Polytope Propagation

Michael Joswig

Fachbereich Mathematik
Technische Universität Darmstadt

May 9, 2005

Inhalt

Inhalt

Inhalt

The Setup

Consider finite directed graph $\Gamma = (V, A)$ without directed cycles and some function $\alpha : A \to \mathbb{R}^d$.

- assume a unique source $q \in V$ and a unique sink $s \in V$

Inductive construction:

- for each node $v \in V$ define polytope $P_v \subset \mathbb{R}^d$
- $P_q = 0$
- if v has the direct predecessors $u_1, u_2, \ldots, u_n$ then

$$P_v = \operatorname{conv}(P_{u_1} + \alpha(u_1, v), \ldots, P_{u_n} + \alpha(u_n, v))$$

- $P(\Gamma, \alpha) = P_s$ is the polytope *propagated* by (Γ, α)

$\to$ Pachter & Sturmfels, 2004 & ASCB

The Setup

Consider finite directed graph $\Gamma = (V, A)$ without directed cycles and some function $\alpha : A \to \mathbb{R}^d$.

- assume a unique source $q \in V$ and a unique sink $s \in V$

Inductive construction:

- for each node $v \in V$ define polytope $P_v \subset \mathbb{R}^d$
- $P_q = 0$
- if v has the direct predecessors $u_1, u_2, \ldots, u_n$ then

$$P_v = \mathrm{conv}(P_{u_1} + \alpha(u_1, v), \ldots, P_{u_n} + \alpha(u_n, v))$$

- $P(\Gamma, \alpha) = P_s$ is the polytope *propagated* by (Γ, α)

$\to$ Pachter & Sturmfels, 2004 & ASCB

The Setup

Consider finite directed graph $\Gamma = (V, A)$ without directed cycles and some function $\alpha : A \to \mathbb{R}^d$.

- assume a unique source $q \in V$ and a unique sink $s \in V$

Inductive construction:

- for each node $v \in V$ define polytope $P_v \subset \mathbb{R}^d$
- $P_q = 0$
- if v has the direct predecessors $u_1, u_2, \ldots, u_n$ then

$$P_v = \operatorname{conv}(P_{u_1} + \alpha(u_1, v), \ldots, P_{u_n} + \alpha(u_n, v))$$

- $P(\Gamma, \alpha) = P_s$ is the polytope *propagated* by (Γ, α)

$\rightarrow$ Pachter & Sturmfels, 2004 & ASCB

The Setup

Consider finite directed graph $\Gamma = (V, A)$ without directed cycles and some function $\alpha : A \to \mathbb{R}^d$.

- assume a unique source $q \in V$ and a unique sink $s \in V$

Inductive construction:

- for each node $v \in V$ define polytope $P_v \subset \mathbb{R}^d$
- $P_q = 0$
- if v has the direct predecessors $u_1, u_2, \ldots, u_n$ then

$$P_v = \operatorname{conv}(P_{u_1} + \alpha(u_1, v), \ldots, P_{u_n} + \alpha(u_n, v))$$

- $P(\Gamma, \alpha) = P_s$ is the polytope *propagated* by (Γ, α)

$\rightarrow$ Pachter & Sturmfels, 2004 & ASCB

The Setup

Consider finite directed graph $\Gamma = (V, A)$ without directed cycles and some function $\alpha : A \to \mathbb{R}^d$.

- assume a unique source $q \in V$ and a unique sink $s \in V$

Inductive construction:

- for each node $v \in V$ define polytope $P_v \subset \mathbb{R}^d$
- $P_q = 0$
- if v has the direct predecessors $u_1, u_2, \ldots, u_n$ then

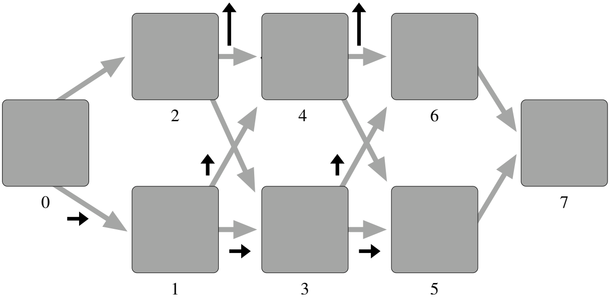
$$P_v = \operatorname{conv}(P_{u_1} + \alpha(u_1, v), \ldots, P_{u_n} + \alpha(u_n, v))$$

- $P(\Gamma, \alpha) = P_s$ is the polytope *propagated* by (Γ, α)

$\to$ Pachter & Sturmfels, 2004 & ASCB

An Easy Example

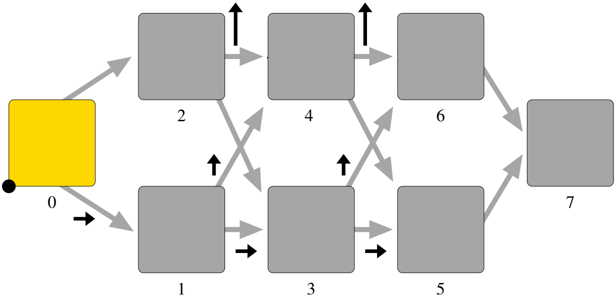
$$\Gamma = (\{0, 1, \ldots, 7\}, \ldots)$$
$$\alpha(0,1) = \alpha(1,3) = \alpha(3,5) = (1,0),$$
$$\alpha(1,4) = \alpha(3,6) = (0,1),\ \alpha(2,4) = \alpha(4,6) = (0,2),$$
$$\alpha(0,2) = \alpha(5,7) = \alpha(6,7) = 0$$



$$P(\Gamma, \alpha) = \operatorname{conv}((0,1), (0,4), (1,0), (1,3), (3,0))$$

An Easy Example

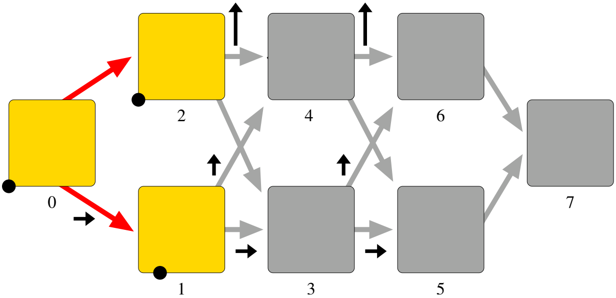
$$\Gamma = (\{0, 1, \ldots, 7\}, \ldots)$$
$$\alpha(0,1) = \alpha(1,3) = \alpha(3,5) = (1,0),$$
$$\alpha(1,4) = \alpha(3,6) = (0,1),\ \alpha(2,4) = \alpha(4,6) = (0,2),$$
$$\alpha(0,2) = \alpha(5,7) = \alpha(6,7) = 0$$



$$P(\Gamma, \alpha) = \mathrm{conv}((0,1), (0,4), (1,0), (1,3), (3,0))$$

An Easy Example

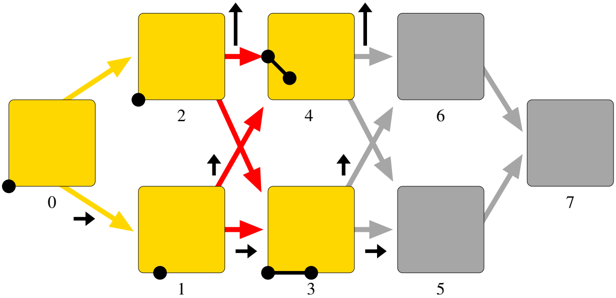
$$\Gamma = (\{0, 1, \ldots, 7\}, \ldots)$$
$$\alpha(0,1) = \alpha(1,3) = \alpha(3,5) = (1,0),$$
$$\alpha(1,4) = \alpha(3,6) = (0,1),\ \alpha(2,4) = \alpha(4,6) = (0,2),$$
$$\alpha(0,2) = \alpha(5,7) = \alpha(6,7) = 0$$



$$P(\Gamma, \alpha) = \mathrm{conv}((0,1), (0,4), (1,0), (1,3), (3,0))$$

An Easy Example

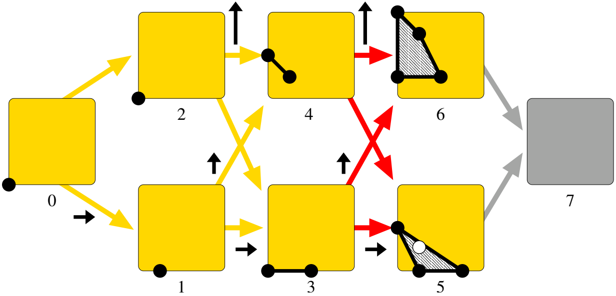
$$\Gamma = (\{0, 1, \ldots, 7\}, \ldots)$$
$$\alpha(0,1) = \alpha(1,3) = \alpha(3,5) = (1,0),$$
$$\alpha(1,4) = \alpha(3,6) = (0,1),\ \alpha(2,4) = \alpha(4,6) = (0,2),$$
$$\alpha(0,2) = \alpha(5,7) = \alpha(6,7) = 0$$



$$P(\Gamma, \alpha) = \mathrm{conv}((0,1),(0,4),(1,0),(1,3),(3,0))$$

An Easy Example

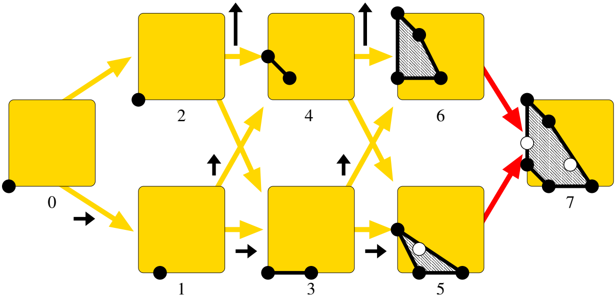
$$\Gamma = (\{0, 1, \ldots, 7\}, \ldots)$$
$$\alpha(0,1) = \alpha(1,3) = \alpha(3,5) = (1,0),$$
$$\alpha(1,4) = \alpha(3,6) = (0,1),\ \alpha(2,4) = \alpha(4,6) = (0,2),$$
$$\alpha(0,2) = \alpha(5,7) = \alpha(6,7) = 0$$



$$P(\Gamma, \alpha) = \mathrm{conv}((0,1), (0,4), (1,0), (1,3), (3,0))$$

An Easy Example

$$\Gamma = (\{0,1,\dots,7\},\dots)$$
$$\alpha(0,1)=\alpha(1,3)=\alpha(3,5)=(1,0),$$
$$\alpha(1,4)=\alpha(3,6)=(0,1),\ \alpha(2,4)=\alpha(4,6)=(0,2),$$
$$\alpha(0,2)=\alpha(5,7)=\alpha(6,7)=0$$



$$P(\Gamma,\alpha) = \mathrm{conv}((0,1),(0,4),(1,0),(1,3),(3,0))$$

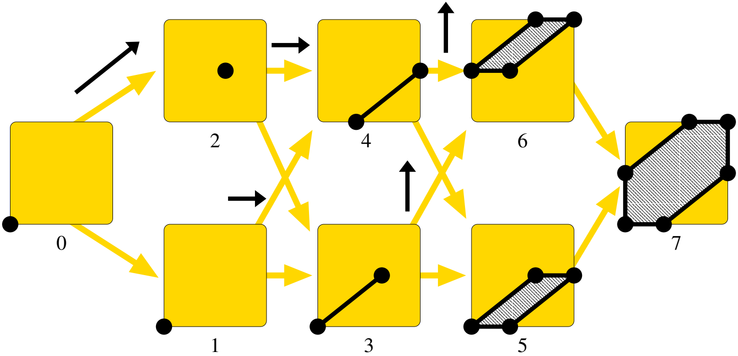
A Trivial Example

Let $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ be finite.

- Define a directed graph Θ with the $n+2$ nodes $\boxed{0, 1, \ldots, n+1}$ such that there are
  - arcs from $\boxed{0}$ to the nodes $\boxed{1, 2, \ldots, n}$ and
  - arcs from each of the nodes $\{$framebox$1, 2, \ldots, n$ to $\boxed{n+1}$.
- Further let $\theta(0, k) = x_k$ and $\theta(k, n+1) = 0$, for $1 \leq k \leq n$.

Then $\boxed{0}$ is the unique source, and $P(\Theta, \theta) = \mathrm{conv}(x_1, \ldots, x_n)$.

Zonotopes as Propagated Polytopes



$$[0,(2,2)] \odot [0,(1,0)] \odot [0,(0,2)]$$

The Polytope Algebra

Let $\mathcal{P}_d$ be the set of all polytopes in $\mathbb{R}^d$. For $P, Q \in \mathcal{P}_d$ let

$$
\begin{aligned}
P \oplus Q &= \operatorname{conv}(P \cup Q) \\
&= \{\lambda p + (1-\lambda) q \mid p \in P,\ q \in Q,\ \lambda \in [0,1]\} \\
P \odot Q &= P + Q \\
&= \{p + q \mid p \in P,\ q \in Q\}
\end{aligned}
$$

Then $(\mathcal{P}_d, \oplus, \odot)$ is a semi-ring with idempotent addition.

The Tropical Semi-Ring

Look at the special case $\mathcal{P}_1$ ...

- and identify $[a, b]$ with $[a - b, 0]$ for $a < b$.

Then

- $[x, 0] \oplus [y, 0] = [\min(x, y), 0]$ and
- $[x, 0] \odot [y, 0] = [x + y, 0]$.

The triplet $(\mathbb{R} \cup \{\infty\}, \min, +)$ is called the *tropical semi-ring*.

The Tropical Semi-Ring

Look at the special case $\mathcal{P}_1$ ...

- and identify $[a, b]$ with $[a - b, 0]$ for $a < b$.

Then

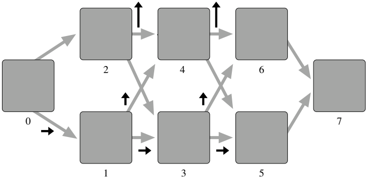
- $[x, 0] \oplus [y, 0] = [\min(x, y), 0]$ and
- $[x, 0] \odot [y, 0] = [x + y, 0]$.

The triplet $(\mathbb{R} \cup \{\infty\}, \min, +)$ is called the *tropical semi-ring*.

Newton Polytopes

For $p \in K[x_1^{\pm}, \ldots, x_d^{\pm}]$ let $\mathrm{NP}(p)$ = conv. hull of exp. vectors.

- $\mathrm{NP}(p+q) = \mathrm{NP}(p) \oplus \mathrm{NP}(q)$, unless cancellation arises
- $\mathrm{NP}(pq) = \mathrm{NP}(p) \odot \mathrm{NP}(q)$



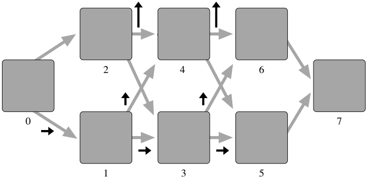
$P(\Gamma, \alpha) = \mathrm{NP}(p_7)$

A Horner type evaluation of p is paralleled by a polytope propagation of $\mathrm{NP}(p)$.

Newton Polytopes

For $p \in K[x_1^{\pm}, \ldots, x_d^{\pm}]$ let $\mathrm{NP}(p)$ = conv. hull of exp. vectors.

- $\mathrm{NP}(p+q) = \mathrm{NP}(p) \oplus \mathrm{NP}(q)$, unless cancellation arises
- $\mathrm{NP}(pq) = \mathrm{NP}(p) \odot \mathrm{NP}(q)$



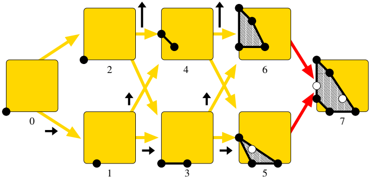
$P(\Gamma, \alpha) = \mathrm{NP}(p_7)$

A Horner type evaluation of p is paralleled by a polytope propagation of $\mathrm{NP}(p)$.

Newton Polytopes

For $p \in K[x_1^{\pm}, \dots, x_d^{\pm}]$ let $\mathrm{NP}(p)$ = conv. hull of exp. vectors.

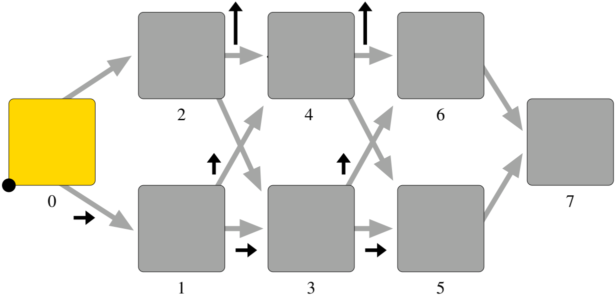
- $\mathrm{NP}(p+q) = \mathrm{NP}(p) \oplus \mathrm{NP}(q)$, unless cancellation arises
- $\mathrm{NP}(pq) = \mathrm{NP}(p) \odot \mathrm{NP}(q)$



$P(\Gamma, \alpha) = \mathrm{NP}(p_7)$

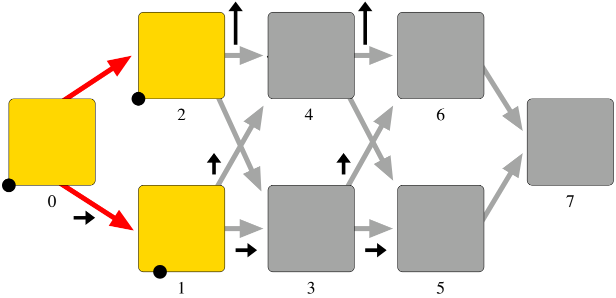
A Horner type evaluation of p is paralleled by a polytope propagation of $\mathrm{NP}(p)$.

Example, continued



$p_0 = 1$, $p_1 = p_0 \cdot x = x$, $p_2 = p_0 \cdot 1 = 1$,
$p_3 = p_1 \cdot x + p_2 \cdot 1 = x^2 + 1$, $p_4 = p_1 \cdot y + p_2 \cdot y^2 = xy + y^2$,
$p_5 = p_3 \cdot x + p_4 \cdot 1 = x^3 + xy + x + y^2$,
$p_6 = p_3 \cdot y + p_4 \cdot y^2 = x^2y + xy^3 + y + y^4$,
$p_7 = p_5 \cdot 1 + p_6 \cdot 1 = x^3 + x^2y + xy + xy^3 + x + y + y^2 + y^4$

Example, continued



$p_0 = 1$, $p_1 = p_0 \cdot x = x$, $p_2 = p_0 \cdot 1 = 1$,

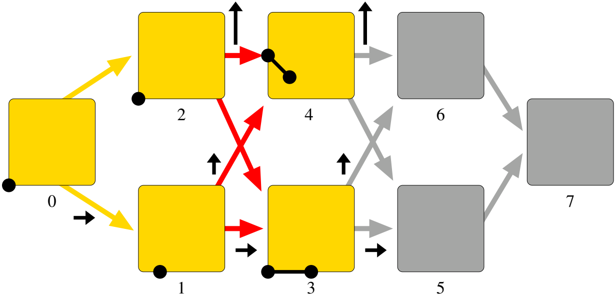
$p_3 = p_1 \cdot x + p_2 \cdot 1 = x^2 + 1$, $p_4 = p_1 \cdot y + p_2 \cdot y^2 = xy + y^2$,

$p_5 = p_3 \cdot x + p_4 \cdot 1 = x^3 + xy + x + y^2$,

$p_6 = p_3 \cdot y + p_4 \cdot y^2 = x^2 y + xy^3 + y + y^4$,

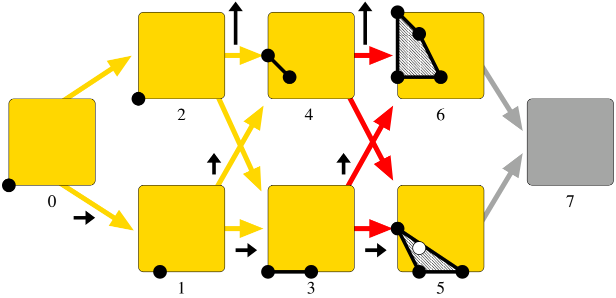
$p_7 = p_5 \cdot 1 + p_6 \cdot 1 = x^3 + x^2 y + xy + xy^3 + x + y + y^2 + y^4$

Example, continued



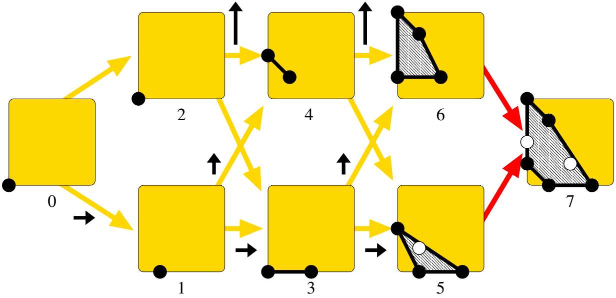
$$p_0 = 1 \ , \ p_1 = p_0 \cdot x = x, \ p_2 = p_0 \cdot 1 = 1 \ ,$$
$$p_3 = p_1 \cdot x + p_2 \cdot 1 = x^2 + 1, \ p_4 = p_1 \cdot y + p_2 \cdot y^2 = xy + y^2 \ ,$$
$$p_5 = p_3 \cdot x + p_4 \cdot 1 = x^3 + xy + x + y^2,$$
$$p_6 = p_3 \cdot y + p_4 \cdot y^2 = x^2 y + xy^3 + y + y^4 \ ,$$
$$p_7 = p_5 \cdot 1 + p_6 \cdot 1 = x^3 + x^2 y + xy + xy^3 + x + y + y^2 + y^4$$

Example, continued



$$p_0 = 1 \ , \ p_1 = p_0 \cdot x = x, \ p_2 = p_0 \cdot 1 = 1 \ ,$$
$$p_3 = p_1 \cdot x + p_2 \cdot 1 = x^2 + 1, \ p_4 = p_1 \cdot y + p_2 \cdot y^2 = xy + y^2 \ ,$$
$$p_5 = p_3 \cdot x + p_4 \cdot 1 = x^3 + xy + x + y^2,$$
$$p_6 = p_3 \cdot y + p_4 \cdot y^2 = x^2y + xy^3 + y + y^4 \ ,$$
$$p_7 = p_5 \cdot 1 + p_6 \cdot 1 = x^3 + x^2y + xy + xy^3 + x + y + y^2 + y^4$$

Example, continued



$$p_0 = 1 \ , \ p_1 = p_0 \cdot x = x, \ p_2 = p_0 \cdot 1 = 1 \ ,$$
$$p_3 = p_1 \cdot x + p_2 \cdot 1 = x^2 + 1, \ p_4 = p_1 \cdot y + p_2 \cdot y^2 = xy + y^2 \ ,$$
$$p_5 = p_3 \cdot x + p_4 \cdot 1 = x^3 + xy + x + y^2,$$
$$p_6 = p_3 \cdot y + p_4 \cdot y^2 = x^2y + xy^3 + y + y^4 \ ,$$
$$p_7 = p_5 \cdot 1 + p_6 \cdot 1 = x^3 + x^2y + xy + xy^3 + x + y + y^2 + y^4$$

Output-Sensitivity

Define $\mathrm{size}(\Gamma, \alpha)$ as the combined sizes of V-representations af all intermediate polytopes.

Theorem. The vertices of $P(\Gamma, \alpha)$ can be computed in time which is polynomially bounded in $\mathrm{size}(\Gamma, \alpha)$.

Proof. Linear optimization gives polynomial time oracle for separation.

- $\longrightarrow$ zonotopes
- $\longrightarrow$ Andrews, 1962: number of vertices of lattice polytope polynomially bounded by volume

Output-Sensitivity

Define $\mathrm{size}(\Gamma, \alpha)$ as the combined sizes of V-representations af all intermediate polytopes.

Theorem. The vertices of $P(\Gamma, \alpha)$ can be computed in time which is polynomially bounded in $\mathrm{size}(\Gamma, \alpha)$.

Proof. Linear optimization gives polynomial time oracle for separation.

$\longrightarrow$ zonotopes

$\longrightarrow$ Andrews, 1962: number of vertices of lattice polytope polynomially bounded by volume

Output-Sensitivity

Define $\operatorname{size}(\Gamma, \alpha)$ as the combined sizes of V-representations af all intermediate polytopes.

Theorem. The vertices of $P(\Gamma, \alpha)$ can be computed in time which is polynomially bounded in $\operatorname{size}(\Gamma, \alpha)$.

Proof. Linear optimization gives polynomial time oracle for separation.

- $\longrightarrow$ zonotopes
- $\longrightarrow$ Andrews, 1962: number of vertices of lattice polytope polynomially bounded by volume

Hidden Markov Models

Finite (*primary*) Markov process with l states and transition matrix $\theta = (\theta_{ij}) \in \mathbb{R}^{l \times l}$ with $\theta_{ij} \geq 0$ and $\sum_j \theta_{ij} = 1$.

Observation as a probabilistic function of the state!

- l' possible observations (where l' may be different from l)
- non-negative matrix

$$\theta' = (\theta'_{ij}) \in \mathbb{R}^{l \times l'}$$

with $\sum_j \theta'_{ij} = 1$
- known initial distribution

Hidden Markov Models

Finite (*primary*) Markov process with l states and transition matrix $\theta = (\theta_{ij}) \in \mathbb{R}^{l \times l}$ with $\theta_{ij} \geq 0$ and $\sum_j \theta_{ij} = 1$.

Observation as a probabilistic function of the state!

- l' possible observations (where l' may be different from l)
- non-negative matrix

$$\theta' = (\theta'_{ij}) \in \mathbb{R}^{l \times l'}$$

with $\sum_j \theta'_{ij} = 1$

- known initial distribution

Hidden Markov Models

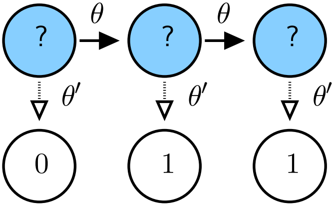
Finite (*primary*) Markov process with l states and transition matrix $\theta = (\theta_{ij}) \in \mathbb{R}^{l \times l}$ with $\theta_{ij} \geq 0$ and $\sum_j \theta_{ij} = 1$.

Observation as a probabilistic function of the state!

- l' possible observations (where l' may be different from l)
- non-negative matrix

  $$\theta' = (\theta'_{ij}) \in \mathbb{R}^{l \times l'}$$

  with $\sum_j \theta'_{ij} = 1$
- known initial distribution



$$l = l' = 2$$

Example

Let $l = l' = 2$ (state space $\{0,1\}$; initial distribution uniform),

$$\theta = \begin{pmatrix} \theta_{00} & \theta_{01} \\ \theta_{10} & \theta_{11} \end{pmatrix}, \qquad \text{and} \qquad \theta' = \begin{pmatrix} \theta'_{00} & \theta'_{01} \\ \theta'_{10} & \theta'_{11} \end{pmatrix}.$$

Observe primary process for $n = 3$ steps: $\beta_1\beta_2\beta_3 \in \{0,1\}^3$:

$$\mathrm{Prob}[Y_1 = \beta_1,\, Y_2 = \beta_2, Y_3 = \beta_3] = \frac{1}{2} \sum_{\sigma \in \{0,1\}^3} \theta_{\sigma_1\sigma_2}\theta_{\sigma_2\sigma_3}\theta'_{\sigma_1\beta_1}\theta'_{\sigma_2\beta_2}\theta'_{\sigma_3\beta_3}$$

A posteriori maximum likelihood:

$$\frac{1}{2} \max_{\sigma \in \{0,1\}^3} \theta_{\sigma_1\sigma_2}\theta_{\sigma_2\sigma_3}\theta'_{\sigma_1\beta_1}\theta'_{\sigma_2\beta_2}\theta'_{\sigma_3\beta_3}$$

Example

Let $l = l' = 2$ (state space $\{0,1\}$; initial distribution uniform),

$$\theta = \begin{pmatrix} \theta_{00} & \theta_{01} \\ \theta_{10} & \theta_{11} \end{pmatrix}, \qquad \text{and} \qquad \theta' = \begin{pmatrix} \theta'_{00} & \theta'_{01} \\ \theta'_{10} & \theta'_{11} \end{pmatrix}.$$

Observe primary process for $n = 3$ steps: $\beta_1\beta_2\beta_3 \in \{0,1\}^3$:

$$\mathrm{Prob}[Y_1 = \beta_1,\, Y_2 = \beta_2, Y_3 = \beta_3] = \frac{1}{2} \sum_{\sigma \in \{0,1\}^3} \theta_{\sigma_1\sigma_2}\theta_{\sigma_2\sigma_3}\theta'_{\sigma_1\beta_1}\theta'_{\sigma_2\beta_2}\theta'_{\sigma_3\beta_3}$$

A posteriori maximum likelihood:

$$\frac{1}{2} \max_{\sigma \in \{0,1\}^3} \theta_{\sigma_1\sigma_2}\theta_{\sigma_2\sigma_3}\theta'_{\sigma_1\beta_1}\theta'_{\sigma_2\beta_2}\theta'_{\sigma_3\beta_3}$$

Example

Let $l = l' = 2$ (state space $\{0,1\}$; initial distribution uniform),

$$\theta = \begin{pmatrix} \theta_{00} & \theta_{01} \\ \theta_{10} & \theta_{11} \end{pmatrix}, \qquad \text{and} \qquad \theta' = \begin{pmatrix} \theta'_{00} & \theta'_{01} \\ \theta'_{10} & \theta'_{11} \end{pmatrix}.$$

Observe primary process for $n = 3$ steps: $\beta_1\beta_2\beta_3 \in \{0,1\}^3$:

$$\mathrm{Prob}[Y_1 = \beta_1,\, Y_2 = \beta_2, Y_3 = \beta_3] = \frac{1}{2} \sum_{\sigma \in \{0,1\}^3} \theta_{\sigma_1\sigma_2}\theta_{\sigma_2\sigma_3}\theta'_{\sigma_1\beta_1}\theta'_{\sigma_2\beta_2}\theta'_{\sigma_3\beta_3}$$

A posteriori maximum likelihood:

$$\frac{1}{2} \max_{\sigma \in \{0,1\}^3} \theta_{\sigma_1\sigma_2}\theta_{\sigma_2\sigma_3}\theta'_{\sigma_1\beta_1}\theta'_{\sigma_2\beta_2}\theta'_{\sigma_3\beta_3}$$

Parameterized Models: Overly Simplified Example

- $l = l' = 2$ and $\theta = \theta' = \left(\begin{smallmatrix} u & v \\ v & w \end{smallmatrix}\right)$ symmetric
- assume $v \neq 0$ (otherwise primary process stationary)
- neglecting probabilistic constraints and re-scaling:

$$\hat{\theta} = \hat{\theta}' = \begin{pmatrix} x & 1 \\ 1 & y \end{pmatrix} \in \mathbb{R}[x, y]^{2\times 2}$$

Treat x, y as indeterminates!

Probability to make the specific observation 011:

$$\mathrm{Prob}[Y_1 = 0, Y_2 = 1, Y_3 = 1] = \frac{1}{2}(x^3 + x^2 y + xy + xy^3 + x + y + y^2 + y^4)$$

Parameterized Models: Overly Simplified Example

- $l = l' = 2$ and $\theta = \theta' = \left(\begin{smallmatrix} u & v \\ v & w \end{smallmatrix}\right)$ symmetric
- assume $v \neq 0$ (otherwise primary process stationary)
- neglecting probabilistic constraints and re-scaling:

$$\hat{\theta} = \hat{\theta}' = \begin{pmatrix} x & 1 \\ 1 & y \end{pmatrix} \in \mathbb{R}[x, y]^{2\times 2}$$

Treat x, y as indeterminates!

Probability to make the specific observation 011:

$$\mathrm{Prob}[Y_1 = 0, Y_2 = 1, Y_3 = 1] = \frac{1}{2}(x^3 + x^2y + xy + xy^3 + x + y + y^2 + y^4)$$

Parameterized Models: Overly Simplified Example

- $l = l' = 2$ and $\theta = \theta' = \begin{pmatrix} u & v \\ v & w \end{pmatrix}$ symmetric
- assume $v \neq 0$ (otherwise primary process stationary)
- neglecting probabilistic constraints and re-scaling:

$$\hat{\theta} = \hat{\theta}' = \begin{pmatrix} x & 1 \\ 1 & y \end{pmatrix} \in \mathbb{R}[x, y]^{2\times 2}$$

Treat x, y as indeterminates!

Probability to make the specific observation 011:

$$\mathrm{Prob}[Y_1 = 0, Y_2 = 1, Y_3 = 1] = \frac{1}{2}(x^3 + x^2y + xy + xy^3 + x + y + y^2 + y^4)$$

Parameterized Models: Overly Simplified Example

- $l = l' = 2$ and $\theta = \theta' = \left(\begin{smallmatrix} u & v \\ v & w \end{smallmatrix}\right)$ symmetric
- assume $v \neq 0$ (otherwise primary process stationary)
- neglecting probabilistic constraints and re-scaling:

$$\hat{\theta} = \hat{\theta}' = \begin{pmatrix} x & 1 \\ 1 & y \end{pmatrix} \in \mathbb{R}[x, y]^{2\times 2}$$

Treat x, y as indeterminates!

Probability to make the specific observation 011:

$$\mathrm{Prob}[Y_1 = 0, Y_2 = 1, Y_3 = 1] = \frac{1}{2}(x^3 + x^2y + xy + xy^3 + x + y + y^2 + y^4)$$

Parameterized Models: Overly Simplified Example

- $l = l' = 2$ and $\theta = \theta' = \left(\begin{smallmatrix} u & v \\ v & w \end{smallmatrix}\right)$ symmetric
- assume $v \neq 0$ (otherwise primary process stationary)
- neglecting probabilistic constraints and re-scaling:

$$\hat{\theta} = \hat{\theta}' = \begin{pmatrix} x & 1 \\ 1 & y \end{pmatrix} \in \mathbb{R}[x, y]^{2\times 2}$$

Treat x, y as indeterminates!

Probability to make the specific observation 011:

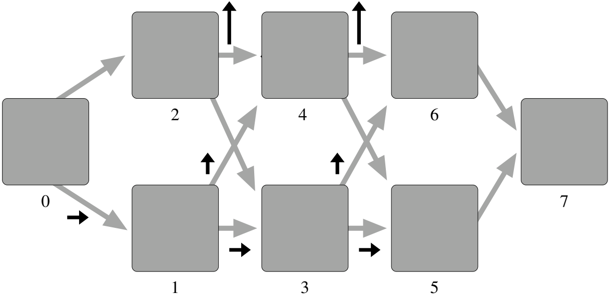
$$\mathrm{Prob}[Y_1 = 0,\, Y_2 = 1, Y_3 = 1] = \frac{1}{2}(x^3 + x^2 y + xy + xy^3 + x + y + y^2 + y^4)$$

Polytope Propagation

Observation β gives graph $\Gamma =$ observation path $\times$ state space and

$$\alpha[(i-1,j) \to (i,k)] = \mathrm{NP}(\theta_{jk}\theta'_{k,\beta_i}),$$

since $\mathrm{Prob}[X_{i-1}=j, X_i=k \,|\, Y_i=\beta_i] = \theta_{jk}\theta'_{k,\beta_i}$. here: monomial!



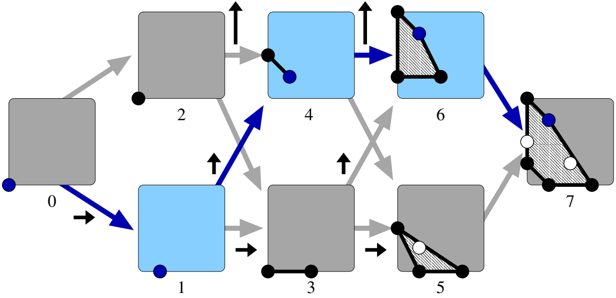
Explanation of β = directed path = point in $\mathrm{NP}(\mathrm{Prob}[Y=\beta])$.

Polytope Propagation

Observation β gives graph $\Gamma =$ observation path $\times$ state space and

$$\alpha[(i-1,j) \to (i,k)] = \mathrm{NP}(\theta_{jk}\theta'_{k,\beta_i}),$$

since $\mathrm{Prob}[X_{i-1}=j, X_i=k \,|\, Y_i=\beta_i] = \theta_{jk}\theta'_{k,\beta_i}$. here: monomial!



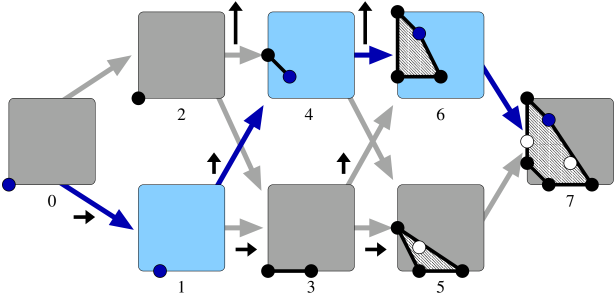
Explanation of β = directed path = point in $\mathrm{NP}(\mathrm{Prob}[Y=\beta])$.

Polytope Propagation

Observation β gives graph $\Gamma =$ observation path $\times$ state space and

$$\alpha[(i-1,j) \to (i,k)] = \mathrm{NP}(\theta_{jk}\theta'_{k,\beta_i}),$$

since $\mathrm{Prob}[X_{i-1} = j, X_i = k \,|\, Y_i = \beta_i] = \theta_{jk}\theta'_{k,\beta_i}$. here: monomial!



Explanation of β = directed path = point in $\mathrm{NP}(\mathrm{Prob}[Y = \beta])$.

Parametric Sequence Alignment

Fitch & Smith, 1983; Dewey & Woods, 2005

Two mRNA alignments of two variants of chicken hemoglobin.
Either

β: UUUGCGUCCUUUGGAACCUCUCCAGCCCCA···
α: UUUCCCCACUUC---GAUCUGUCACAC----···

or

β: UUUGCGUCCUUUGGAACCUCUCCAGCCCCA···
α: UUUCCCCACUUCG---AUCUGUCACAC----···

?

A denine, C ytosine, G uanine, U racil

Overview

since 1997 w/ Ewgenij Gawrilow
and contributions by Thilo Schröder & Niko Witte (et al.)

- convex polytopes
  - analysis of combinatorial and geometric properties
  - constructions & visualization
- TOPology Application Zoo
  - simplicial (co-)homology with coefficients in $\mathbb{Z}$ and GF(2)
  - intersection forms (of 4-manifolds), Stiefel-Whitney characteristic classes, etc.
- tropical geometry [prototype]

$>$ 50 000 uloc C++/Perl www.math.tu-berlin.de/polymake

Overview

since 1997 w/ Ewgenij Gawrilow
and contributions by Thilo Schröder & Niko Witte (et al.)

- convex polytopes
  - analysis of combinatorial and geometric properties
  - constructions & visualization
- TOPology Application Zoo
  - simplicial (co-)homology with coefficients in $\mathbb{Z}$ and $\mathrm{GF}(2)$
  - intersection forms (of 4-manifolds), Stiefel-Whitney characteristic classes, etc.
- tropical geometry [prototype]

$>$ 50 000 uloc C++/Perl `www.math.tu-berlin.de/polymake`

Overview

since 1997 w/ Ewgenij Gawrilow

and contributions by Thilo Schröder & Niko Witte (et al.)

- convex polytopes
  - analysis of combinatorial and geometric properties
  - constructions & visualization
- TOP ology A pplication Z oo
  - simplicial (co-)homology with coefficients in $\mathbb{Z}$ and GF(2)
  - intersection forms (of 4-manifolds), Stiefel-Whitney characteristic classes, etc.
- tropical geometry [prototype]

$>$ 50 000 uloc C++/Perl www.math.tu-berlin.de/polymake

Overview

since 1997 w/ Ewgenij Gawrilow

and contributions by Thilo Schröder & Niko Witte (et al.)

- convex polytopes
  - analysis of combinatorial and geometric properties
  - constructions & visualization
- TOP ology A pplication Z oo
  - simplicial (co-)homology with coefficients in $\mathbb{Z}$ and GF(2)
  - intersection forms (of 4-manifolds), Stiefel-Whitney characteristic classes, etc.
- tropical geometry [prototype]

$> 50\,000$ uloc C++/Perl `www.math.tu-berlin.de/polymake`

Perl Scripts in Version 2.1

```
application 'polytope';

die "usage:  polymake --script ... \n" unless @ARGV;

my $p=load($ARGV[0]);
my @list=();

FACETS:
for (my $i=0; $i< $p->N_FACETS ; ++$i) {
    my $facet=new Apps::polytope::RationalPolytope("...");
    Modules::client("facet", $facet, $p, $i, "-relabel");
    foreach my $other_facet (@list) {
        next FACETS if ( check_iso($facet, $other_facet) );
    }
    push @list, $facet;
}

static_javaview;
$_->VISUAL_GRAPH for @list;
```

Polytope Propagation

Printed by Michael Joswig

May 07, 05 14:52 **sum–product.cc** Page 1/2

```
/* Copyright (c) 1997-2004
   Ewgenij Gawrilow, Michael Joswig (Technische Universitaet Berlin, Germany)
   http://www.math.tu-berlin.de/polymake,  mailto:polymake@math.tu-berlin.de

   This program is free software; you can redistribute it and/or modify it
   under the terms of the GNU General Public License as published by the
   Free Software Foundation; either version 2, or (at your option) any
   later version: http://www.gnu.org/licenses/gpl.txt.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.
*/

#ident "$Project: polymake $$Id: sum-product.cc,v 1.3 2004/07/22 20:26:36 joswig Exp $"

#include <cmath>
#include <list>
#include <Poly.h>
#include <Rational.h>
#include <Graph.h>
#include <IncidenceMatrix.h>
#include <Array.h>
#include <Vector.h>
#include <Matrix.h>
#include <ListMatrix.h>
#include <linalg.h>

namespace polymake { namespace polytope {

// The graphs that we read already come with a (translation) vector specified for each edge.
// We assume that the graph is acyclic with a unique sink.  Arbitrarily many sources allowed.
typedef Graph< nothing, Vector<Rational>, directed > graph;

// Gathers next generation of graph nodes which can be defined now, since all predecessors known.
void add_next_generation(std::list<int>& next_nodes, const int v, const graph& G, const Array<Poly>& pa)
{
   for (Entire<graph::out_edge_list>::const_iterator e=entire(G.out_edges(v)); ¬e.at_end(); ++e) {
      const int x=e.to_node();
      Entire<graph::in_edge_list>::const_iterator f=entire(G.in_edges(x));
      for( ; ¬f.at_end() ∧ pa[f.from_node()].get_mode(); ++f);
      if (f.at_end())
         next_nodes.push_back(x);
   }
}

// Returns translation matrix (to be applied to row vectors from the right) for given vector.
Matrix<Rational> translation_by(const Vector<Rational>& vec)
{
   const int d=vec.dim();
   return unit_vector<Rational>(d+1,0) | (vec / unit_matrix<Rational>(d));
}

// Sum-product algorithm for polytopes.
// Operates on one Poly object from which the SUM_PRODUCT_GRAPH is read.
// Writes the VERTICES and VERTEX_NORMALS of the polytope corresponding to
// the unique sink in the graph.
void sum_product(Poly& p)
{
   // Read the graph.
   const graph G=p.give("SUM_PRODUCT_GRAPH");
   const int n(G.nodes());
   if (n≡0)
      throw std::runtime_error("SUM_PRODUCT_GRAPH must be non-empty");

   // The dimension of the ambient space.
   const int d=p.give("AMBIENT_DIM");

   // This is the description of the origin as a 0-dimensional polytope (living in d-space).
   // Used to initialize the computation at the sources of the graph.
   const Matrix<Rational> single_point_vertices(vector2row(unit_vector<Rational>(d+1,0)));
   const IncidenceMatrix<> single_point_vif;

   // This is where all the intermediate polytopes are stored.
   // The nodes in the graph are consecutively numbered, starting at 0.
   // The corresponding polytope can be accessed by indexing with the node number.
   // In the beginning the polytopes are undefined.
   Array<Poly> pa(n);
   std::list<int> next_nodes;

   // Initialize by assigning a single point (origin) to each source in the graph.
   for (int v=0; v<n; ++v) {
      if (G.in_degree(v)≡0) {
         pa[v].init(0, ios::in | ios::out | ios::trunc, "RationalPolytope");
         pa[v].take("VERTICES") << single_point_vertices;
         pa[v].take("VERTICES_IN_FACETS") << single_point_vif;
         add_next_generation(next_nodes,v,G,pa);
      }
   }
```

Saturday May 07, 2005 1/2

Printed by Michael Joswig

May 07, 05 14:52 **sum–product.cc** Page 2/2

```
   // At each node of the graph recursively define a polytope as the convex hull of
   // the translated predecessors.
   // We also try to find the sink on the way.
   int sink=-1;  // no valid node number; indicates sink not found yet
   while(¬next_nodes.empty()) {
      // Get some node w for which we know all the predecessors.
      const int w=next_nodes.front(); next_nodes.pop_front();
      if (pa[w].get_mode())
         throw std::runtime_error("unvisited node already initialized");
      // Initialize client-server communication for that node's polytope.
      pa[w].init(0, ios::in | ios::out | ios::trunc, "RationalPolytope");

      // The polytope will be specified as the convex hull of points, which will be collected
      // from other polytopes.
      // The special data type ListMatrix is efficient in terms of concatenating rows
      // (which correspond to points); not efficient in terms of matrix operations
      // (although all operations are defined), but this is ok, since we do not compute
      // anything in this step.
      ListMatrix< Vector<Rational> > points(0,d+1);
      for (Entire<graph::in_edge_list>::const_iterator e=entire(G.in_edges(w)); ¬e.at_end(); ++e) {
         // Node v is the current predecessor to process.
         const int v=e.from_node();
         // Translation vector in the edge from v to w.
         const Vector<Rational> vec=*e;
         // We read the vertices of the predecessor polytope.  polymake's rule basis by
         // default uses cdd' implementation to check for redundant (= non-vertex) points
         // among the input by solving linear programs.
         // No convex hull computation necessary.
         const Matrix<Rational> these_vertices=pa[v].give("VERTICES");
         // Concatenates the translated matrix (where the rows correspond to the vertices
         // of the predecessor) to what we already have.
         points /= these_vertices*translation_by(vec);
      }
      // Define the polytope as the convex hull of all those points.
      pa[w].take("POINTS") << points;
      if (G.out_degree(w)≡0)
         sink=w; // We did find a sink.
      else
         add_next_generation(next_nodes,w,G,pa);
   }

   // This will be just any sink; indeterministic if graph has several sinks.
   if (sink<0)
      throw std::runtime_error("no sink found in digraph");

   const Matrix<Rational> sink_vertices = pa[sink].give("VERTICES");
   const Matrix<Rational> sink_normals = pa[sink].give("VERTEX_NORMALS");

   // The sink defines the polytope we are after.
   p.take("VERTICES") << sink_vertices;
   p.take("VERTEX_NORMALS") << sink_normals;
}

} }

using namespace polymake;

/** @file sum-product
 *
 *  Sum-product algorithm for polytopes.
 *
 *  @synopsis sum_product <file>
 *
 *  @reading  SUM_PRODUCT_GRAPH
 *  @writing  VERTICES, VERTEX_NORMALS
 *
 *  @index clients ...
 */

int main(int argc, char *argv[]) {
   if (argc ≠ 2) {
      cerr << "usage: " << argv[0] << " <file>" << endl;
      return 1;
   }
   try {
      Poly p(argv[1], ios::in | ios::out);
      polytope::sum_product(p);
   }
   catch (const std::exception& e) {
      cerr << e.what() << endl;
      return 1;
   }
   return 0;
}
```

2/2 Saturday May 07, 2005

Example, continued

\> binary-markov-graph b011.poly 011

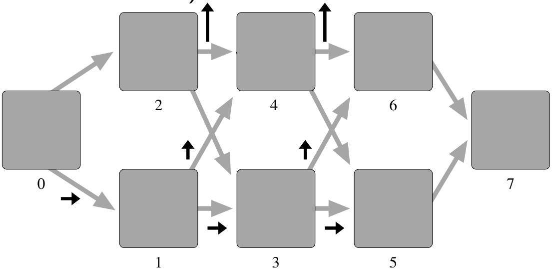
\> polymake b011.poly "numbered(SUM_PRODUCT_GRAPH)"

```
numbered(SUM_PRODUCT_GRAPH)
0:{(1 1 0) (2 0 0)}
1:{(3 1 0) (4 0 1)}
2:{(3 0 0) (4 0 2)}
3:{(5 1 0) (6 0 1)}
4:{(5 0 0) (6 0 2)}
5:{(7 0 0)}
6:{(7 0 0)}
7:{ }
```

Example, continued

> binary-markov-graph b011.poly 011

> polymake b011.poly "numbered(SUM_PRODUCT_GRAPH)"
numbered(SUM_PRODUCT_GRAPH)
0:{(1 1 0) (2 0 0)}
1:{(3 1 0) (4 0 1)}
2:{(3 0 0) (4 0 2)}
3:{(5 1 0) (6 0 1)}
4:{(5 0 0) (6 0 2)}
5:{(7 0 0)}
6:{(7 0 0)}
7:{ }



Example, the End

> sum_product b011.poly
> polymake b011.poly "numbered(VERTICES)"\
"numbered(VERTEX_NORMALS)"

numbered(VERTICES)
0:1 3 0
1:1 1 0
2:1 0 1
3:1 1 3
4:1 0 4
numbered(VERTEX_NORMALS)
0:0 1/2 0
1:0 -1/2 -3/2
2:0 -2 -1
3:0 4 3
4:0 -1/2 1/2

Vertex $(1, 3)$ corresponds to ...

- path $0 \to 1 \to 4 \to 6 \to 7$
- state sequence 011 in primary process

If we are to maximize $4x + 3y$, then we can trust our observation 011.

Example, the End

> sum_product b011.poly

> polymake b011.poly "numbered(VERTICES)"\
"numbered(VERTEX_NORMALS)"

numbered(VERTICES)
0:1 3 0
1:1 1 0
2:1 0 1
3:1 1 3
4:1 0 4
numbered(VERTEX_NORMALS)
0:0 1/2 0
1:0 -1/2 -3/2
2:0 -2 -1
3:0 4 3
4:0 -1/2 1/2

Vertex $(1, 3)$ corresponds to ...

- path $0 \to 1 \to 4 \to 6 \to 7$
- state sequence 011 in primary process

If we are to maximize $4x + 3y$, then we can trust our observation 011.

Example, the End

> sum_product b011.poly

> polymake b011.poly "numbered(VERTICES)"\
"numbered(VERTEX_NORMALS)"

numbered(VERTICES)
0:1 3 0
1:1 1 0
2:1 0 1
3:1 1 3
4:1 0 4
numbered(VERTEX_NORMALS)
0:0 1/2 0
1:0 -1/2 -3/2
2:0 -2 -1
3:0 4 3
4:0 -1/2 1/2

Vertex $(1, 3)$ corresponds to ...

- path $0 \to 1 \to 4 \to 6 \to 7$
- state sequence 011 in primary process

If we are to maximize $4x + 3y$, then we can trust our observation 011.