Polynomial time vertex enumeration of convex polytopes of bounded branch-width

Leen Stougie

In collaboration with Arne Reimers
Google, Munich

MED2^{3}3^2, London
July 16, 2018

Outline

Vertex enumeration

Given polytope by its outer description
$P = \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b, x \geq 0\}$
enumerate all its vertices

Vertex enumeration

Given polytope by its outer description
$P = \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b, x \geq 0\}$
enumerate all its vertices

Given polytope by its outer description
$P = \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b, x \geq 0\}$
compute its inner description
$P = \mathrm{conv.hull}\{x_1, \ldots, x_{|\mathcal{V}|}\}$

Vertex enumeration

Given polytope by its outer description
$P = \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b, x \geq 0\}$
enumerate all its vertices

Given polytope by its outer description
$P = \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b, x \geq 0\}$
compute its inner description
$P = \mathrm{conv.hull}\{x_1, \ldots, x_{|\mathcal{V}|}\}$

Can this be done in total polynomial time?

Vertex enumeration

Given polytope by its outer description
$P = \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b, x \geq 0\}$
enumerate all its vertices

Given polytope by its outer description
$P = \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b, x \geq 0\}$
compute its inner description
$P = \mathrm{conv.hull}\{x_1, \ldots, x_{|\mathcal{V}|}\}$

Can this be done in total polynomial time?
Time polynomial in size of input and output

Vertex enumeration

Open Problem: Can vertex enumeration of a polytope be done in total polynomial time?

Vertex enumeration

Open Problem: Can vertex enumeration of a polytope be done in total polynomial time?

It cannot be done for unbounded polyhedra unless $P = NP$ [Boros et al. 97]

Vertex enumeration

Open Problem: Can vertex enumeration of a polytope be done in total polynomial time?

It cannot be done for unbounded polyhedra unless $P = NP$ [Boros et al. 97]

It can be done for non-degenerate polytopes [Dyer 83]

Vertex enumeration

Open Problem: Can vertex enumeration of a polytope be done in total polynomial time?

It cannot be done for unbounded polyhedra unless $P = NP$ [Boros et al. 97]

It can be done for non-degenerate polytopes [Dyer 83]

Most popular method: Double Description method [Fukuda] or more sophisticated [Terzer & Stelling 08]

Decomposition of polyhedra

Many graph optimization problems are polytime solvable for graphs with bounded treewidth.

Decomposition of polyhedra

Many graph optimization problems are polytime solvable for graphs with bounded treewidth.

Tree decomposition and treewidth not applicable to polyhedra.

But the related concepts of branch-decomposition and branch-width are!

Decomposition of polyhedra

Many graph optimization problems are polytime solvable for graphs with bounded treewidth.

Tree decomposition and treewidth not applicable to polyhedra.

But the related concepts of branch-decomposition and branch-width are!

Branch-decomposition and branch width defined for matroids

Decomposition of polyhedra

Many graph optimization problems are polytime solvable for graphs with bounded treewidth.

Tree decomposition and treewidth not applicable to polyhedra.

But the related concepts of branch-decomposition and branch-width are!

Branch-decomposition and branch width defined for matroids

Our result: For $P = \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b, x \geq 0\}$, if the branch-width of the linear matroid on the columns of S is bounded by k, then we can enumerate all vertices $\mathcal{V}$ in running time $O(\mathcal{N}|\mathcal{V}|^{O(k)}t)$, where t is time for solving some LP's for feasibility checks

Outline

Metabolic Network

- Network of chemical reactions
  together performing some constructive and destructive tasks in a living cell
  e.g. photosynthesis, glycolysis

Metabolic Network

- Network of chemical reactions
  together performing some constructive and destructive tasks in a living cell
  e.g. photosynthesis, glycolysis
- A reaction transforms some chemical molecules into others
  - NH_3 and O_2

Metabolic Network

- Network of chemical reactions
  together performing some constructive and destructive tasks in a living cell
  e.g. photosynthesis, glycolysis
- A reaction transforms some chemical molecules into others
  - NH_3 and O_2

Metabolic Network

- Network of chemical reactions
  together performing some constructive and destructive tasks in a living cell
  e.g. photosynthesis, glycolysis
- A reaction transforms some chemical molecules into others
  - NH_3 and O_2
- Each reaction gives a column of the stoichiometric matrix

Stougiometric Matrix

Example: $1NH_3+2O_2 \rightarrow 1HNO_3+1H_2O$

	R
·	0
·	0
NH_3	-1
O_2	-2
HNO_3	$+1$
H_2O	$+1$
·	0
·	0

Flux

- The flux v_r of a reaction r is the rate at which the reaction takes place.

Flux

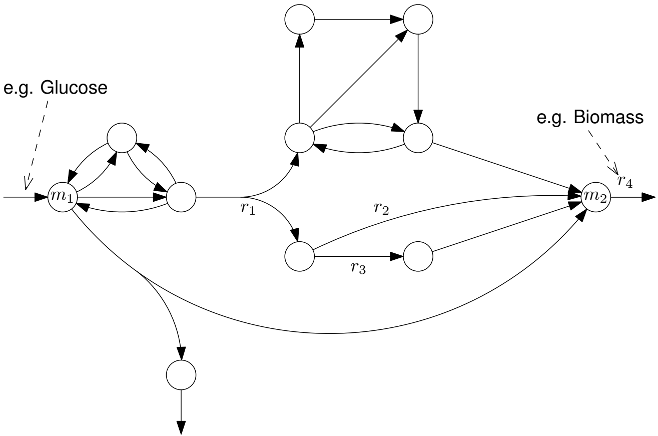
- The flux v_r of a reaction r is the rate at which the reaction takes place.
- The vector v with for every reaction r a coordinate v_r is called the flux vector.

Flux

- The flux v_r of a reaction r is the rate at which the reaction takes place.
- The vector v with for every reaction r a coordinate v_r is called the flux vector.
- Steady state assumption

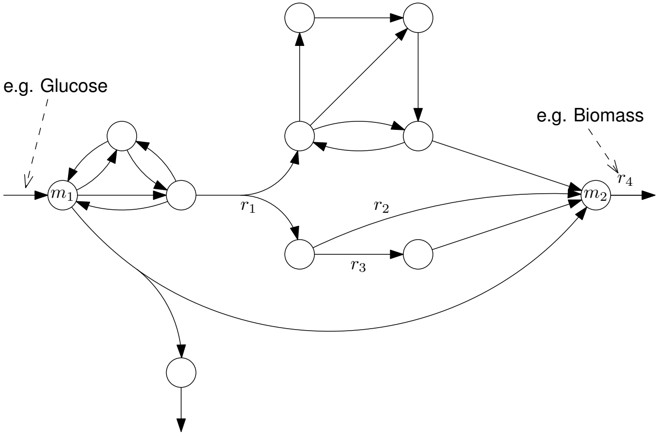
$$Sv = 0, v \geq 0$$

Running Example

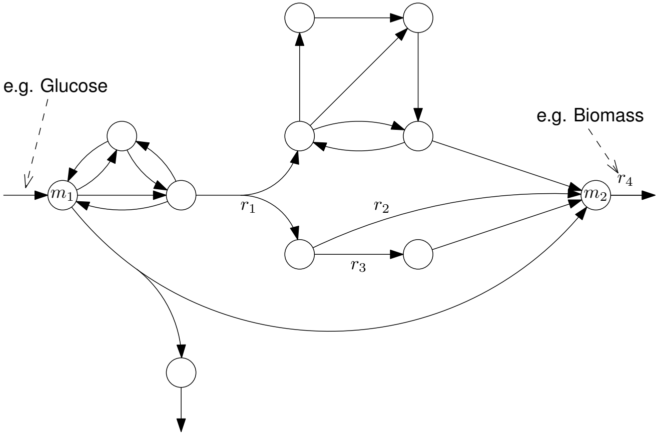


Here all coefficients of the stoichiometric matrix are $-1, 0, +1$.

Running Example

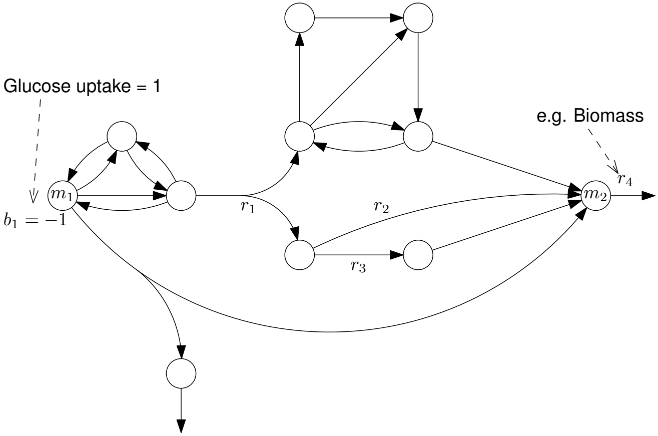


Running Example



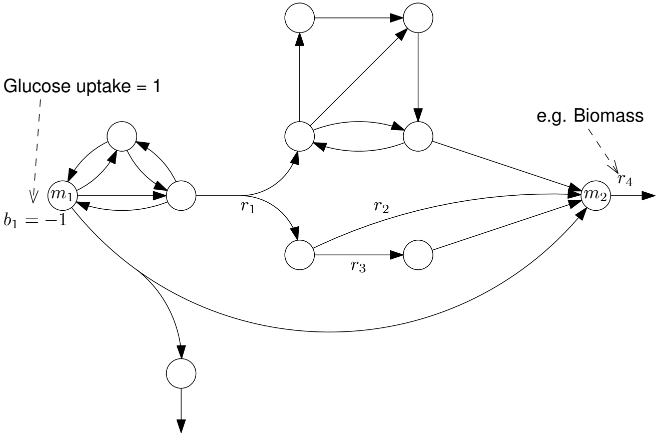
- Flux Space: $\{v : Sv = 0, v \geq 0\}$

Running Example



- Flux Space: $\{v : Sv = b, v \geq 0\}$

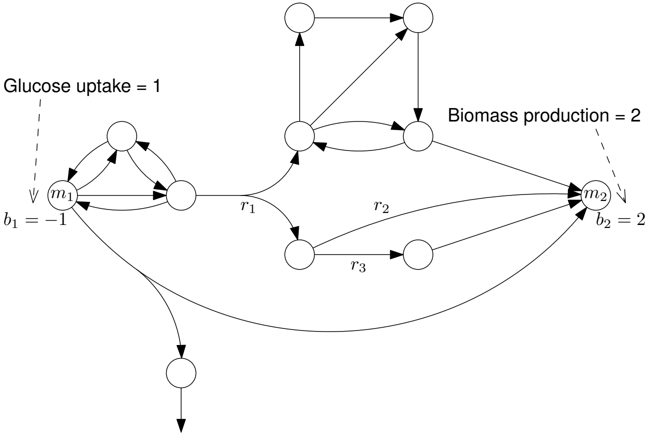
Running Example



- Flux Space: $\{v : Sv = b, v \geq 0\}$
- Optimize Biomass production (linear programming)

$$\max v_{biomass} \quad \text{subject to } Sv = 0, \; v_{glucose} = 1$$

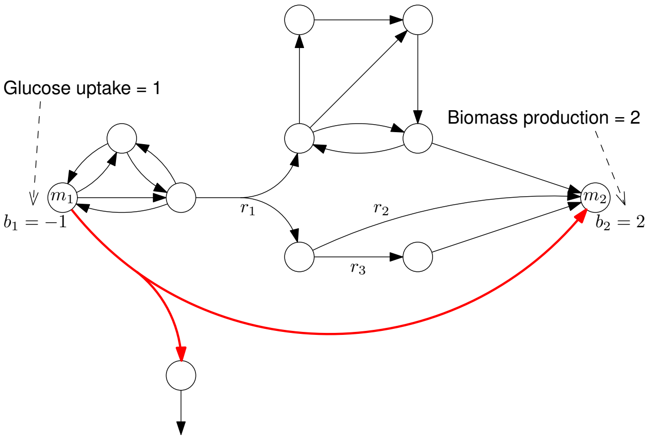
Running Example



- Flux Space: $\{v : Sv = b, v \geq 0\}$
- Optimize Biomass production (linear programming)

$$\max v_{biomass} \quad \text{subject to } Sv = 0,\ v_{glucose} = 1$$

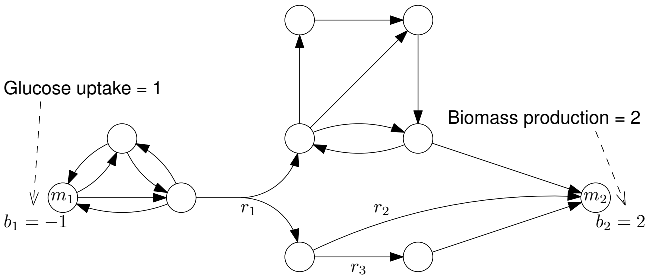
Running Example



- Flux Space: $\{v : Sv = b, v \geq 0\}$
- Optimize Biomass production (linear programming)

$$\max v_{biomass} \quad \text{subject to } Sv = 0, \; v_{glucose} = 1$$

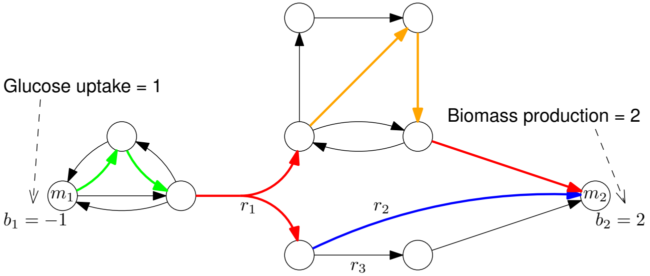
Running Example



- Flux Space: $\{v : Sv = b, v \geq 0\}$
- Optimize Biomass production (linear programming)

$$\max v_{biomass} \quad \text{subject to } Sv = 0,\ v_{glucose} = 1$$

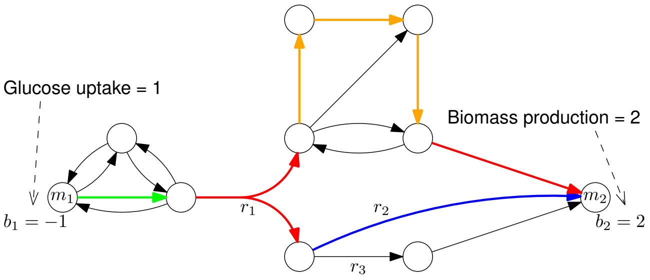
Running Example



- Flux Space: $\{v : Sv = b, v \geq 0\}$
- Optimize Biomass production (linear programming)

$$\max v_{biomass} \quad \text{subject to } Sv = 0,\ v_{glucose} = 1$$

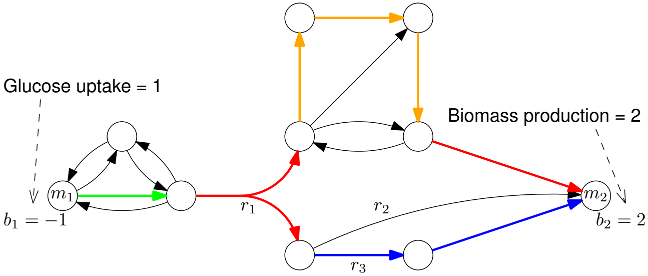
Running Example



- Flux Space: $\{v : Sv = b, v \geq 0\}$
- Optimize Biomass production (linear programming)

$$\max v_{biomass} \quad \text{subject to } Sv = 0, \; v_{glucose} = 1$$

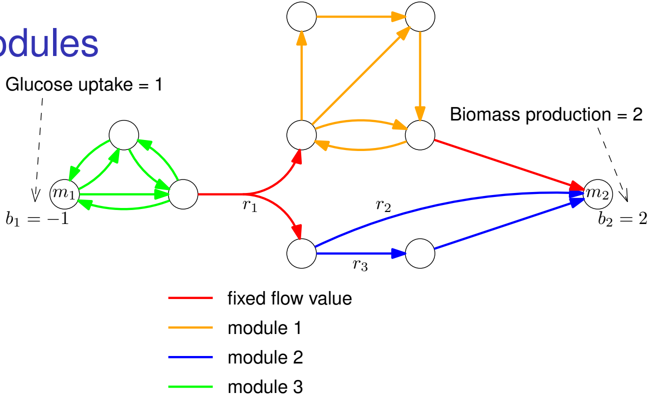
Running Example



- Flux Space: $\{v : Sv = b, v \geq 0\}$
- Optimize Biomass production (linear programming)

$$\max v_{biomass} \quad \text{subject to } Sv = 0, \; v_{glucose} = 1$$

Flux-Modules



Observation (Kelk, Olivier, S., Bruggeman ’12)

Reaction rates in the green module are independent from reaction rates in the orange module are independent from reaction rates in the blue module.

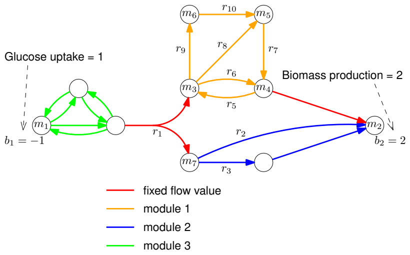
Kelk et al., Optimal flux spaces of genome-scale stoichiometric models are determined by a few subnetworks *Nature Scientific Reports*, 2:580, 2012.

Flux-Modules: A Definition

Notation:

- $\mathcal{N}$ reactions (variables, columns)
- $\mathcal{M}$ metabolites (constraints, rows)
- S stoichiometric matrix
- $P \subseteq \mathbb{R}^{\mathcal{N}}$ flux space: In Ex.

$\{v : Sv = 0, v_{glucose} = 1, v_{biomass} = 2, v \geq 0\}$



Definition (Reimers '13)

$A \subseteq \mathcal{N}$ is a P-module if $\exists d \in \mathbb{R}^{\mathcal{M}}$ s.t. $S_A v_A = d$ for all $v \in P$.

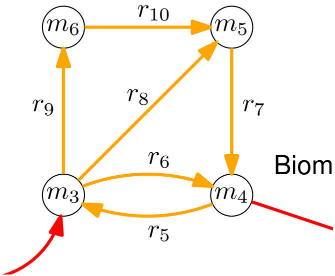
- A module is a set of reactions A.
- The interface flux $d = S_A v_A$ of a module is constant for all feasible flux vectors ($v \in P$).

Flux-Modules: A Definition

Notation:

- $\mathcal{N}$ reactions (variables, columns)
- $\mathcal{M}$ metabolites (constraints, rows)
- S stoichiometric matrix
- $P \subseteq \mathbb{R}^{\mathcal{N}}$ flux space: In Ex.

$\{v : Sv = 0, v_{glucose} = 1, v_{biomass} = 2, v \geq 0\}$



$d_3 = -1, \; d_4 = 1,$
$d_i = 0 \; \forall i \neq 3, 4$

Definition (Reimers '13)

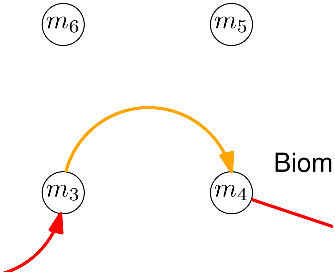
$A \subseteq \mathcal{N}$ is a P-module if $\exists d \in \mathbb{R}^{\mathcal{M}}$ s.t. $S_A v_A = d$ for all $v \in P$.

- A module is a set of reactions A.
- The interface flux $d = S_A v_A$ of a module is constant for all feasible flux vectors ($v \in P$).

Flux-Modules: A Definition

Notation:

- $\mathcal{N}$ reactions (variables, columns)
- $\mathcal{M}$ metabolites (constraints, rows)
- S stoichiometric matrix
- $P \subseteq \mathbb{R}^{\mathcal{N}}$ flux space: In Ex.
  $\{v : Sv = 0, v_{glucose} = 1, v_{biomass} = 2, v \geq 0\}$



$d_3 = -1,\ d_4 = 1,$
$d_i = 0\ \forall i \neq 3, 4$

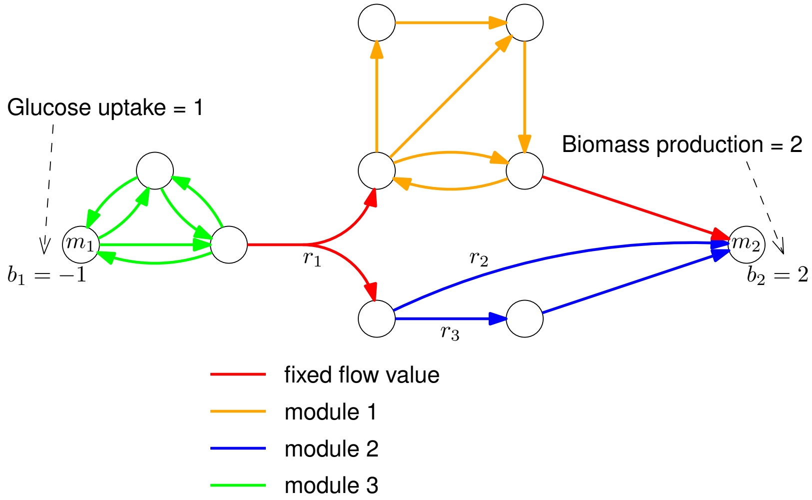
Definition (Reimers '13)

$A \subseteq \mathcal{N}$ is a P-module if $\exists d \in \mathbb{R}^{\mathcal{M}}$ s.t. $S_A v_A = d$ for all $v \in P$.

- A module is a set of reactions A.
- The interface flux $d = S_A v_A$ of a module is constant for all feasible flux vectors ($v \in P$).

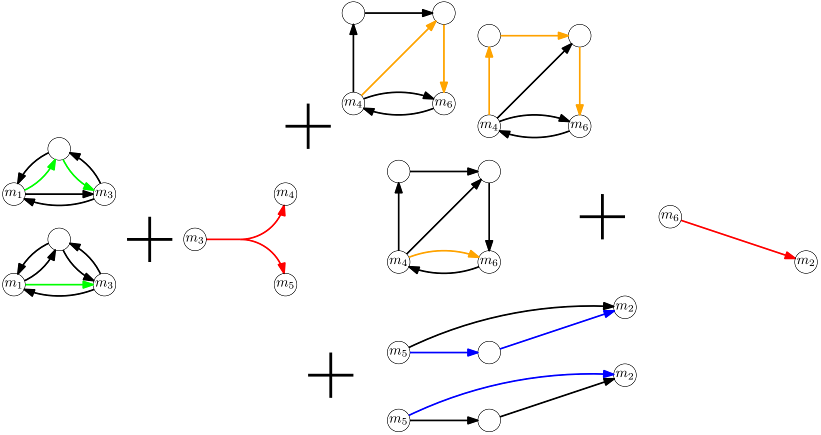
Decomposition of Elementary Flux Modes (EFM)

A graphical visualization of all 12 EFMs in the example network



Decomposition of Elementary Flux Modes (EFM)

A graphical visualization of all 12 EFMs in the example network



Outline

k-modules

Definition (module)

$A \subseteq \mathcal{N}$ is a module of P if there exists a $d \in \mathbb{R}^{\mathcal{M}}$, s.t. for all $x \in P$

$$S_A x_A = d \qquad .$$

k-modules

Definition (k-module)

$A \subseteq \mathcal{N}$ is a k-module of P if there exists a $d \in \mathbb{R}^{\mathcal{M}}$, $D \in \mathbb{R}^{\mathcal{M} \times k}$ s.t. for all $x \in P$ exists a $\alpha \in \mathbb{R}^k$ with

$$S_A x_A = d + D\alpha.$$

k-modules

Definition (k-module)

$A \subseteq \mathcal{N}$ is a k-module of P if there exists a $d \in \mathbb{R}^{\mathcal{M}}$, $D \in \mathbb{R}^{\mathcal{M} \times k}$ s.t. for all $x \in P$ exists a $\alpha \in \mathbb{R}^k$ with

$$S_A x_A = d + D\alpha.$$

For $P \subset \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b\}$ we have

k-modules

Definition (k-module)

$A \subseteq \mathcal{N}$ is a k-module of P if there exists a $d \in \mathbb{R}^{\mathcal{M}}$, $D \in \mathbb{R}^{\mathcal{M} \times k}$ s.t. for all $x \in P$ exists a $\alpha \in \mathbb{R}^k$ with

$$S_A x_A = d + D\alpha.$$

For $P \subset \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b\}$ we have

1) Flux module is a 0-module

k-modules

Definition (k-module)

$A \subseteq \mathcal{N}$ is a k-module of P if there exists a $d \in \mathbb{R}^{\mathcal{M}}$, $D \in \mathbb{R}^{\mathcal{M} \times k}$ s.t. for all $x \in P$ exists a $\alpha \in \mathbb{R}^k$ with

$$S_A x_A = d + D\alpha.$$

For $P \subset \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b\}$ we have

1) Flux module is a 0-module
2) Every set with k elements is a k-module

k-modules

Definition (k-module)

$A \subseteq \mathcal{N}$ is a k-module of P if there exists a $d \in \mathbb{R}^{\mathcal{M}}$, $D \in \mathbb{R}^{\mathcal{M} \times k}$ s.t. for all $x \in P$ exists a $\alpha \in \mathbb{R}^k$ with

$$S_A x_A = d + D\alpha.$$

For $P \subset \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b\}$ we have

1) Flux module is a 0-module
2) Every set with k elements is a k-module
3) A is a $k-1$-module $\Rightarrow$ A is a k-module

k-modules

Definition (k-module)

$A \subseteq \mathcal{N}$ is a k-module of P if there exists a $d \in \mathbb{R}^{\mathcal{M}}$, $D \in \mathbb{R}^{\mathcal{M} \times k}$ s.t. for all $x \in P$ exists a $\alpha \in \mathbb{R}^k$ with

$$S_A x_A = d + D\alpha.$$

For $P \subset \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b\}$ we have

1) Flux module is a 0-module
2) Every set with k elements is a k-module
3) A is a $k-1$-module $\Rightarrow$ A is a k-module
5) A is a k-module if and only if $\mathcal{N} \setminus A$ is a k-module

k-modules

Definition (k-module)

$A \subseteq \mathcal{N}$ is a k-module of P if there exists a $d \in \mathbb{R}^{\mathcal{M}}$, $D \in \mathbb{R}^{\mathcal{M} \times k}$ s.t. for all $x \in P$ exists a $\alpha \in \mathbb{R}^k$ with

$$S_A x_A = d + D\alpha.$$

For $P \subset \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b\}$ we have

1) Flux module is a 0-module
2) Every set with k elements is a k-module
3) A is a $k-1$-module $\Rightarrow$ A is a k-module
5) A is a k-module if and only if $\mathcal{N} \setminus A$ is a k-module
6) $\emptyset$ is a k-module for all k

k-modules

Definition (k-module)

$A \subseteq \mathcal{N}$ is a k-module of P if there exists a $d \in \mathbb{R}^{\mathcal{M}}$, $D \in \mathbb{R}^{\mathcal{M} \times k}$ s.t. for all $v \in P$ exists a $\alpha \in \mathbb{R}^k$ with

$$S_A v_A = d + D\alpha.$$

If $d = 0$ we call A a linear k-module

k-modules

Definition (*k*-module)

$A \subseteq \mathcal{N}$ is a k-module of P if there exists a $d \in \mathbb{R}^{\mathcal{M}}$, $D \in \mathbb{R}^{\mathcal{M} \times k}$ s.t. for all $v \in P$ exists a $\alpha \in \mathbb{R}^k$ with

$$S_A v_A = d + D\alpha.$$

Notice that D is not unique but $\langle D \rangle$, the span of D is unique.

k-modules

Definition (*k*-module)

$A \subseteq \mathcal{N}$ is a k-module of P if there exists a $d \in \mathbb{R}^{\mathcal{M}}$, $D \in \mathbb{R}^{\mathcal{M} \times k}$ s.t. for all $v \in P$ exists a $\alpha \in \mathbb{R}^k$ with

$$S_A v_A = d + D\alpha.$$

Notice that D is not unique but $\langle D \rangle$, the span of D is unique.
Allow decomposition of general polyhedra

k-modules

Definition (*k*-module)

$A \subseteq \mathcal{N}$ is a k-module of P if there exists a $d \in \mathbb{R}^{\mathcal{M}}$, $D \in \mathbb{R}^{\mathcal{M} \times k}$ s.t. for all $v \in P$ exists a $\alpha \in \mathbb{R}^k$ with

$$S_A v_A = d + D\alpha.$$

Notice that D is not unique but $\langle D \rangle$, the span of D is unique.
Allow decomposition of general polyhedra

- k-modules relate to k-separations of the matroid with ground set the columns of the constraint matrix

k-modules on linear vector spaces

$$P \subset \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b\}$$

k-modules on linear vector spaces

$P \subset \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b\}$

Let ker(S)$= \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = 0\}$, the kernel of S

Theorem

A is a k-module for $P \Leftrightarrow A$ is a k-module for ker(S)

k-modules on linear vector spaces

$P \subset \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = b\}$

Let ker(S)$= \{x \in \mathbb{R}^{\mathcal{N}} \mid Sx = 0\}$, the kernel of S

Theorem

A is a k-module for P $\Leftrightarrow$ A is a k-module for ker(S)

Under the condition that no x_i has fixed value on P
which can be ensured by preprocessing if necessary.

k-modules and Matroids

Definition (*k*-separator, Oxley 2011)

Let M be a matroid on the element set $\mathcal{N}$. A set $A \subseteq \mathcal{N}$ is a k-separator if and only if

$$\mathrm{rank}(A) + \mathrm{rank}(\mathcal{N} \setminus A) - \mathrm{rank}(\mathcal{N}) < k.$$

k-modules and Matroids

Definition (k-separator, Oxley 2011)

Let M be a matroid on the element set $\mathcal{N}$. A set $A \subseteq \mathcal{N}$ is a k-separator if and only if

$$\text{rank}(A) + \text{rank}(\mathcal{N} \setminus A) - \text{rank}(\mathcal{N}) < k.$$

Separation is a measure of connectivity of the matroid

k-modules and Matroids

Definition (*k*-separator, Oxley 2011)

Let M be a matroid on the element set $\mathcal{N}$. A set $A \subseteq \mathcal{N}$ is a *k*-separator if and only if

$$\mathrm{rank}(A) + \mathrm{rank}(\mathcal{N} \setminus A) - \mathrm{rank}(\mathcal{N}) < k.$$

k-modules and Matroids

Definition (k-separator, Oxley 2011)

Let M be a matroid on the element set $\mathcal{N}$. A set $A \subseteq \mathcal{N}$ is a k-separator if and only if

$$\mathrm{rank}(A) + \mathrm{rank}(\mathcal{N} \setminus A) - \mathrm{rank}(\mathcal{N}) < k.$$

Theorem

$A \subseteq \mathcal{N}$ is a $(\ker S)$-k-module if and only if A is a $k+1$-separator in the linear matroid M represented by S.

Branch Decomposition of columns of S

Definition (branch-width)

- A *branch decomposition* (T, τ) consists of a tree T with nodes of degree 3 and 1 and a bijective map $\tau : T \to \mathcal{N}$. Define $\tau(A) := \{\tau(a) : a \in A\}$.
- The *width* of edge e of T is $\rho(\tau(A_e))$, where (A_e, B_e) the partition of the leaves of T given by $T \setminus e$. (Note that $\rho(A) = \rho(\mathcal{N} \setminus A)$.)
- The width of a branch decomposition is the maximal width of an edge $e \in T$.
- The *branch-width* of M is the minimal width of all possible branch-decompositions.

With connectivity function

$$\rho(A) := \mathrm{rank}(A) + \mathrm{rank}(\mathcal{N} \setminus A) - \mathrm{rank}(\mathcal{N}) + 1$$

Branch Decomposition of columns of S

Definition (branch-width)

- A *branch decomposition* (T, τ) consists of a tree T with nodes of degree 3 and 1 and a bijective map $\tau : T \to \mathcal{N}$. Define $\tau(A) := \{\tau(a) : a \in A\}$.
- The *width* of edge e of T is $\rho(\tau(A_e))$, where (A_e, B_e) the partition of the leaves of T given by $T \setminus e$. (Note that $\rho(A) = \rho(\mathcal{N} \setminus A)$.)
- The width of a branch decomposition is the maximal width of an edge $e \in T$.
- The *branch-width* of M is the minimal width of all possible branch-decompositions.

With connectivity function

$$\rho(A) := \mathrm{rank}(A) + \mathrm{rank}(\mathcal{N} \setminus A) - \mathrm{rank}(\mathcal{N}) + 1$$

Note: $\rho(A) \leq k + 1 \Leftrightarrow A$ is $k + 1$-separator $\Leftrightarrow A$ is k-module for P

Rooted Decomposition

Suppose a branch decomposition of $\mathcal{N}$ of width $k + 1$;
Create a hierarchical family $\texttt{Mod}$ of k-modules

Rooted Decomposition

Suppose a branch decomposition of $\mathcal{N}$ of width $k+1$;
Create a hierarchical family `Mod` of *k*-modules

- Take an arbitrary edge of the branch decomposition,

Rooted Decomposition

Suppose a branch decomposition of $\mathcal{N}$ of width $k+1$;
Create a hierarchical family `Mod` of k-modules

- Take an arbitrary edge of the branch decomposition,
- subdivide it and make the created vertex the root corresponding to $\mathcal{N}$,

Rooted Decomposition

Suppose a branch decomposition of $\mathcal{N}$ of width $k+1$;
Create a hierarchical family `Mod` of *k*-modules

- Take an arbitrary edge of the branch decomposition,
- subdivide it and make the created vertex the root corresponding to $\mathcal{N}$,
- Direct all other edges away from the root.

Rooted Decomposition

Suppose a branch decomposition of $\mathcal{N}$ of width $k+1$;
Create a hierarchical family `Mod` of k-modules

- Take an arbitrary edge of the branch decomposition,
- subdivide it and make the created vertex the root corresponding to $\mathcal{N}$,
- Direct all other edges away from the root.

Every internal node of this rooted binary tree corresponds to the set of elements in `Mod` formed by the leaves in the subtree below it.

Rooted Decomposition

Suppose a branch decomposition of $\mathcal{N}$ of width $k+1$;
Create a hierarchical family $\mathtt{Mod}$ of k-modules

- Take an arbitrary edge of the branch decomposition,
- subdivide it and make the created vertex the root corresponding to $\mathcal{N}$,
- Direct all other edges away from the root.

Every internal node of this rooted binary tree corresponds to the set of elements in $\mathtt{Mod}$ formed by the leaves in the subtree below it.

The hierarchic family $\mathtt{Mod}$ of subsets of $\mathcal{N}$ thus created has property: If $C \in \mathtt{Mod}$ then $\exists A, B \in \mathtt{Mod}$, $A \cap B = \emptyset$, $C = A \cup B$.

Vertex enumeration

For module $A \in \texttt{Mod}$ let

$$P^A := \{x \in \mathbb{R}^A : S_A x = D^A \alpha + d, x \geq 0, \exists \alpha \in \mathbb{R}^k\}.$$

Vertex enumeration

For module $A \in \texttt{Mod}$ let

$$P^A := \{x \in \mathbb{R}^A : S_A x = D^A \alpha + d, x \geq 0, \exists \alpha \in \mathbb{R}^k\}.$$

Idea: For $C = A \cup B$ combine vertices of P^A with vertices of P^B into vertices of P^C.

Vertex enumeration

For module $A \in \mathtt{Mod}$ let

$$P^A := \{x \in \mathbb{R}^A : S_A x = D^A \alpha + d, x \geq 0, \exists \alpha \in \mathbb{R}^k\}.$$

Idea: For $C = A \cup B$ combine vertices of P^A with vertices of P^B into vertices of P^C.

Cartesian product only works if A and B are 0-modules (1-separators)! (restricted to C)

Vertex enumeration

For module $A \in \texttt{Mod}$ let

$$P^A := \{x \in \mathbb{R}^A : S_A x = D^A \alpha + d, x \geq 0, \exists \alpha \in \mathbb{R}^k\}.$$

Idea: For $C = A \cup B$ combine vertices of P^A with vertices of P^B into vertices of P^C.
Cartesian product only works if A and B are 0-modules (1-separators)! (restricted to C)

Every face of P is determined uniquely by $Sx = b$ and a subset of the non-negativity restrictions being tight: for some $F \in \mathcal{N}$ $x_F = 0$

Vertex enumeration

For module $A \in \texttt{Mod}$ let

$$P^A := \{x \in \mathbb{R}^A : S_A x = D^A \alpha + d, x \geq 0, \exists \alpha \in \mathbb{R}^k\}.$$

Idea: For $C = A \cup B$ combine vertices of P^A with vertices of P^B into vertices of P^C.
Cartesian product only works if A and B are 0-modules (1-separators)! (restricted to C)

Every face of P is determined uniquely by $Sx = b$ and a subset of the non-negativity restrictions being tight: for some $F \in \mathcal{N}$ $x_F = 0$ a face of P

Vertex enumeration

For module $A \in \texttt{Mod}$ let

$$P^A := \{x \in \mathbb{R}^A : S_A x = D^A \alpha + d, x \geq 0, \exists \alpha \in \mathbb{R}^k\}.$$

Idea: For $C = A \cup B$ combine vertices of P^A with vertices of P^B into vertices of P^C.
Cartesian product only works if A and B are 0-modules (1-separators)! (restricted to C)

Every face of P is determined uniquely by $Sx = b$ and a subset of the non-negativity restrictions being tight: for some $F \in \mathcal{N}$ $x_F = 0$ a face of P

Definition (A-cface (combinatorial represent. of A-face))

For $A \subseteq \mathcal{N}$ a set $F \subseteq A$ is called a *A-cface* if there exists a $x \in P$ with $x_F = 0$ and $x_{A \setminus F} > 0$.

Vertex enumeration

Definition (A-cface)

For $A \subseteq \mathcal{N}$ a set $F \subseteq A$ is called a *A-cface* if there exists a $x \in P$ with $x_F = 0$ and $x_{A \setminus F} > 0$.

In fact we are interested in enumerating

Definition (vertex inducing A-cface)

For $A \subseteq \mathcal{N}$ a set $F \subseteq A$ is called *vertex inducing A-cface* if there exists a vertex v of P with $v_F = 0$ and $v_{A \setminus F} > 0$.

Vertex enumeration

Definition (A-cface)

For $A \subseteq \mathcal{N}$ a set $F \subseteq A$ is called a *A-cface* if there exists a $x \in P$ with $x_F = 0$ and $x_{A \setminus F} > 0$.

In fact we are interested in enumerating

Definition (vertex inducing A-cface)

For $A \subseteq \mathcal{N}$ a set $F \subseteq A$ is called *vertex inducing A-cface* if there exists a vertex v of P with $v_F = 0$ and $v_{A \setminus F} > 0$.

Testing if $F \subset A$ is vertex-inducing is NP-hard [Fukuda EA 97]

Vertex enumeration

Definition (A-cface)

For $A \subseteq \mathcal{N}$ a set $F \subseteq A$ is called a *A-cface* if there exists a $x \in P$ with $x_F = 0$ and $x_{A \setminus F} > 0$.

In fact we are interested in enumerating

Definition (vertex inducing A-cface)

For $A \subseteq \mathcal{N}$ a set $F \subseteq A$ is called *vertex inducing A-cface* if there exists a vertex v of P with $v_F = 0$ and $v_{A \setminus F} > 0$.

Testing if $F \subset A$ is vertex-inducing is NP-hard [Fukuda EA 97]

Definition (injective A-cface)

For $A \subseteq \mathcal{N}$ set $F \subseteq A$ is injective A-cface if $\nexists y, z \in P^A : y_F = z_F = 0$ with $S_A y = S_A z$; i.e. S_A injective on $\{x \in P^A : x_F = 0\}$.

Vertex enumeration

Definition (A-cface)

For $A \subseteq \mathcal{N}$ a set $F \subseteq A$ is called a *A-cface* if there exists a $x \in P$ with $x_F = 0$ and $x_{A \setminus F} > 0$.

In fact we are interested in enumerating

Definition (vertex inducing A-cface)

For $A \subseteq \mathcal{N}$ a set $F \subseteq A$ is called *vertex inducing A-cface* if there exists a vertex v of P with $v_F = 0$ and $v_{A \setminus F} > 0$.

Testing if $F \subset A$ is vertex-inducing is NP-hard [Fukuda EA 97]

Definition (injective A-cface)

For $A \subseteq \mathcal{N}$ set $F \subseteq A$ is injective A-cface if $\nexists y, z \in P^A : y_F = z_F = 0$ with $S_A y = S_A z$; i.e. S_A injective on $\{x \in P^A : x_F = 0\}$.

Testing if $F \subset A$ is injective can be done in polytime

Injective cFase Enumeration

Recursively for every $C = A \dot{\cup} B$.
Let $\mathcal{F}^A :=$ Injective A-cFaces
Let $\mathcal{F}^B :=$ Injective B-cFaces

Construct

$$\mathcal{F} := \{F^A \cup F^B : F^A \in \mathcal{F}^A, F^B \in \mathcal{F}^B\}$$

For every $F \in \mathcal{F}$ test if it is a cface and injective for C.
If not delete F from $\mathcal{F}$

Set the resulting set $\mathcal{F}$ to $\mathcal{F}^C$

Vertex enumeration

We have enumerated all injective $\mathcal{N}$-cfaces, whereas we should have enumerated all vertex-induced $\mathcal{N}$-cfaces :-(

Vertex enumeration

We have enumerated all injective $\mathcal{N}$-cfaces, whereas we should have enumerated all vertex-induced $\mathcal{N}$-cfaces :-(

Theorem

Let $A \in \texttt{Mod}$ be a 0-module. Then, a A-cface is injective if and only if it is vertex inducing.

Vertex enumeration

We have enumerated all injective $\mathcal{N}$-cfaces, whereas we should have enumerated all vertex-induced $\mathcal{N}$-cfaces :-(

Theorem

Let $A \in \texttt{Mod}$ be a 0-module. Then, a A-cface is injective if and only if it is vertex inducing.

Corollary

The injective $\mathcal{N}$-cfaces are the vertices of P and Injective cFace Enumeration Algorithm applied to $C = \mathcal{N}$ computes all vertices of P.

Vertex enumeration

We have enumerated all injective $\mathcal{N}$-cfaces, whereas we should have enumerated all vertex-induced $\mathcal{N}$-cfaces :-(

Theorem

Let $A \in \texttt{Mod}$ be a 0-module. Then, a A-cface is injective if and only if it is vertex inducing.

Corollary

The injective $\mathcal{N}$-cfaces are the vertices of P and Injective cFace Enumeration Algorithm applied to $C = \mathcal{N}$ computes all vertices of P.

Did we not enumerate too many minimal faces on the way?

Vertex enumeration in total polytime

Proposition

Every injective and vertex inducing A-cface F for $A \in \texttt{Mod}$ *has* $\dim\{x \in P^A : x_F = 0\} \leq k$.

Vertex enumeration in total polytime

Proposition

Every injective and vertex inducing A-cface F for $A \in \texttt{Mod}$ *has* $\dim\{x \in P^A : x_F = 0\} \leq k$.

Assume P is bounded.

Vertex enumeration in total polytime

Proposition

Every injective and vertex inducing A-cface F for $A \in \mathtt{Mod}$ *has* $\dim\{x \in P^A : x_F = 0\} \leq k$.

Assume P is bounded.

Essentially Caratheodory:

Vertex enumeration in total polytime

Proposition

Every injective and vertex inducing A-cface F for $A \in \texttt{Mod}$ *has* $\dim\{x \in P^A : x_F = 0\} \leq k$.

Assume P is bounded.

Essentially Caratheodory:

Lemma

For A-cface F, $A \in \texttt{Mod}$*, there exist* $\ell \leq k+1$ *vertex inducing A-cfaces* $F^1, \ldots, F^\ell$ *such that* $F = F^1 \cap F^2 \cap \ldots \cap F^\ell$.

Vertex enumeration in total polytime

Proposition

Every injective and vertex inducing A-cface F for $A \in \texttt{Mod}$ has $\dim\{x \in P^A : x_F = 0\} \leq k$.

Assume P is bounded.

Lemma

For A-cface F, $A \in \texttt{Mod}$, there exist $\ell \leq k+1$ vertex inducing A-cfaces $F^1, \ldots, F^\ell$ such that $F = F^1 \cap F^2 \cap \ldots \cap F^\ell$.

Proposition

If P is bounded then holds for all $A \in \texttt{Mod}$ that

$$|\{F \subseteq A : F \text{ injective } A\text{-cface}\}| \leq$$
$$|\{F \subseteq A : F \text{ vertex inducing } A\text{-cface}\}|^{k+1}.$$

Vertex enumeration in total polytime

Proposition

If P is bounded then holds for all $A \in \texttt{Mod}$ that

$$|\{F \subseteq A : F \text{ injective } A\text{-cface}\}| \leq |\{F \subseteq A : F \text{ vertex inducing } A\text{-cface}\}|^{k+1}.$$

Vertex enumeration in total polytime

Proposition

If P is bounded then holds for all $A \in \mathtt{Mod}$ that

$$|\{F \subseteq A : F \text{ injective } A\text{-cface}\}| \leq |\{F \subseteq A : F \text{ vertex inducing } A\text{-cface}\}|^{k+1}.$$

Proposition

For $A \in \mathtt{Mod}$ holds that

$$|\{F \subseteq A : F \text{ vertex inducing } A\text{-cface}\}| \leq |\mathcal{V} =: \{v \in \mathbb{R}^{\mathcal{N}} : v \text{ is a vertex of } P\}|.$$

Vertex enumeration in total polytime

$$|\{F \subseteq A : F \text{ injective } A\text{-cface}\}| \leq |\{F \subseteq A : F \text{ vertex induced } A\text{-cface}\}|^{k+1}.$$

$$|\{F \subseteq A : F \text{ vertex induced } A\text{-cface}\}| \leq |\mathcal{V} =: \{v \in \mathbb{R}^{\mathcal{N}} : v \text{ is a vertex of } P\}|.$$

Vertex enumeration in total polytime

$$|\{F \subseteq A : F \text{ injective } A\text{-cface}\}| \leq |\{F \subseteq A : F \text{ vertex induced } A\text{-cface}\}|^{k+1}.$$

$$|\{F \subseteq A : F \text{ vertex induced } A\text{-cface}\}| \leq |\mathcal{V} =: \{v \in \mathbb{R}^{\mathcal{N}} : v \text{ is a vertex of } P\}|.$$

Let t time to test for injectiveness and the cface property.

Theorem

For P bounded and $A, B, C \in \texttt{Mod}$ with $C = A \dot{\cup} B$. Given the set of injective A-cfaces $\mathcal{F}^A$ and the set of injective B-cfaces $\mathcal{F}^B$, the set of injective C-cfaces $\mathcal{F}^C$ can be computed in time $O\left(|\mathcal{V}|^{2k+2} t\right)$,

Vertex enumeration in total polytime

Theorem

For P bounded and $A, B, C \in \mathtt{Mod}$ with $C = A \dot{\cup} B$. Given the set of injective A-cfaces $\mathcal{F}^A$ and the set of injective B-cfaces $\mathcal{F}^B$, the set of injective C-cfaces $\mathcal{F}^C$ can be computed in time $O\left(|\mathcal{V}|^{2k+2} t\right)$,

Vertex enumeration in total polytime

Theorem

For P bounded and $A, B, C \in \texttt{Mod}$ with $C = A \dot{\cup} B$. Given the set of injective A-cfaces $\mathcal{F}^A$ and the set of injective B-cfaces $\mathcal{F}^B$, the set of injective C-cfaces $\mathcal{F}^C$ can be computed in time $O\left(|\mathcal{V}|^{2k+2} t\right)$,

Since there are $O(\mathcal{N})$ internal nodes in the binary tree defining $\texttt{Mod}$, we have

Theorem

For P bounded its set of vertices can be computed in time $O\left(\mathcal{N}|\mathcal{V}|^{2k+2} t\right)$,

The power of *k*-modularity?

- Positive

The power of *k*-modularity?

- Positive
  - Vertex enumeration in total polynomial time
  - ILP with positive constraint matrix is FPT if $\max_i b_i$ is fixed.

The power of k-modularity?

- Positive
  - Vertex enumeration in total polynomial time
  - ILP with positive constraint matrix is FPT if $\max_i b_i$ is fixed.
  - LP can be solved in strongly polynomial time if the feasible polyhedron is 1-modular

The power of k-modularity?

- Positive
  - Vertex enumeration in total polynomial time
  - ILP with positive constraint matrix is FPT if $\max_i b_i$ is fixed.
  - LP can be solved in strongly polynomial time if the feasible polyhedron is 1-modular
- Negative

The power of k-modularity?

- Positive
  - Vertex enumeration in total polynomial time
  - ILP with positive constraint matrix is FPT if $\max_i b_i$ is fixed.
  - LP can be solved in strongly polynomial time if the feasible polyhedron is 1-modular
- Negative
  - General ILP remains NP-hard even for 3-modular polyhedra

The power of k-modularity?

- Positive
  - Vertex enumeration in total polynomial time
  - ILP with positive constraint matrix is FPT if $\max_i b_i$ is fixed.
  - LP can be solved in strongly polynomial time if the feasible polyhedron is 1-modular
- Negative
  - General ILP remains NP-hard even for 3-modular polyhedra
  - Deciding if a vertex exists that contains two variables in its support remains NP-hard even for 3-modular polyhedra

The power of *k*-modularity?

- Positive
  - Vertex enumeration in total polynomial time
  - ILP with positive constraint matrix is FPT if $\max_i b_i$ is fixed.
  - LP can be solved in strongly polynomial time if the feasible polyhedron is 1-modular
- Negative
  - General ILP remains NP-hard even for 3-modular polyhedra
  - Deciding if a vertex exists that contains two variables in its support remains NP-hard even for 3-modular polyhedra
- Open

The power of *k*-modularity?

- Positive
  - Vertex enumeration in total polynomial time
  - ILP with positive constraint matrix is FPT if $\max_i b_i$ is fixed.
  - LP can be solved in strongly polynomial time if the feasible polyhedron is 1-modular
- Negative
  - General ILP remains NP-hard even for 3-modular polyhedra
  - Deciding if a vertex exists that contains two variables in its support remains NP-hard even for 3-modular polyhedra
- Open
  - Vertex enumeration in unbounded *k*-modular polyhedra?

The power of *k*-modularity?

- Positive
  - Vertex enumeration in total polynomial time
  - ILP with positive constraint matrix is FPT if $\max_i b_i$ is fixed.
  - LP can be solved in strongly polynomial time if the feasible polyhedron is 1-modular
- Negative
  - General ILP remains NP-hard even for 3-modular polyhedra
  - Deciding if a vertex exists that contains two variables in its support remains NP-hard even for 3-modular polyhedra
- Open
  - Vertex enumeration in unbounded *k*-modular polyhedra?
  - Can LP be solved in strongly polynomial time with *k*-modular polyhedra? Or would this imply that general LP can be solved in strongly polynomial time?

The power of k-modularity?

- Positive
  - Vertex enumeration in total polynomial time
  - ILP with positive constraint matrix is FPT if $\max_i b_i$ is fixed.
  - LP can be solved in strongly polynomial time if the feasible polyhedron is 1-modular
- Negative
  - General ILP remains NP-hard even for 3-modular polyhedra
  - Deciding if a vertex exists that contains two variables in its support remains NP-hard even for 3-modular polyhedra
- Open
  - Vertex enumeration in unbounded k-modular polyhedra?
  - Can LP be solved in strongly polynomial time with k-modular polyhedra? Or would this imply that general LP can be solved in strongly polynomial time?
  - Can the vertex with minimum size support of a k-modular polyhedron be found in polynomial time?