$$

Euler characteristic of an abstract simplicial complex



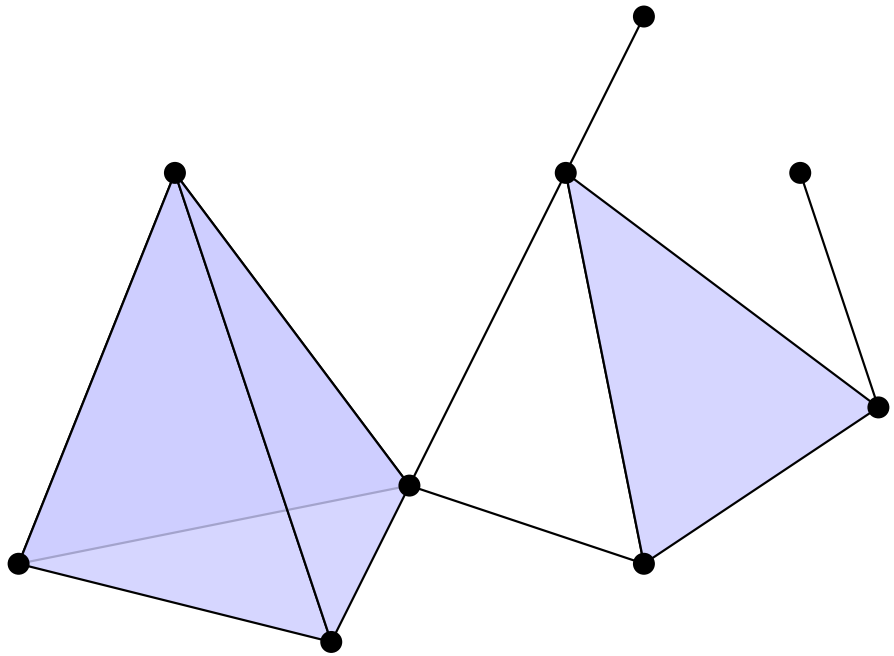
$$\chi(\Delta) = \beta_0 - \beta_1 + \beta_2 - \beta_3 + \cdots$$

Euler characteristic of an abstract simplicial complex



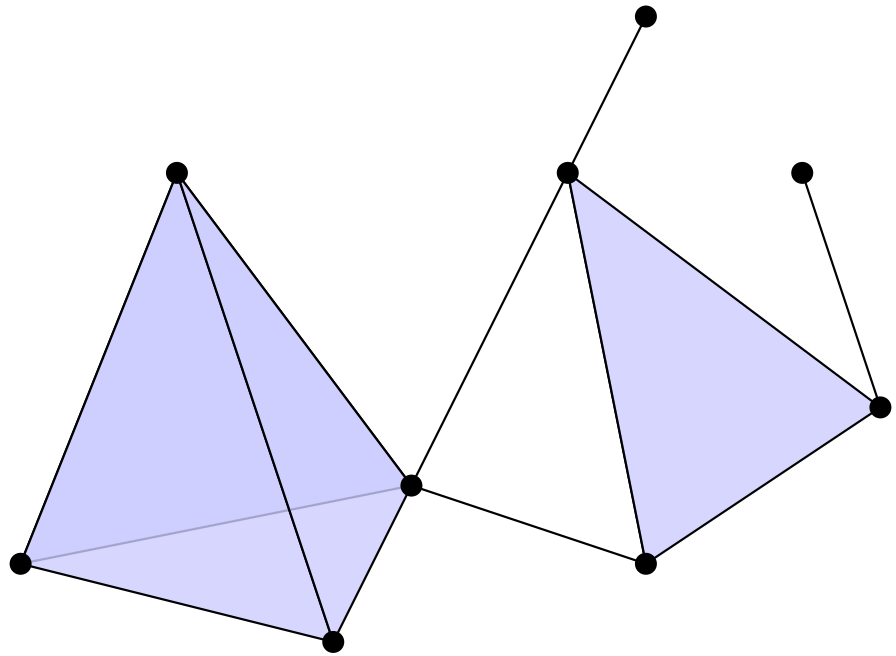
$$\chi(\Delta) = \beta_0 - \beta_1 + \beta_2 - \beta_3 + \cdots$$
$$= 1 - 1 + 1 = 1$$

Reduced Euler characteristic of an abstract simplicial complex



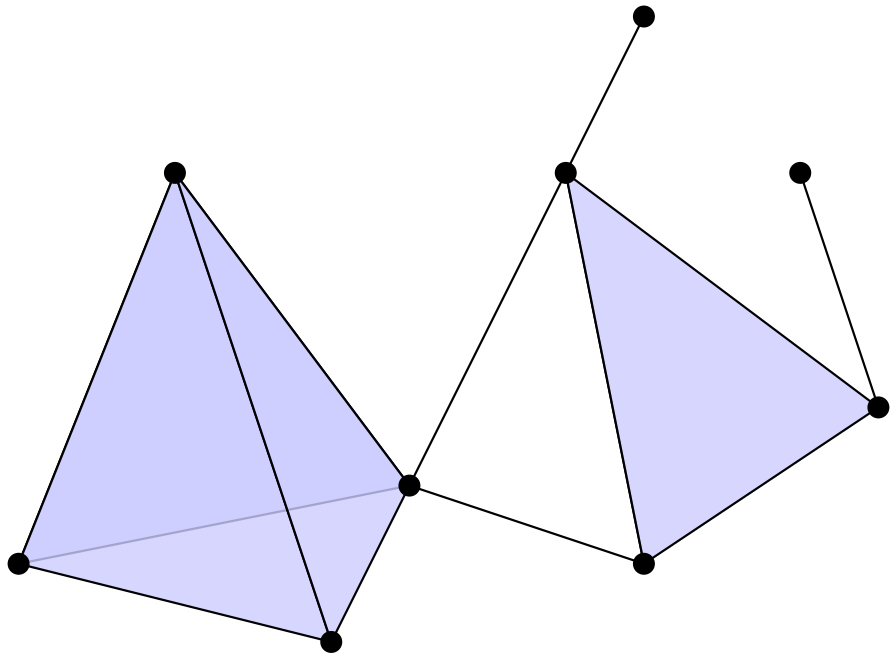
$$\tilde{\chi}(\Delta) = -f_{-1} + f_0 - f_1 + f_2 - f_3 + \cdots$$

Reduced Euler characteristic of an abstract simplicial complex



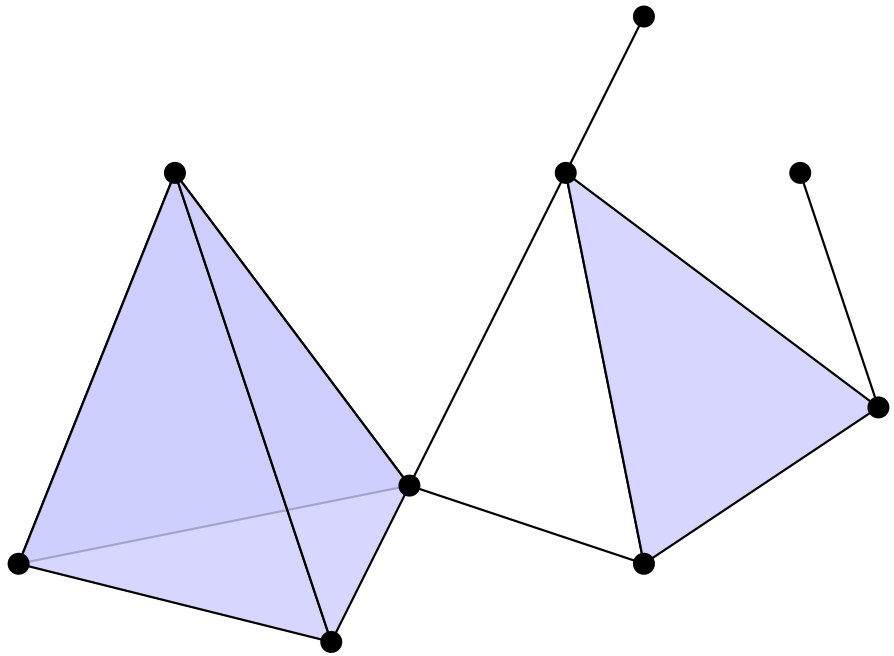
$$\begin{aligned}\tilde{\chi}(\Delta) &= -f_{-1} + f_0 - f_1 + f_2 - f_3 + \cdots \\ &= -1 + 9 - 13 + 5 = 0\end{aligned}$$

Reduced Euler characteristic of an abstract simplicial complex



$$\tilde{\chi}(\Delta) = \tilde{\beta}_0 - \tilde{\beta}_1 + \tilde{\beta}_2 - \tilde{\beta}_3 + \cdots$$

Reduced Euler characteristic of an abstract simplicial complex



$$\tilde{\chi}(\Delta) = \tilde{\beta}_0 - \tilde{\beta}_1 + \tilde{\beta}_2 - \tilde{\beta}_3 + \cdots$$
$$= 0 - 1 + 1 = 0$$

Outline

Outline

Complexity class of EulerChar

Combinatorial commutative algebra algorithms

- Based on Hilbert-Poincaré series
- Based on Mayer-Vietoris trees
- Experiments

Complexity class of $\tilde{\chi}(\Delta)$

- Kaibel and Pfetsch [2003]
  The problem of determining the complexity class of the Euler characteristic of an abstract simplicial complex given its facets is open.
- Best reported algorithm: Kaibel and Pfetsch [2002] via Hasse diagram of the facet intersection lattice and its Möbius function.
  Complexity of computing Hasse diagram V of facet intersection: $O(min\{m, n\} \cdot d \cdot \alpha)$ with $n =$ number of vertices, $m =$ number of facets, $d = |V|$, $\alpha =$ number of incidences between vertices and facets.
  Complexity of computing Möbius function: same.

Related results:

- The problem of computing the f-vector of a simplicial complex given its facet is $\#P$-hard [Pfetsch 2002]
- The complexity of Euler characteristic has been studied for other inputs:
  - The Euler characteristic of a sheaf is $\#P$-hard [Bach 1999]
  - Single exponential time algorithm for computing Euler characteristic of arbitrary closed semi-algebraic sets [Basu 1999]
  - A closed semi-linear set given by an additive circuit. Studied in the context of additive BSS-machines. EulerChar is $FP_{add}^{\#P_{add}}$-complete [Bürgisser and Cucker 2004]

Complexity class of EulerChar

- $\#P$: counting problems associated to the decision problems in NP [Valiant 1979].
- $\#P$-complete:
  - $\#P$
  - Any other problem in $\#P$ can be reduced to it in polynomial time

Complexity class of EulerChar

- $\#P$: counting problems associated to the decision problems in NP [Valiant 1979].
- $\#P$-complete:
  - $\#P$
  - Any other problem in $\#P$ can be reduced to it in polynomial time

Complexity class of EulerChar

- $\#P$: counting problems associated to the decision problems in NP [Valiant 1979].
- $\#P$-complete:
  - $\#P$
  - Any other problem in $\#P$ can be reduced to it in polynomial time

Complexity class of EulerChar

- $\#P$: counting problems associated to the decision problems in NP [Valiant 1979].
- $\#P$-complete:
  - $\#P$
  - Any other problem in $\#P$ can be reduced to it in polynomial time

Example

Counting version of **kSAT** is $\#P$-complete:

- **kSAT** formula: conjunction of disjunctions of **k** literals
- Counting problem: How many satisfying truth assignments? (Decision version: Is the formula satisfiable?)

$$(a \vee \neg b) \wedge (a \vee c) \wedge (\neg b \vee c)$$

$$\{(0,0,1),(1,0,1),(1,0,0),(1,1,1)\}$$

Theorem

The problem of computing the Euler characteristic of an abstract simplicial complex given by its facets is $\#P$-complete.

Proof:

1. Show **EulerChar** is in $\#P$.
2. Reduce **Count-SAT** to **EulerChar** in polynomial time □

Theorem

The problem of computing the Euler characteristic of an abstract simplicial complex given by its facets is $\#P$-complete.

Proof:

1. Show **EulerChar** is in $\#P$.
2. Reduce **Count-SAT** to **EulerChar** in polynomial time □

Step 1: **EulerChar** is in $\#P$

- Compute $\tilde{\chi}(\Delta)$ is equivalent to compute $\tilde{\chi}(\Delta) + 2^{n-1}$ (we add the extra term to avoid negative numbers, which do not **count**).
- $2^{n-1} =$ number of even subsets of a set of n elements.
- $\tilde{\chi}(\Delta) =$ number of odd faces of Δ minus number of even faces.
- With the extra term we count odd faces and even non-faces.
- The decision problem we are using is ”does this complex have any odd face or even non-face?”
- **EulerChar** is in $\#P$ □

Step 1: **EulerChar** is in $\#P$

- Compute $\tilde{\chi}(\Delta)$ is equivalent to compute $\tilde{\chi}(\Delta) + 2^{n-1}$ (we add the extra term to avoid negative numbers, which do not **count**).
- $2^{n-1} =$ number of even subsets of a set of n elements.
- $\tilde{\chi}(\Delta) =$ number of odd faces of Δ minus number of even faces.
- With the extra term we count odd faces and even non-faces.
- The decision problem we are using is ”does this complex have any odd face or even non-face?”
- **EulerChar** is in $\#P$ □

Step 1: **EulerChar** is in $\#P$

- Compute $\tilde{\chi}(\Delta)$ is equivalent to compute $\tilde{\chi}(\Delta) + 2^{n-1}$ (we add the extra term to avoid negative numbers, which do not **count**).
- $2^{n-1} =$ number of even subsets of a set of n elements.
- $\tilde{\chi}(\Delta) =$ number of odd faces of Δ minus number of even faces.
- With the extra term we count odd faces and even non-faces.
- The decision problem we are using is "does this complex have any odd face or even non-face?"
- **EulerChar** is in $\#P$ □

Step 1: **EulerChar** is in $\#P$

- Compute $\tilde{\chi}(\Delta)$ is equivalent to compute $\tilde{\chi}(\Delta) + 2^{n-1}$ (we add the extra term to avoid negative numbers, which do not **count**).
- $2^{n-1} =$ number of even subsets of a set of n elements.
- $\tilde{\chi}(\Delta) =$ number of odd faces of Δ minus number of even faces.
- With the extra term we count odd faces and even non-faces.
- The decision problem we are using is "does this complex have any odd face or even non-face?"
- **EulerChar** is in $\#P$ □

Step 1: **EulerChar** is in $\#P$

- Compute $\tilde{\chi}(\Delta)$ is equivalent to compute $\tilde{\chi}(\Delta) + 2^{n-1}$ (we add the extra term to avoid negative numbers, which do not **count**).
- $2^{n-1} =$ number of even subsets of a set of n elements.
- $\tilde{\chi}(\Delta) =$ number of odd faces of Δ minus number of even faces.
- With the extra term we count odd faces and even non-faces.
- The decision problem we are using is "does this complex have any odd face or even non-face?"
- **EulerChar** is in $\#P$ □

Step 1: **EulerChar** is in $\#P$

- Compute $\tilde{\chi}(\Delta)$ is equivalent to compute $\tilde{\chi}(\Delta) + 2^{n-1}$ (we add the extra term to avoid negative numbers, which do not **count**).
- $2^{n-1} =$ number of even subsets of a set of n elements.
- $\tilde{\chi}(\Delta) =$ number of odd faces of Δ minus number of even faces.
- With the extra term we count odd faces and even non-faces.
- The decision problem we are using is "does this complex have any odd face or even non-face?"
- **EulerChar** is in $\#P$ □

Step 1: **EulerChar** is in $\#P$

- Compute $\tilde{\chi}(\Delta)$ is equivalent to compute $\tilde{\chi}(\Delta) + 2^{n-1}$ (we add the extra term to avoid negative numbers, which do not **count**).
- $2^{n-1} =$ number of even subsets of a set of n elements.
- $\tilde{\chi}(\Delta) =$ number of odd faces of Δ minus number of even faces.
- With the extra term we count odd faces and even non-faces.
- The decision problem we are using is ”does this complex have any odd face or even non-face?”
- **EulerChar** is in $\#P$ □

Step 1: **EulerChar** is in $\#P$

- Compute $\tilde{\chi}(\Delta)$ is equivalent to compute $\tilde{\chi}(\Delta) + 2^{n-1}$ (we add the extra term to avoid negative numbers, which do not **count**).
- $2^{n-1} =$ number of even subsets of a set of n elements.
- $\tilde{\chi}(\Delta) =$ number of odd faces of Δ minus number of even faces.
- With the extra term we count odd faces and even non-faces.
- The decision problem we are using is ”does this complex have any odd face or even non-face?”
- **EulerChar** is in $\#P$ □

Step 2: **Count-SAT** can be reduced to **EulerChar**

- Given a **SAT** formula S we construct a graph $G(S)$.
- From $G(S)$ we construct an abstract simplicial complex $\Delta(G(S))$ such that
- $\tilde{\chi}(\Delta(G(S))$ equals the number of solutions of S.

Step 2: **Count-SAT** can be reduced to **EulerChar**

- Given a **SAT** formula S we construct a graph $G(S)$.
- From $G(S)$ we construct an abstract simplicial complex $\Delta(G(S))$ such that
- $\tilde{\chi}(\Delta(G(S))$ equals the number of solutions of S.

Step 2: **Count-SAT** can be reduced to **EulerChar**

- Given a **SAT** formula S we construct a graph $G(S)$.
- From $G(S)$ we construct an abstract simplicial complex $\Delta(G(S))$ such that
- $\tilde{\chi}(\Delta(G(S))$ equals the number of solutions of S.

Step 2: **Count-SAT** can be reduced to **EulerChar**

- Given a **SAT** formula S we construct a graph $G(S)$.
- From $G(S)$ we construct an abstract simplicial complex $\Delta(G(S))$ such that
- $\tilde{\chi}(\Delta(G(S))$ equals the number of solutions of S.

Step 2: **Count-SAT** can be reduced to **EulerChar**

Given a graph G, $\Delta(G)$ is the simplicial complex on the vertices of G whose facets are the complements of the edges of G.

A set of vertices is then a face of $\Delta(G)$ if and only if the complement is dependent in G (contains an edge).

Then

$$\tilde{\chi}(\Delta(G)) = (-1)^n \sum_{\sigma \text{ independent in } G} (-1)^{|\sigma|}$$

Step 2: **Count-SAT** can be reduced to **EulerChar**

Given a graph G, $\Delta(G)$ is the simplicial complex on the vertices of G whose facets are the complements of the edges of G.
A set of vertices is then a face of $\Delta(G)$ if and only if the complement is dependent in G (contains an edge).

Then

$$\tilde{\chi}(\Delta(G)) = (-1)^n \sum_{\sigma \text{ independent in } G} (-1)^{|\sigma|}$$

Step 2: **Count-SAT** can be reduced to **EulerChar**

Given a graph G, $\Delta(G)$ is the simplicial complex on the vertices of G whose facets are the complements of the edges of G.
A set of vertices is then a face of $\Delta(G)$ if and only if the complement is dependent in G (contains an edge).

Then

$$\tilde{\chi}(\Delta(G)) = (-1)^n \sum_{\sigma \text{ independent in } G} (-1)^{|\sigma|}$$

Step 2: **Count-SAT** can be reduced to **EulerChar**

Given a graph G, $\Delta(G)$ is the simplicial complex on the vertices of G whose facets are the complements of the edges of G.
A set of vertices is then a face of $\Delta(G)$ if and only if the complement is dependent in G (contains an edge).

Then

$$\tilde{\chi}(\Delta(G)) = (-1)^n \sum_{\sigma \text{ independent in } G} (-1)^{|\sigma|}$$

Construction of the graph $G(S)$

Let S be a **SAT** formula on variables $v_1, v_2, \ldots$ and clauses $c_1, c_2, \ldots$.

- For each variable v_i introduce a 3-clique with vertices T_i, F_i, D_i (True, False, Dummy).
- For each clause c_j introduce a vertex C_j
- If v_i appears in c_j with no negation we add an edge between T_i and C_j, if it appears negated, we add an edge between F_i and C_j.

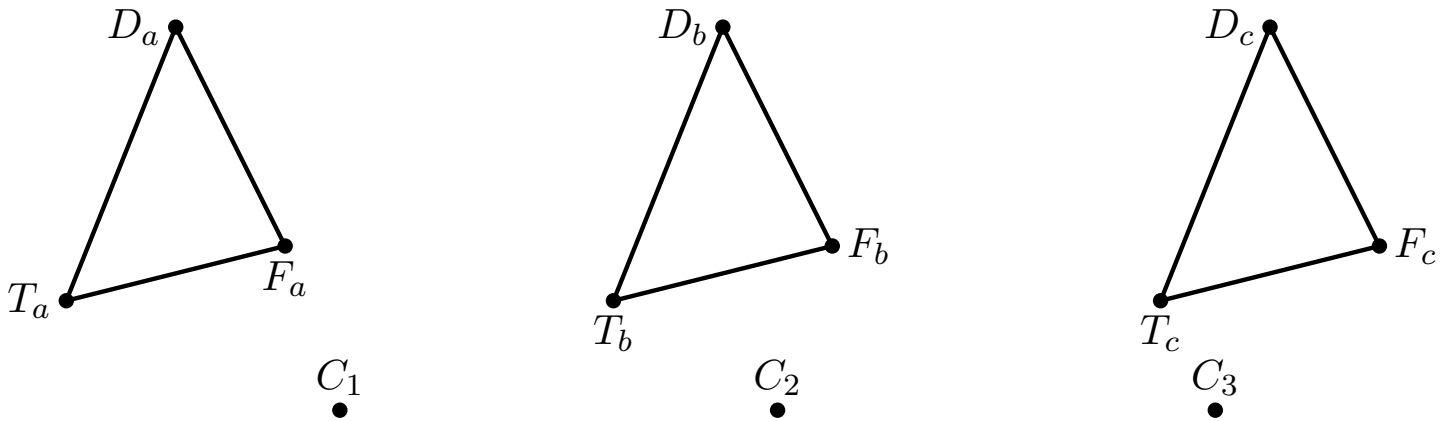
$$C_1 = (a \vee \neg b), C_2 = (a \vee \neg c), C_3 = (\neg b \vee c)$$

Construction of the graph $G(S)$

Let S be a **SAT** formula on variables $v_1, v_2, \ldots$ and clauses $c_1, c_2, \ldots$.

- For each variable v_i introduce a 3-clique with vertices T_i, F_i, D_i (True, False, Dummy).
- For each clause c_j introduce a vertex C_j
- If v_i appears in c_j with no negation we add an edge between T_i and C_j, if it appears negated, we add an edge between F_i and C_j.

$$C_1 = (a \vee \neg b), C_2 = (a \vee \neg c), C_3 = (\neg b \vee c)$$

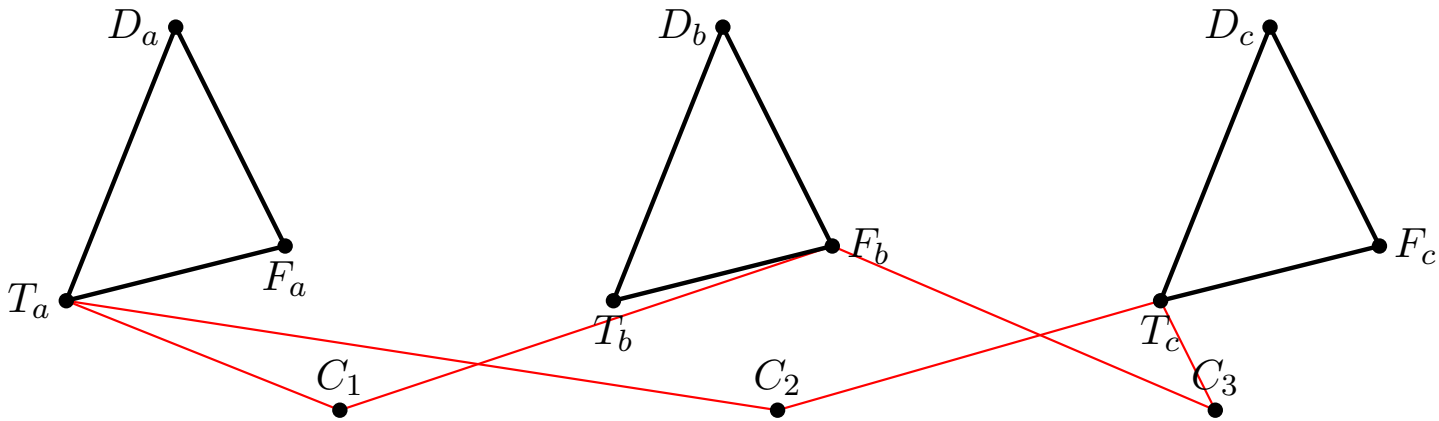


Construction of the graph $G(S)$

Let S be a **SAT** formula on variables $v_1, v_2, \ldots$ and clauses $c_1, c_2, \ldots$.

- For each variable v_i introduce a 3-clique with vertices T_i, F_i, D_i (True, False, Dummy).
- For each clause c_j introduce a vertex C_j
- If v_i appears in c_j with no negation we add an edge between T_i and C_j, if it appears negated, we add an edge between F_i and C_j.

$$C_1 = (a \vee \neg b), C_2 = (a \vee \neg c), C_3 = (\neg b \vee c)$$



Independent sets of the graph $G(S)$

- Let A be the set of vertices named D_i or C_j.
- Let B the set of vertices named T_i or F_i.
- Let I be the set of independent sets of $G(S)$.
- LetI_B be the sets of independent sets of $G(S)$ that are subsets of B.

Independent sets of the graph $G(S)$

- Let A be the set of vertices named D_i or C_j.
- Let B the set of vertices named T_i or F_i.
- Let I be the set of independent sets of $G(S)$.
- LetI_B be the sets of independent sets of $G(S)$ that are subsets of B.

Independent sets of the graph $G(S)$

- Let A be the set of vertices named D_i or C_j.
- Let B the set of vertices named T_i or F_i.
- Let I be the set of independent sets of $G(S)$.
- LetI_B be the sets of independent sets of $G(S)$ that are subsets of B.

Independent sets of the graph $G(S)$

- Let A be the set of vertices named D_i or C_j.
- Let B the set of vertices named T_i or F_i.
- Let I be the set of independent sets of $G(S)$.
- LetI_B be the sets of independent sets of $G(S)$ that are subsets of B.

Independent sets of the graph $G(S)$

- Let A be the set of vertices named D_i or C_j.
- Let B the set of vertices named T_i or F_i.
- Let I be the set of independent sets of $G(S)$.
- LetI_B be the sets of independent sets of $G(S)$ that are subsets of B.

Define a function $p : I \longrightarrow I_B$ by $d \mapsto d \setminus A$.

Independent sets of the graph $G(S)$

- Let A be the set of vertices named D_i or C_j.
- Let B the set of vertices named T_i or F_i.
- Let I be the set of independent sets of $G(S)$.
- LetI_B be the sets of independent sets of $G(S)$ that are subsets of B.

Define a function $p : I \to I_B$ by $d \mapsto d \setminus A$.
THEN:

1. $p^{-1}(d) = \{d\} \Rightarrow |d| = n$.
2. The set of such d is in bijection with the set of truth assignments that satisfy S.
3. If $p^{-1} \neq \{d\}$ then $P(p^{-1}(d)) = 0$.

Independent sets of the graph $G(S)$

- Let A be the set of vertices named D_i or C_j.
- Let B the set of vertices named T_i or F_i.
- Let I be the set of independent sets of $G(S)$.
- LetI_B be the sets of independent sets of $G(S)$ that are subsets of B.

Define a function $p : I \to I_B$ by $d \mapsto d \setminus A$.
THEN:

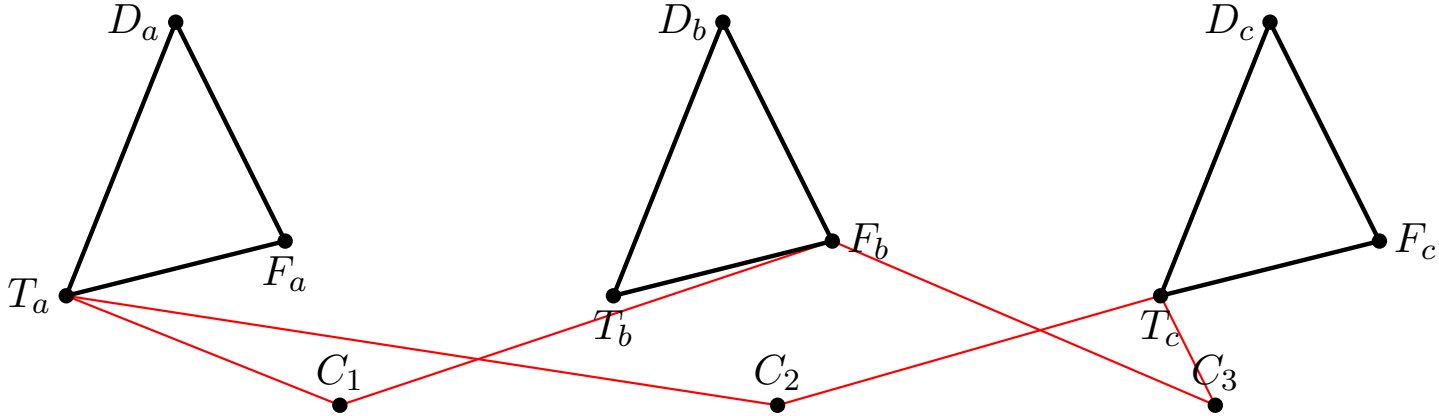
1. $p^{-1}(d) = \{d\} \Rightarrow |d| = n$.
2. The set of such d is in bijection with the set of truth assignments that satisfy S.
3. If $p^{-1} \neq \{d\}$ then $P(p^{-1}(d)) = 0$.

$$\#(\text{SAT assignments}) = (-1)^n \sum_{d \in I_B} P(p^{-1}(d)) = (-1)^n P(I)$$

$p^{-1}(d) = \{d\} \Rightarrow |d| = n$

Let $d \in I_B$ such that $p^{-1}(d) = \{d\}$

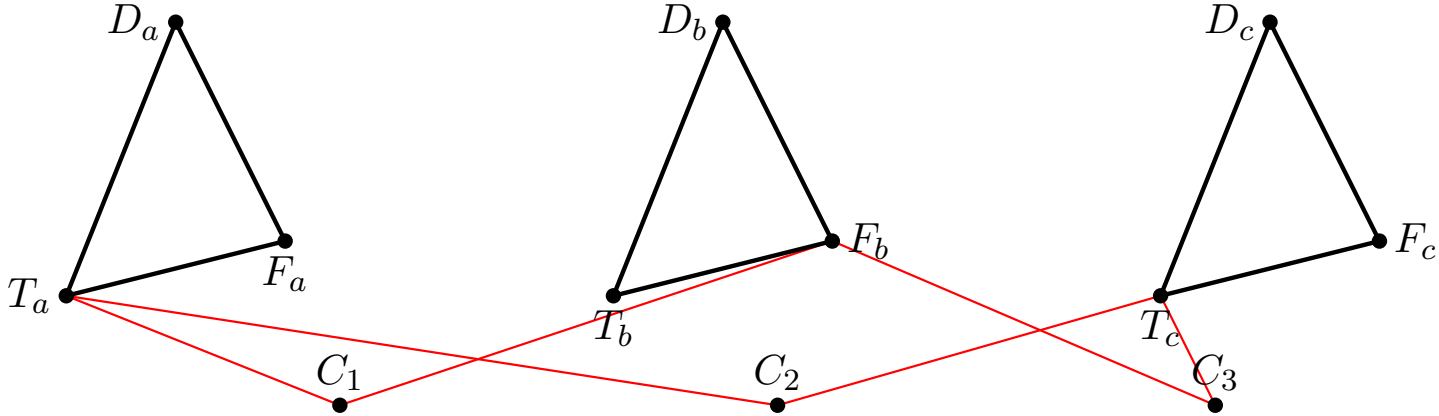
- $\forall i \; d \cup \{D_i\}$ is dependent, since D_i is only adjacent to T_i and F_i then d must contain one of T_i and F_i
- Hence d encodes a truth assignment to the variables of S
- $|d| = n$



$p^{-1}(d) = \{d\} \Rightarrow |d| = n$

Let $d \in I_B$ such that $p^{-1}(d) = \{d\}$

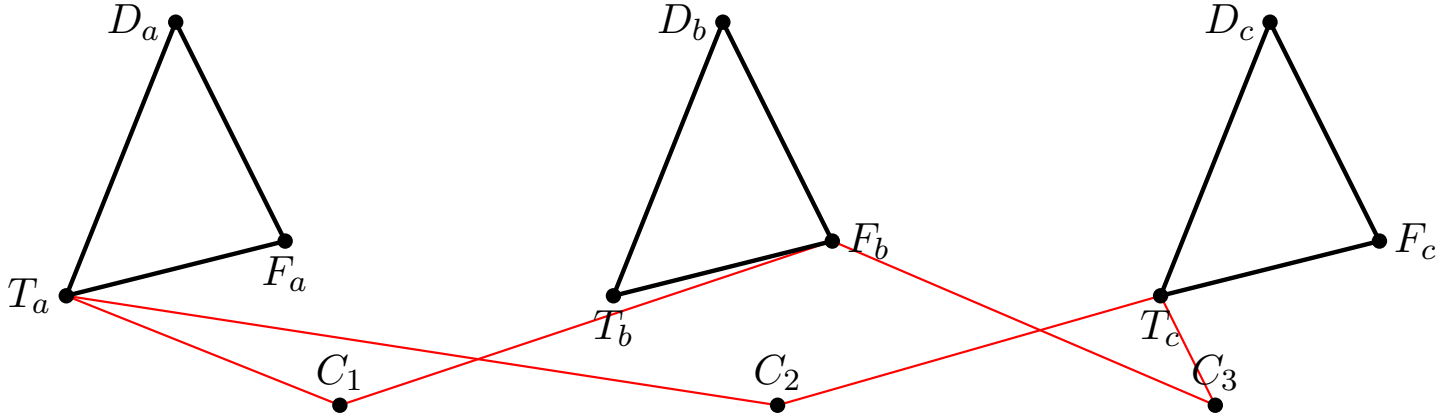
- $\forall i \; d \cup \{D_i\}$ is dependent, since D_i is only adjacent to T_i and F_i then d must contain one of T_i and F_i
- Hence d encodes a truth assignment to the variables of S
- $|d| = n$



$p^{-1}(d) = \{d\} \Rightarrow |d| = n$

Let $d \in I_B$ such that $p^{-1}(d) = \{d\}$

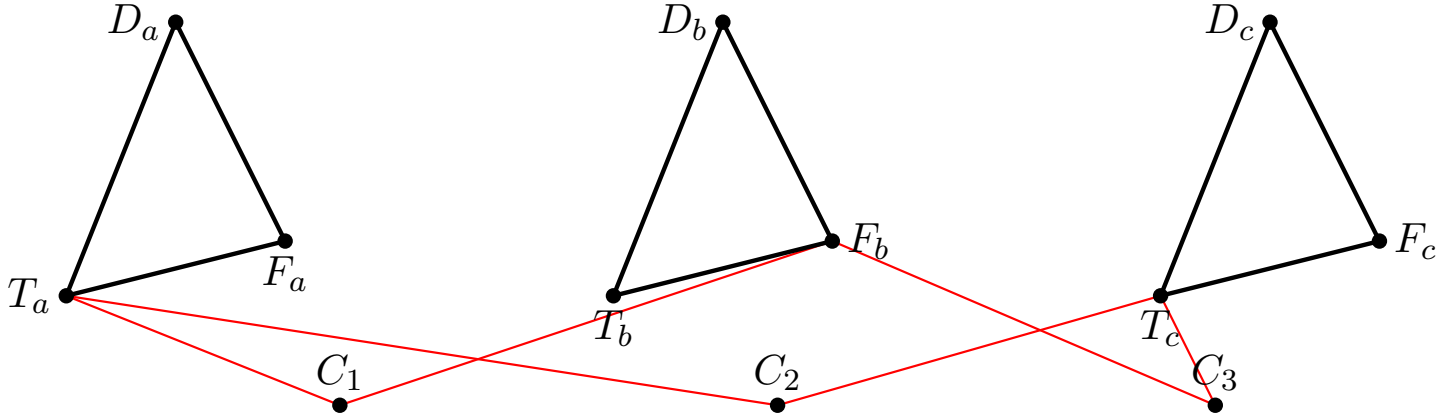
- $\forall i \; d \cup \{D_i\}$ is dependent, since D_i is only adjacent to T_i and F_i then d must contain one of T_i and F_i
- Hence d encodes a truth assignment to the variables of S
- $|d| = n$



$p^{-1}(d) = \{d\} \Rightarrow |d| = n$

Let $d \in I_B$ such that $p^{-1}(d) = \{d\}$

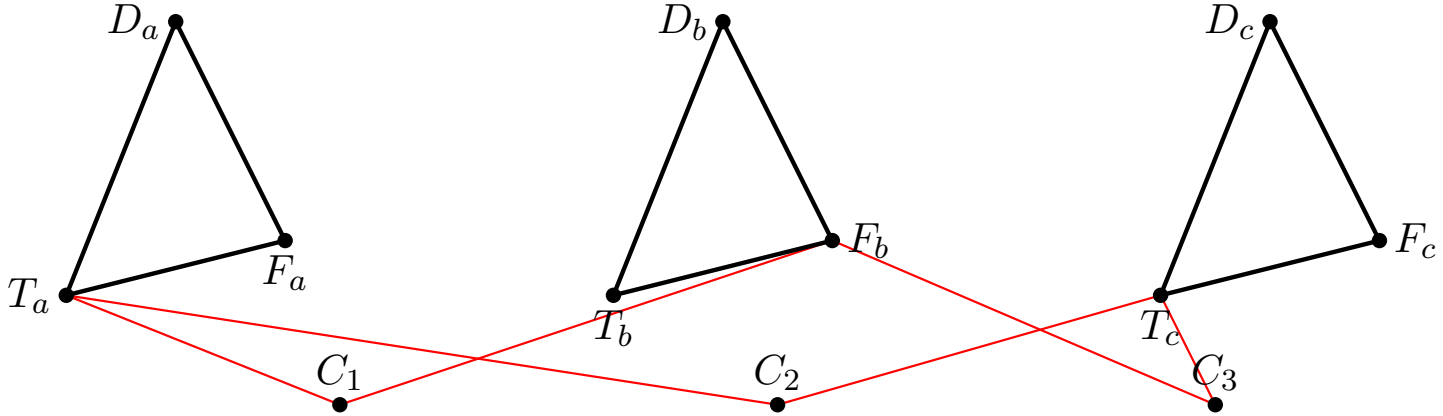
- $\forall i \; d \cup \{D_i\}$ is dependent, since D_i is only adjacent to T_i and F_i then d must contain one of T_i and F_i
- Hence d encodes a truth assignment to the variables of S
- $|d| = n$



The set of such d is in bijection with the set of truth assignments that satisfy S

Let $d \in I_B$ such that $p^{-1}(d) = \{d\}$

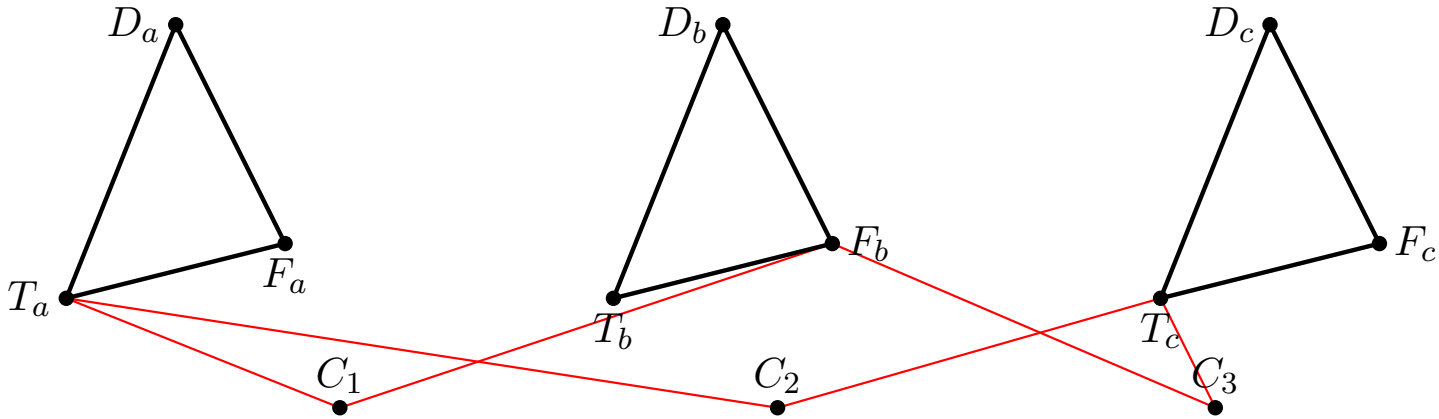
- $\forall j \; d \cup \{C_j\}$ is dependent, so d must contain one of T_i and F_i adjacent to C_j.
- Hence the truth assignment corresponding to d satisfies c_j.
- The other direction is trivial.



The set of such d is in bijection with the set of truth assignments that satisfy S

Let $d \in I_B$ such that $p^{-1}(d) = \{d\}$

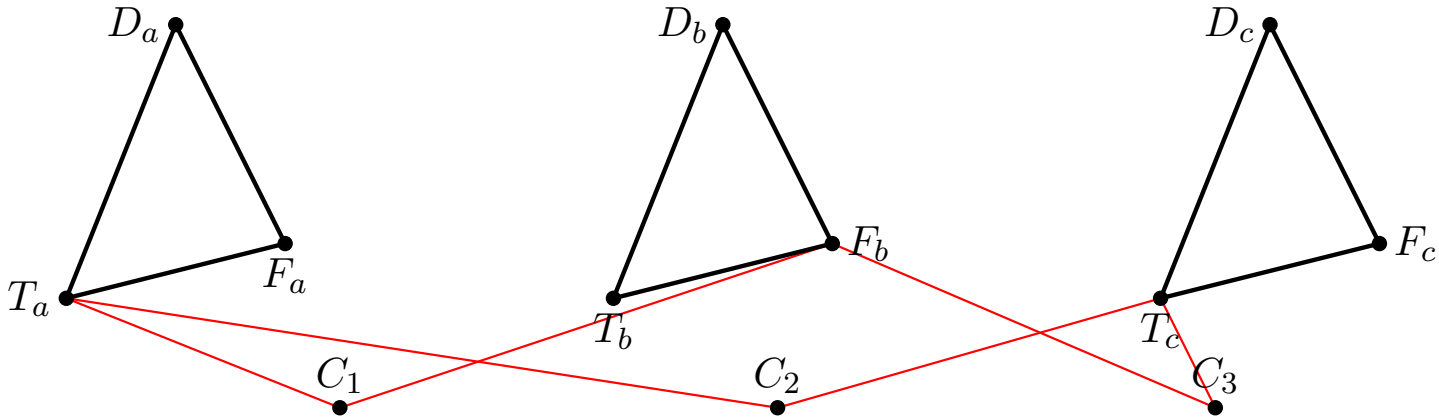
- $\forall j \; d \cup \{C_j\}$ is dependent, so d must contain one of T_i and F_i adjacent to C_j.
- Hence the truth assignment corresponding to d satisfies c_j.
- The other direction is trivial.



The set of such d is in bijection with the set of truth assignments that satisfy S

Let $d \in I_B$ such that $p^{-1}(d) = \{d\}$

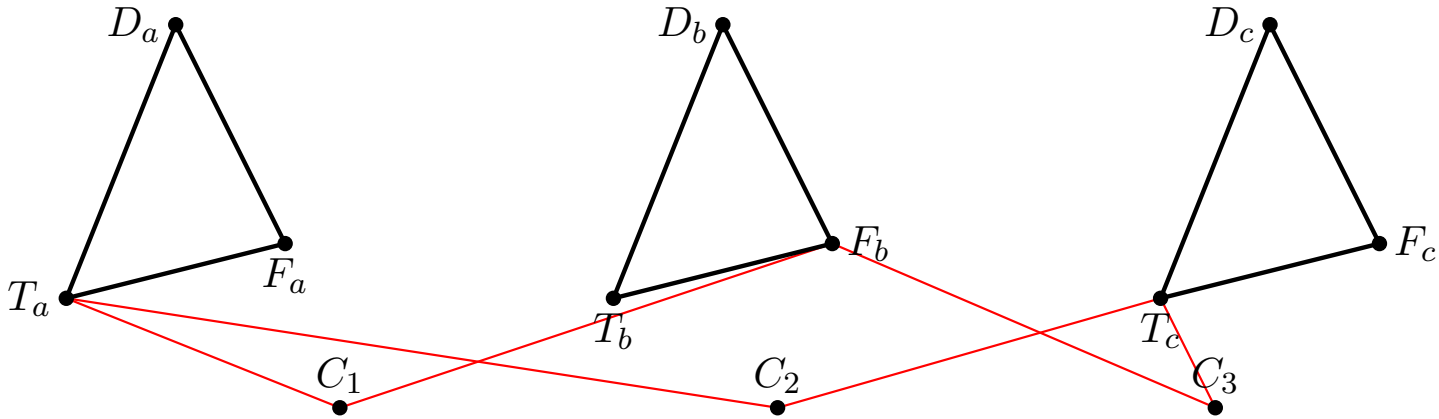
- $\forall j \; d \cup \{C_j\}$ is dependent, so d must contain one of T_i and F_i adjacent to C_j.
- Hence the truth assignment corresponding to d satisfies c_j.
- The other direction is trivial.



The set of such d is in bijection with the set of truth assignments that satisfy S

Let $d \in I_B$ such that $p^{-1}(d) = \{d\}$

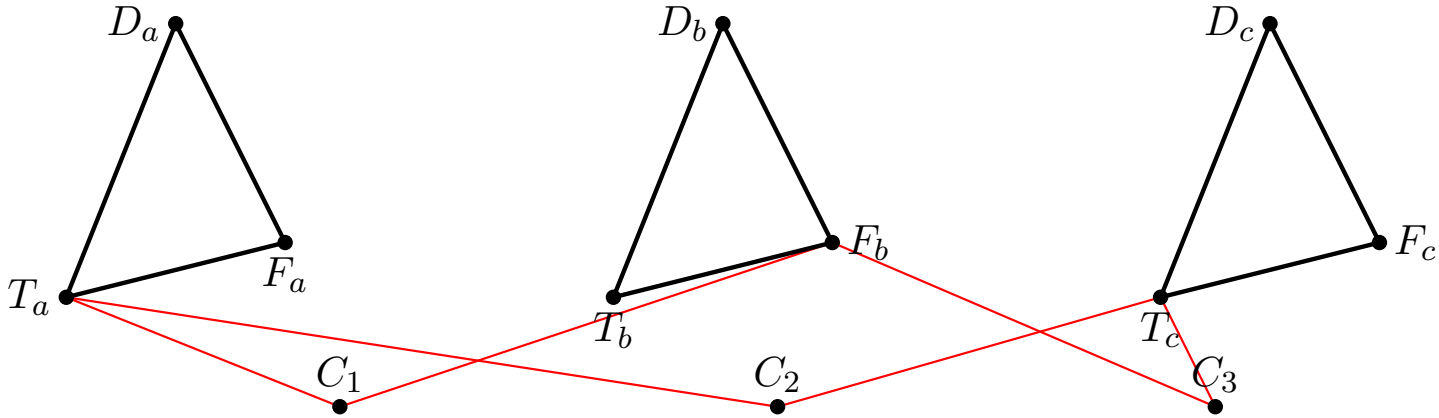
- $\forall j \; d \cup \{C_j\}$ is dependent, so d must contain one of T_i and F_i adjacent to C_j.
- Hence the truth assignment corresponding to d satisfies c_j.
- The other direction is trivial.



If $p^{-1} \neq \{d\}$ then $P(p^{-1}(d)) = 0$.

Let $d \in I_B$ such that $p^{-1}(d) \neq \{d\}$

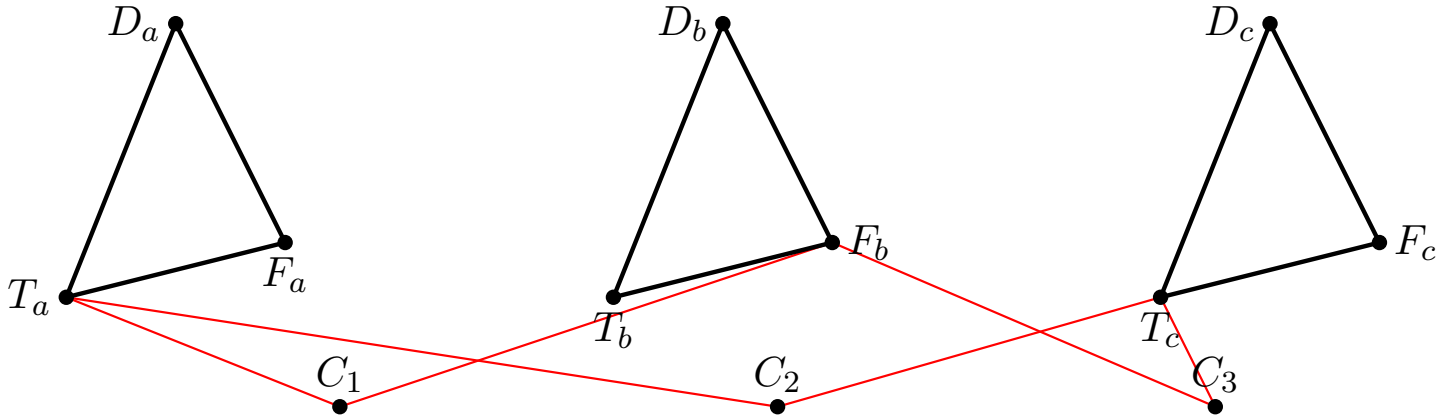
- We can pick $a \in A$ such that $d \cup \{a\}$ is still independent.
- a is not adjacent to any vertex in d or in A so let $E = \{h \in p^{-1}(d) | a \notin h\}$, $F = \{h \in p^{-1}(d) | a \in h\}$.
- $h \mapsto h \cup \{a\}$ is a bijection from E to F
- $P(E) = -P(F)$ and $\{E, F\}$ is a partition of $p^{-1}(d)$
- $P(p^{-1}(d)) = P(E) + P(F) = 0$



If $p^{-1} \neq \{d\}$ then $P(p^{-1}(d)) = 0$.

Let $d \in I_B$ such that $p^{-1}(d) \neq \{d\}$

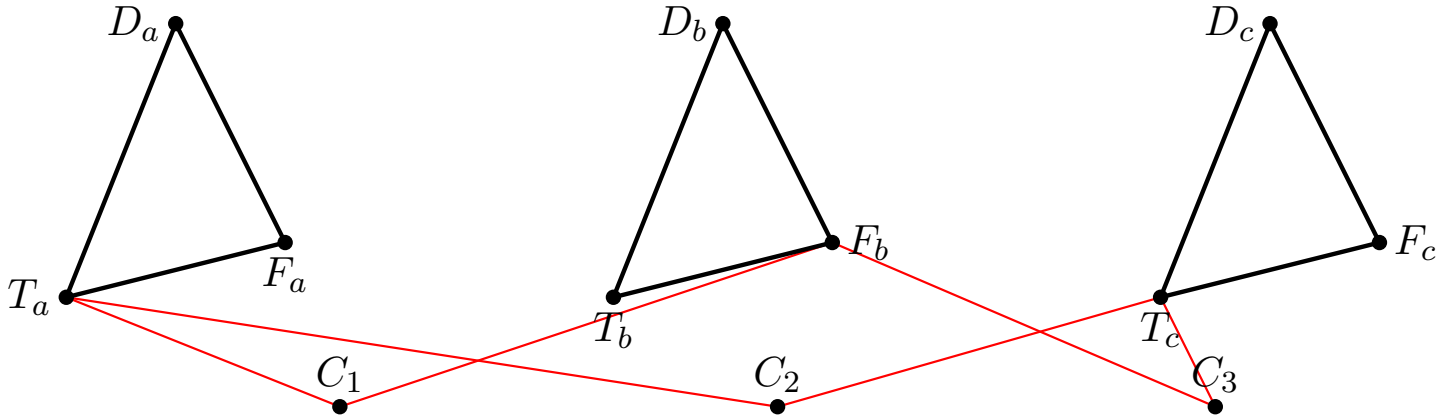
- We can pick $a \in A$ such that $d \cup \{a\}$ is still independent.
- a is not adjacent to any vertex in d or in A so let $E = \{h \in p^{-1}(d) | a \notin h\}$, $F = \{h \in p^{-1}(d) | a \in h\}$.
- $h \mapsto h \cup \{a\}$ is a bijection from E to F
- $P(E) = -P(F)$ and $\{E, F\}$ is a partition of $p^{-1}(d)$
- $P(p^{-1}(d)) = P(E) + P(F) = 0$



If $p^{-1} \neq \{d\}$ then $P(p^{-1}(d)) = 0$.

Let $d \in I_B$ such that $p^{-1}(d) \neq \{d\}$

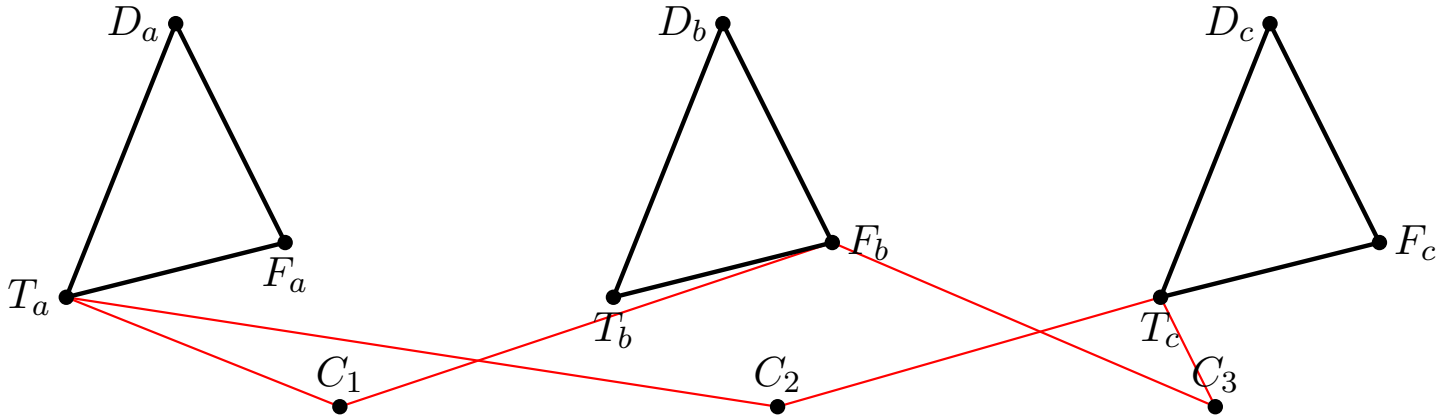
- We can pick $a \in A$ such that $d \cup \{a\}$ is still independent.
- a is not adjacent to any vertex in d or in A so let $E = \{h \in p^{-1}(d) | a \notin h\}$, $F = \{h \in p^{-1}(d) | a \in h\}$.
- $h \mapsto h \cup \{a\}$ is a bijection from E to F
- $P(E) = -P(F)$ and $\{E, F\}$ is a partition of $p^{-1}(d)$
- $P(p^{-1}(d)) = P(E) + P(F) = 0$



If $p^{-1} \neq \{d\}$ then $P(p^{-1}(d)) = 0$.

Let $d \in I_B$ such that $p^{-1}(d) \neq \{d\}$

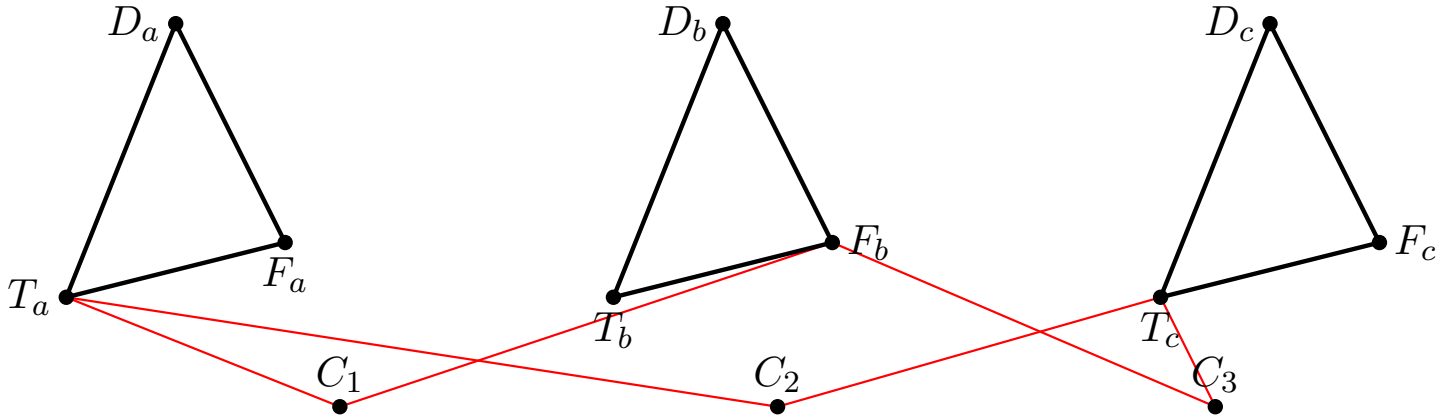
- We can pick $a \in A$ such that $d \cup \{a\}$ is still independent.
- a is not adjacent to any vertex in d or in A so let $E = \{h \in p^{-1}(d) | a \notin h\}$, $F = \{h \in p^{-1}(d) | a \in h\}$.
- $h \mapsto h \cup \{a\}$ is a bijection from E to F
- $P(E) = -P(F)$ and $\{E, F\}$ is a partition of $p^{-1}(d)$
- $P(p^{-1}(d)) = P(E) + P(F) = 0$



If $p^{-1} \neq \{d\}$ then $P(p^{-1}(d)) = 0$.

Let $d \in I_B$ such that $p^{-1}(d) \neq \{d\}$

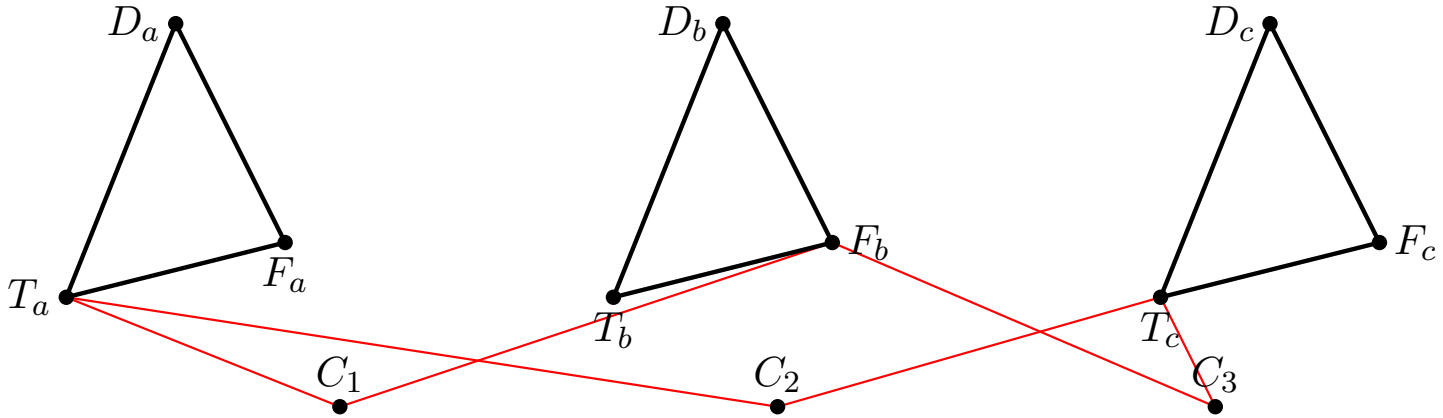
- We can pick $a \in A$ such that $d \cup \{a\}$ is still independent.
- a is not adjacent to any vertex in d or in A so let $E = \{h \in p^{-1}(d) | a \notin h\}$, $F = \{h \in p^{-1}(d) | a \in h\}$.
- $h \mapsto h \cup \{a\}$ is a bijection from E to F
- $P(E) = -P(F)$ and $\{E, F\}$ is a partition of $p^{-1}(d)$
- $P(p^{-1}(d)) = P(E) + P(F) = 0$



If $p^{-1} \neq \{d\}$ then $P(p^{-1}(d)) = 0$.

Let $d \in I_B$ such that $p^{-1}(d) \neq \{d\}$

- We can pick $a \in A$ such that $d \cup \{a\}$ is still independent.
- a is not adjacent to any vertex in d or in A so let $E = \{h \in p^{-1}(d) | a \notin h\}$, $F = \{h \in p^{-1}(d) | a \in h\}$.
- $h \mapsto h \cup \{a\}$ is a bijection from E to F
- $P(E) = -P(F)$ and $\{E, F\}$ is a partition of $p^{-1}(d)$
- $P(p^{-1}(d)) = P(E) + P(F) = 0$



Outline

Combinatorial commutative algebra algorithms

- Simplicial complex $\Delta \rightsquigarrow$ Stanley-Reisner ideal I_Δ
- Betti numbers of $\Delta \rightsquigarrow$ Betti numbers of I_Δ

We construct a polynomial ring with one variable x_v for each element v in the ground set of Δ. Then define

$$\psi(\sigma) = \prod_{v \notin \sigma} x_v \qquad I^\Delta = \langle \psi(\sigma) | \sigma \text{ facet of } \Delta \rangle$$

- Minimal generators of I^Δ in linear time from the facets of Δ.
- lcm-lattice of the generators of I^Δ is the same as the intersection lattice of the facets of Δ.
- I^Δ is the Alexander dual of the Stanley-Reisner ideal I_Δ.

Combinatorial commutative algebra algorithms

- Simplicial complex $\Delta \rightsquigarrow$ Stanley-Reisner ideal I_Δ
- Betti numbers of $\Delta \rightsquigarrow$ Betti numbers of I_Δ

We construct a polynomial ring with one variable x_v for each element v in the ground set of Δ. Then define

$$\psi(\sigma) = \prod_{v \notin \sigma} x_v \qquad I^\Delta = \langle \psi(\sigma) | \sigma \text{ facet of } \Delta \rangle$$

- Minimal generators of I^Δ in linear time from the facets of Δ.
- lcm-lattice of the generators of I^Δ is the same as the intersection lattice of the facets of Δ.
- I^Δ is the Alexander dual of the Stanley-Reisner ideal I_Δ.

Combinatorial commutative algebra algorithms

- Simplicial complex $\Delta \rightsquigarrow$ Stanley-Reisner ideal I_Δ
- Betti numbers of $\Delta \rightsquigarrow$ Betti numbers of I_Δ

We construct a polynomial ring with one variable x_v for each element v in the ground set of Δ. Then define

$$\psi(\sigma) = \prod_{v \notin \sigma} x_v \qquad I^\Delta = \langle \psi(\sigma) | \sigma \text{ facet of } \Delta \rangle$$

- Minimal generators of I^Δ in linear time from the facets of Δ.
- lcm-lattice of the generators of I^Δ is the same as the intersection lattice of the facets of Δ.
- I^Δ is the Alexander dual of the Stanley-Reisner ideal I_Δ.

Combinatorial commutative algebra algorithms

- Simplicial complex $\Delta \rightsquigarrow$ Stanley-Reisner ideal I_Δ
- Betti numbers of $\Delta \rightsquigarrow$ Betti numbers of I_Δ

We construct a polynomial ring with one variable x_v for each element v in the ground set of Δ. Then define

$$\psi(\sigma) = \prod_{v \notin \sigma} x_v \qquad I^\Delta = \langle \psi(\sigma) | \sigma \text{ facet of } \Delta \rangle$$

- Minimal generators of I^Δ in linear time from the facets of Δ.
- lcm-lattice of the generators of I^Δ is the same as the intersection lattice of the facets of Δ.
- I^Δ is the Alexander dual of the Stanley-Reisner ideal I_Δ.

Combinatorial commutative algebra algorithms

- Simplicial complex $\Delta \rightsquigarrow$ Stanley-Reisner ideal I_Δ
- Betti numbers of $\Delta \rightsquigarrow$ Betti numbers of I_Δ

We construct a polynomial ring with one variable x_v for each element v in the ground set of Δ. Then define

$$\psi(\sigma) = \prod_{v \notin \sigma} x_v \qquad I^\Delta = \langle \psi(\sigma) | \sigma \text{ facet of } \Delta \rangle$$

- Minimal generators of I^Δ in linear time from the facets of Δ.
- lcm-lattice of the generators of I^Δ is the same as the intersection lattice of the facets of Δ.
- I^Δ is the Alexander dual of the Stanley-Reisner ideal I_Δ.

Based on Hilbert-Poincaré series

Hilbert series, univariate

The h-vector of Δ equals the set of coefficients of the numerator of the Hilbert-Poincaré series.

The n-th entry of the h-vector of Δ equals $\tilde{\chi}(\Delta)$.

Hilbert series, multivariate

The multigraded Hilbert-Poincaré series of a monomial ideal I^{Δ} can be written as a fraction with $(x_1 - 1) \cdots (x_n - 1)$ in the denominator and a polynomial $H(I^{\Delta})$ in the numerator.

The reduced Euler characteristic of Δ is the coefficient of the term $x_1 \cdots x_n$ in $H(I^{\Delta})$.

Based on Hilbert-Poincaré series

Hilbert series, univariate
The h-vector of Δ equals the set of coefficients of the numerator of the Hilbert-Poincaré series.
The n-th entry of the h-vector of Δ equals $\tilde{\chi}(\Delta)$.

Hilbert series, multivariate
The multigraded Hilbert-Poincaré series of a monomial ideal I^{Δ} can be written as a fraction with $(x_1 - 1) \cdots (x_n - 1)$ in the denominator and a polynomial $H(I^{\Delta})$ in the numerator.
The reduced Euler characteristic of Δ is the coefficient of the term $x_1 \cdots x_n$ in $H(I^{\Delta})$.

Based on Hilbert-Poincaré series

The best known algorithm for Euler-Poincaré series is the Bigatti et al. algorithm, which is based on the equation

$$H(I) = H(I : p)p + H(I + \langle p \rangle)$$

for monomials p.

Since I^{Δ} is square free, and we can choose p to be so as well, we get that

$$\tilde{\chi}(I) = \tilde{\chi}(I : p) + \tilde{\chi}(I + \langle p \rangle). \tag{1}$$

This suggests a recursive algorithm for computing $\tilde{\chi}(\Delta)$ in terms of $\tilde{\chi}(I^{\Delta})$.

Based on Hilbert-Poincaré series

The best known algorithm for Euler-Poincaré series is the Bigatti et al. algorithm, which is based on the equation

$$H(I) = H(I : p)p + H(I + \langle p \rangle)$$

for monomials p.
Since I^{Δ} is square free, and we can choose p to be so as well, we get that

$$\tilde{\chi}(I) = \tilde{\chi}(I : p) + \tilde{\chi}(I + \langle p \rangle). \tag{1}$$

This suggests a recursive algorithm for computing $\tilde{\chi}(\Delta)$ in terms of $\tilde{\chi}(I^{\Delta})$.

Based on Mayer-Vietoris trees

We use the Mayer-Vietoris sequence

$$0 \to \tilde{I} \to I' \oplus \langle m_r \rangle \to I \to 0 \tag{2}$$

where $I' = \langle m_1, \ldots, m_{r-1} \rangle$ and $\tilde{I} = I' \cap \langle m_r \rangle$.

This sequence leads to an iterative process to compute resolutions of monomial ideals. In particular, this provides a fast algorithm to compute the support of a multigraded resolution of I [S09].

In such a multigraded resolution $\gamma_{i,x_1 \cdots x_n}(I)$ denotes the rank of the multidegree $x_1 \cdots x_n$ component of its i-th module.

Using additivity of the Euler characteristic and the fact that the resolution is multigraded, we have that

$$\tilde{\chi}(\Delta) = \sum_i (-1)^i \gamma_{i,x_1 \cdots, x_n}(I_\Delta) \tag{3}$$

Based on Mayer-Vietoris trees

We use the Mayer-Vietoris sequence

$$0 \to \tilde{I} \to I' \oplus \langle m_r \rangle \to I \to 0 \tag{2}$$

where $I' = \langle m_1, \ldots, m_{r-1} \rangle$ and $\tilde{I} = I' \cap \langle m_r \rangle$.
This sequence leads to an iterative process to compute resolutions of monomial ideals. In particular, this provides a fast algorithm to compute the support of a multigraded resolution of I [S09].

In such a multigraded resolution $\gamma_{i,x_1 \cdots x_n}(I)$ denotes the rank of the multidegree $x_1 \cdots x_n$ component of its i-th module.
Using additivity of the Euler characteristic and the fact that the resolution is multigraded, we have that

$$\tilde{\chi}(\Delta) = \sum_i (-1)^i \gamma_{i,x_1 \cdots, x_n}(I_\Delta) \tag{3}$$

Based on Mayer-Vietoris trees

We use the Mayer-Vietoris sequence

$$0 \to \tilde{I} \to I' \oplus \langle m_r \rangle \to I \to 0 \tag{2}$$

where $I' = \langle m_1, \ldots, m_{r-1} \rangle$ and $\tilde{I} = I' \cap \langle m_r \rangle$.
This sequence leads to an iterative process to compute resolutions of monomial ideals. In particular, this provides a fast algorithm to compute the support of a multigraded resolution of I [S09].
In such a multigraded resolution $\gamma_{i,x_1 \cdots x_n}(I)$ denotes the rank of the multidegree $x_1 \cdots x_n$ component of its i-th module.
Using additivity of the Euler characteristic and the fact that the resolution is multigraded, we have that

$$\tilde{\chi}(\Delta) = \sum_i (-1)^i \gamma_{i,x_1 \cdots, x_n}(I_\Delta) \tag{3}$$

Based on Mayer-Vietoris trees

We use the Mayer-Vietoris sequence

$$0 \to \tilde{I} \to I' \oplus \langle m_r \rangle \to I \to 0 \tag{2}$$

where $I' = \langle m_1, \ldots, m_{r-1} \rangle$ and $\tilde{I} = I' \cap \langle m_r \rangle$.
This sequence leads to an iterative process to compute resolutions of monomial ideals. In particular, this provides a fast algorithm to compute the support of a multigraded resolution of I [S09].
In such a multigraded resolution $\gamma_{i,x_1 \cdots x_n}(I)$ denotes the rank of the multidegree $x_1 \cdots x_n$ component of its i-th module.
Using additivity of the Euler characteristic and the fact that the resolution is multigraded, we have that

$$\tilde{\chi}(\Delta) = \sum_i (-1)^i \gamma_{i,x_1 \cdots, x_n}(I_\Delta) \tag{3}$$

Implementations

Both algorithms have been implemented in C++ by the authors:

- The algorithm based on Hilbert-Poincaré series is implemented in **Frobby**.
- The algorithm based on Mayer-Vietoris trees is implemented in **CoCoALib**.

Implementations

Both algorithms have been implemented in C++ by the authors:

- The algorithm based on Hilbert-Poincaré series is implemented in **Frobby**.
- The algorithm based on Mayer-Vietoris trees is implemented in **CoCoALib**.

Implementations

Both algorithms have been implemented in C++ by the authors:

- The algorithm based on Hilbert-Poincaré series is implemented in **Frobby**.
- The algorithm based on Mayer-Vietoris trees is implemented in **CoCoALib**.

vars.	gens.	H-P	M-V	vars/gens	H-P/M-V
40	200	0.9s	3.7s	0.20	0.243
50	200	3.5s	12.2s	0.25	0.287
60	200	12.6s	34.1s	0.33	0.369
20	2000	0.2s	5.8s	0.0100	0.034
25	2000	0.9s	18.3s	0.0125	0.049
30	2000	4.2s	71.0s	0.0150	0.059
35	2000	19.7s	288.0s	0.0175	0.068
500	20	1.2s	0.1s	25	12.0
1000	20	3.5s	0.2s	50	17.5
1500	20	7.9s	0.3s	75	26.3
2000	20	12.1s	0.3s	100	40.3
500	30	21.4s	2.6s	16.67	8.2
1000	30	142.4s	12.3s	33.33	11.6
1500	30	360.0s	20.2s	50.00	17.8
2000	30	661.6s	32.8s	66.67	20.2

Table: Times for computing Euler characteristic.

vars.	gens.	H-P	M-V	vars/gens	H-P/M-V
40	200	0.9s	3.7s	0.20	0.243
50	200	3.5s	12.2s	0.25	0.287
60	200	12.6s	34.1s	0.33	0.369
20	2000	0.2s	5.8s	0.0100	0.034
25	2000	0.9s	18.3s	0.0125	0.049
30	2000	4.2s	71.0s	0.0150	0.059
35	2000	19.7s	288.0s	0.0175	0.068
500	20	1.2s	0.1s	25	12.0
1000	20	3.5s	0.2s	50	17.5
1500	20	7.9s	0.3s	75	26.3
2000	20	12.1s	0.3s	100	40.3
500	30	21.4s	2.6s	16.67	8.2
1000	30	142.4s	12.3s	33.33	11.6
1500	30	360.0s	20.2s	50.00	17.8
2000	30	661.6s	32.8s	66.67	20.2

Table: Times for computing Euler characteristic.