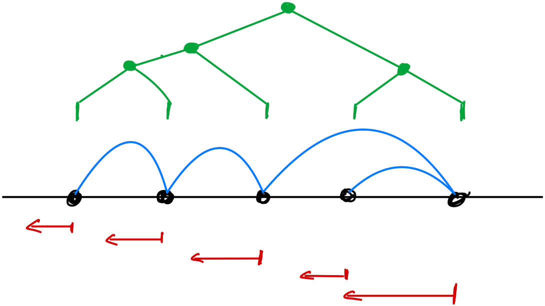
From linear programming to particle collisions



Raman Sanyal

Goethe-Universität Frankfurt

joint with

Alex Black and Niklas Lütjeharms

This talk is based on a true story.

Linear programs and the simplex algorithm

Linear Program (LP)

$$\begin{array}{lll} \max & c_1\, x_1 + \cdots + c_d\, x_d & \\ \text{s.t.} & a_{i1}x_1 + \cdots + a_{id}x_d \leq \; b_i & \text{for } i = 1, \ldots, n \end{array}$$

Dantzig's simplex algorithm is the method of choice for solving LPs

Linear programs and the simplex algorithm

Linear Program (LP)

$$\begin{array}{lll} \max & c_1\, x_1 + \cdots + c_d\, x_d & \\ \text{s.t.} & a_{i1}x_1 + \cdots + a_{id}x_d \leq \ b_i & \text{for } i = 1, \ldots, n \end{array}$$

Dantzig's simplex algorithm is the method of choice for solving LPs

Geometer's view:

Linear programs and the simplex algorithm

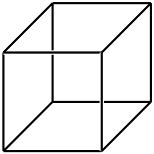
Linear Program (LP)

$$\begin{array}{lll} \max & c_1\, x_1 + \cdots + c_d\, x_d & \\ \text{s.t.} & a_{i1}x_1 + \cdots + a_{id}x_d \leq \; b_i & \text{for } i = 1, \ldots, n \end{array}$$

Dantzig's simplex algorithm is the method of choice for solving LPs

Geometer's view:

$P = \{x \in \mathbb{R}^d : Ax \leq b\}$ is a polytope



Linear programs and the simplex algorithm

Linear Program (LP)

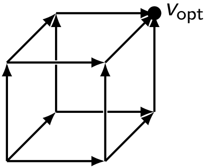
$$\begin{array}{lll} \max & c_1\, x_1 + \cdots + c_d\, x_d & \\ \text{s.t.} & a_{i1}x_1 + \cdots + a_{id}x_d \leq \; b_i & \text{for } i = 1, \ldots, n \end{array}$$

Dantzig's simplex algorithm is the method of choice for solving LPs

Geometer's view:

$P = \{x \in \mathbb{R}^d : Ax \leq b\}$ is a polytope

$c = (c_1, \ldots, c_d)$ yields a unique sink orientation of the graph of P



Linear programs and the simplex algorithm

Linear Program (LP)

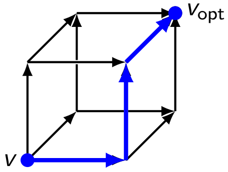
$$\begin{array}{lll} \max & c_1\, x_1 + \cdots + c_d\, x_d & \\ \text{s.t.} & a_{i1}x_1 + \cdots + a_{id}x_d \leq\ b_i & \text{for } i = 1, \ldots, n \end{array}$$

Dantzig's simplex algorithm is the method of choice for solving LPs

Geometer's view:

$P = \{x \in \mathbb{R}^d : Ax \leq b\}$ is a polytope

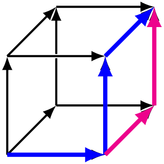
$c = (c_1, \ldots, c_d)$ yields a unique sink orientation of the graph of P



simplex algorithm produces a path from *any* starting node v to the sink v_{opt}

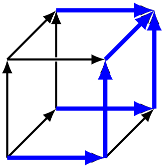
Pivot rules

The pivot rule chooses the path from *any* starting vertex.



Pivot rules

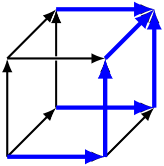
The pivot rule chooses the path from *any* starting vertex.



Pivot rule is memory-less if choices are made independently and locally.

Pivot rules

The pivot rule chooses the path from *any* starting vertex.



Pivot rule is memory-less if choices are made independently and locally.

For fixed LP (P, c), memory-less pivot rules are given by arborescences

$$\mathcal{A} : V \setminus v_{\text{opt}} \to V \qquad \mathcal{A}(v) \in \mathrm{Nb}_{+}(v) \quad \text{for all } v \neq v_{\text{opt}} ,$$

where $\mathrm{Nb}_{+}(v)$ are the improving neighbors of v.

Some pivot rules

(P, c) fixed and a generic weight $\omega \in \mathbb{R}^d$.

- Greatest improvement

$$\langle c, u - v \rangle$$

- p-Steepest Edge

$$\frac{\langle c, u - v \rangle}{\|u - v\|_p}$$

- Max-slope

$$\frac{\langle \omega, u - v \rangle}{\langle c, u - v \rangle}$$

Some pivot rules

(P, c) fixed and a generic weight $\omega \in \mathbb{R}^d$.

- Greatest improvement

$$\langle \boldsymbol{\omega}, u - v \rangle$$

- p-Steepest Edge

$$\frac{\langle \boldsymbol{\omega}, u - v \rangle}{\|u - v\|_p}$$

- Max-slope

$$\frac{\langle \boldsymbol{\omega}, u - v \rangle}{\langle c, u - v \rangle}$$

Some pivot rules

(P, c) fixed and a generic weight $\omega \in \mathbb{R}^d$.

- Greatest improvement

$$\langle \boldsymbol{\omega}, u - v \rangle$$

- p-Steepest Edge

$$\frac{\langle \boldsymbol{\omega}, u - v \rangle}{\|u - v\|_p}$$

- Max-slope

$$\frac{\langle \boldsymbol{\omega}, u - v \rangle}{\langle c, u - v \rangle}$$

For a weight ω and normalization $\eta : \mathbb{R}^d \to \mathbb{R}$
the associated normalized-weight pivot rule is $\mathcal{A}^\omega : V \setminus v_{\text{opt}} \to V$

$$\mathcal{A}^\omega(v) \;:=\; \operatorname{argmax} \left\{ \frac{\langle \omega, u - v \rangle}{\eta(u - v)} : u \in \mathrm{Nb}_+(v) \right\} .$$

A polytope of pivot rules

Polytope $P \subset \mathbb{R}^d$, objective function $c \in \mathbb{R}^d$, and normalization $\eta : \mathbb{R}^d \to \mathbb{R}$.

A polytope of pivot rules

Polytope $P \subset \mathbb{R}^d$, objective function $c \in \mathbb{R}^d$, and normalization $\eta : \mathbb{R}^d \to \mathbb{R}$.

For an arborescence $\mathcal{A} : V \setminus v_{\text{opt}} \to V$ define

$$\Psi(\mathcal{A}) \; := \; \sum_{v \neq v_{\text{opt}}} \frac{\mathcal{A}(v) - v}{\eta(\mathcal{A}(v) - v)}$$

A polytope of pivot rules

Polytope $P \subset \mathbb{R}^d$, objective function $c \in \mathbb{R}^d$, and normalization $\eta : \mathbb{R}^d \to \mathbb{R}$.

For an arborescence $\mathcal{A} : V \setminus v_{\text{opt}} \to V$ define

$$\Psi(\mathcal{A}) \ := \ \sum_{v \neq v_{\text{opt}}} \frac{\mathcal{A}(v) - v}{\eta(\mathcal{A}(v) - v)}$$

Pivot rule polytope

$$\Pi(P, c) \ := \ \text{conv}\{\Psi(\mathcal{A}) : \mathcal{A} \text{ arborescence of } (P, c)\}$$

A polytope of pivot rules

Polytope $P \subset \mathbb{R}^d$, objective function $c \in \mathbb{R}^d$, and normalization $\eta : \mathbb{R}^d \to \mathbb{R}$.

For an arborescence $\mathcal{A} : V \setminus v_{\text{opt}} \to V$ define

$$\Psi(\mathcal{A}) \;:=\; \sum_{v \neq v_{\text{opt}}} \frac{\mathcal{A}(v) - v}{\eta(\mathcal{A}(v) - v)}$$

Pivot rule polytope

$$\Pi(P, c) \;:=\; \text{conv}\{\Psi(\mathcal{A}) : \mathcal{A} \text{ arborescence of } (P, c)\}$$

Theorem (Black, De Loera, Lütjeharms, S.'22)

For generic $\omega \in \mathbb{R}^d$ and arborescence $\mathcal{A}$ the following are equivalent:

1. $\Psi(\mathcal{A})$ *is the vertex of* $\Pi(P, c)$ *maximizing* ω.
2. $\mathcal{A}$ *is the normalized weight arborescence* $\mathcal{A}^\omega$.

A polytope of pivot rules

Polytope $P \subset \mathbb{R}^d$, objective function $c \in \mathbb{R}^d$, and normalization $\eta : \mathbb{R}^d \to \mathbb{R}$.

For an arborescence $\mathcal{A} : V \setminus v_{\text{opt}} \to V$ define

$$\Psi(\mathcal{A}) \;:=\; \sum_{v \neq v_{\text{opt}}} \frac{\mathcal{A}(v) - v}{\eta(\mathcal{A}(v) - v)}$$

Pivot rule polytope

$$\Pi(P, c) \;:=\; \text{conv}\{\Psi(\mathcal{A}) : \mathcal{A} \text{ arborescence of } (P, c)\}$$

Theorem (Black, De Loera, Lütjeharms, S.'22)

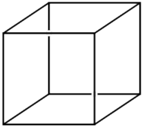
For generic $\omega \in \mathbb{R}^d$ and arborescence $\mathcal{A}$ the following are equivalent:

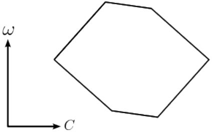
1. $\Psi(\mathcal{A})$ *is the vertex of* $\Pi(P, c)$ *maximizing* ω.
2. $\mathcal{A}$ *is the normalized weight arborescence* $\mathcal{A}^\omega$.

(coherent) triangulations? GKZ-vectors? secondary/fiber polytopes?

Max-slope and monotone path polytopes

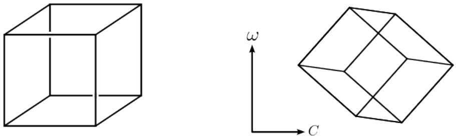
c, ω define a projection $\pi : \mathbb{R}^d \to \mathbb{R}^2$





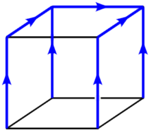
Max-slope and monotone path polytopes

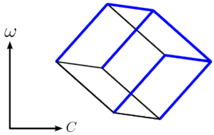
c, ω define a projection $\pi : \mathbb{R}^d \to \mathbb{R}^2$



Max-slope and monotone path polytopes

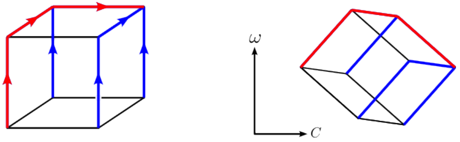
c, ω define a projection $\pi : \mathbb{R}^d \to \mathbb{R}^2$





Max-slope and monotone path polytopes

c, ω define a projection $\pi : \mathbb{R}^d \to \mathbb{R}^2$

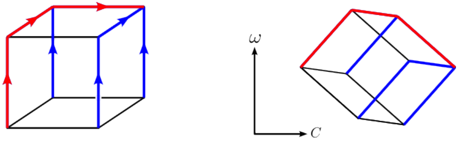


Path from minimizer $v_{-\text{opt}}$ to v_{opt} is a c-monotone path.

This is a coherent monotone path in the sense of Billera–Sturmfels.

Max-slope and monotone path polytopes

c, ω define a projection $\pi : \mathbb{R}^d \to \mathbb{R}^2$



Path from minimizer $v_{-\text{opt}}$ to v_{opt} is a c-monotone path.

This is a coherent monotone path in the sense of Billera–Sturmfels.

Theorem (Billera-Sturmfels'92)

The monotone path polytope $\Sigma(P, c)$ parametrizes coherent c-monotone paths.

Theorem (Black, De Loera, Lütjeharms, S.'22)

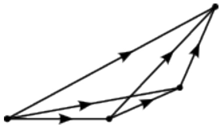
$\Sigma(P, c)$ *is a weak Minkowski summand of* $\Pi(P, c)$.

An example: max-slope pivot rules on simplices

n-dimensional simplex

$$\Delta_n \;=\; \mathrm{conv}(e_1, e_2, \ldots, e_{n+1}) \subset \mathbb{R}^{n+1}$$

Objective function $c = (c_1 < c_2 < \cdots < c_{n+1})$.

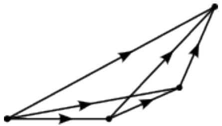


An example: max-slope pivot rules on simplices

n-dimensional simplex

$$\Delta_n = \mathrm{conv}(e_1, e_2, \ldots, e_{n+1}) \subset \mathbb{R}^{n+1}$$

Objective function $c = (c_1 < c_2 < \cdots < c_{n+1})$.



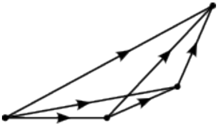
```
[42]: PivPolytopes = []
for n in [2, 3, 4, 5, 6, 7]:
    P = polytopes.simplex(n)
    c = vector([ 2 ** i for i in range(n+1) ])
    D = P.graph().orient( lambda e: e if c*e[0].vector() < c*e[1].vector() else (e[1],e[0], e[2]) )

    PP = []
    for v in D.vertices():
        Nb = [ e[1].vector() - v.vector() for e in D.outgoing_edges(v) ]
        if Nb != []:
            PP.append( Polyhedron( [ u / (c*u) for u in Nb ] ) )
    PP = sum(PP)
    PivPolytopes.append( (n, PP) )
[ PP.n_vertices() for n, PP in PivPolytopes ]
```

An example: max-slope pivot rules on simplices

n-dimensional simplex

$$\Delta_n \; = \; \mathrm{conv}(e_1, e_2, \ldots, e_{n+1}) \subset \mathbb{R}^{n+1}$$

Objective function $c = (c_1 < c_2 < \cdots < c_{n+1})$.



```
[42]: PivPolytopes = []
      for n in [2, 3, 4, 5, 6, 7]:
          P = polytopes.simplex(n)
          c = vector([ 2 ** i for i in range(n+1) ])
          D = P.graph().orient( lambda e: e if c*e[0].vector() < c*e[1].vector() else (e[1],e[0], e[2]) )

          PP = []
          for v in D.vertices():
              Nb = [ e[1].vector() - v.vector() for e in D.outgoing_edges(v) ]
              if Nb != []:
                  PP.append( Polyhedron( [ u / (c*u) for u in Nb ] ) )
          PP = sum(PP)
          PivPolytopes.append( (n, PP) )
      [ PP.n_vertices() for n, PP in PivPolytopes ]
```

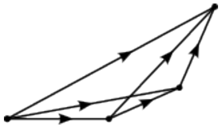
```
[42]: [2, 5, 14, 42, 132, 429]
```

An example: max-slope pivot rules on simplices

n-dimensional simplex

$$\Delta_n \;=\; \mathrm{conv}(e_1, e_2, \ldots, e_{n+1}) \subset \mathbb{R}^{n+1}$$

Objective function $c = (c_1 < c_2 < \cdots < c_{n+1})$.



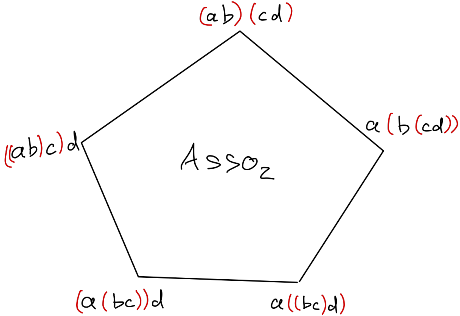
```
[42]: PivPolytopes = []
      for n in [2, 3, 4, 5, 6, 7]:
          P = polytopes.simplex(n)
          c = vector([ 2 ** i for i in range(n+1) ])
          D = P.graph().orient( lambda e: e if c*e[0].vector() < c*e[1].vector() else (e[1],e[0], e[2]) )

          PP = []
          for v in D.vertices():
              Nb = [ e[1].vector() - v.vector() for e in D.outgoing_edges(v) ]
              if Nb != []:
                  PP.append( Polyhedron( [ u / (c*u) for u in Nb ] ) )
          PP = sum(PP)
          PivPolytopes.append( (n, PP) )
      [ PP.n_vertices() for n, PP in PivPolytopes ]

[42]: [2, 5, 14, 42, 132, 429]

[43]: [ PP.is_combinatorially_isomorphic( polytopes.associahedron(['A',n-1] ) ) for n, PP in PivPolytopes ]

[43]: [True, True, True, True, True, True]
```

The associahedron Asso_{n-1}

or Stasheff polytope is an $(n-1)$-dim. simple polytope that parametrizes...

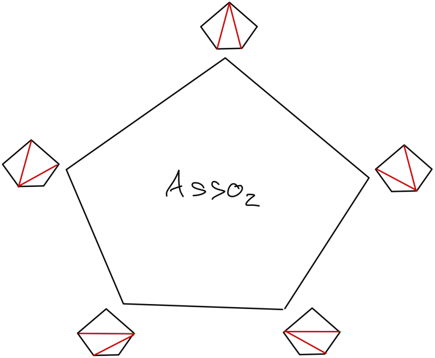
The associahedron Asso_{n-1}

or Stasheff polytope is an $(n-1)$-dim. simple polytope that parametrizes...
...parenthesizations of a (non-associative) product $a_1 \cdot a_2 \cdots a_{n+2}$



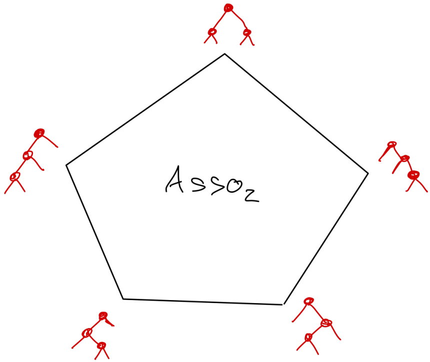
The associahedron Asso_{n-1}

or Stasheff polytope is an $(n-1)$-dim. simple polytope that parametrizes...
...triangulations of $(n+2)$-gon



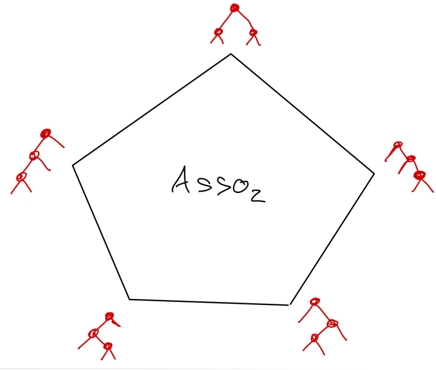
The associahedron Asso_{n-1}

or Stasheff polytope is an $(n-1)$-dim. simple polytope that parametrizes...
...binary trees with $n+1$ leaves



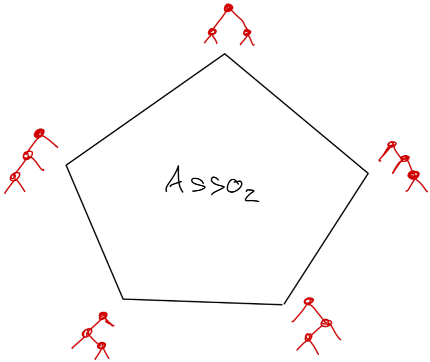
The associahedron Asso_{n-1}

or Stasheff polytope is an $(n-1)$-dim. simple polytope that parametrizes...
...collisions of $n+1$ particles



The associahedron Asso_{n-1}

or Stasheff polytope is an $(n-1)$-dim. simple polytope that parametrizes...
...collisions of $n+1$ particles



Theorem (Black-Lütjeharms-S.'24)

The max-slope pivot rule polytope of (Δ_n, c) *is isomorphic to* Asso_{n-1}.

Prisms over simplices

Prism over Δ_n is $\text{prism}(\Delta_n) = \Delta_n \times [0,1] \subset \mathbb{R}^{n+2}$.

Generic objective function $c' = (c_1 < c_2 < \cdots < c_{n+1} < d_1)$.

Prisms over simplices

Prism over Δ_n is $\text{prism}(\Delta_n) = \Delta_n \times [0,1] \subset \mathbb{R}^{n+2}$.

Generic objective function $c' = (c_1 < c_2 < \cdots < c_{n+1} < d_1)$.

Number of max-slope arborescences of $(\text{prism}(\Delta_n), c')$ for $n \geq 0$

$$1, 2, 6, 21, 80, 322, \ldots$$

Prisms over simplices

Prism over Δ_n is $\text{prism}(\Delta_n) = \Delta_n \times [0,1] \subset \mathbb{R}^{n+2}$.

Generic objective function $c' = (c_1 < c_2 < \cdots < c_{n+1} < d_1)$.

Number of max-slope arborescences of $(\text{prism}(\Delta_n), c')$ for $n \geq 0$

$$1, 2, 6, 21, 80, 322, \ldots$$

Sloane database (OEIS): Number of vertices of the multiplihedron Mul_n.

Prisms over simplices

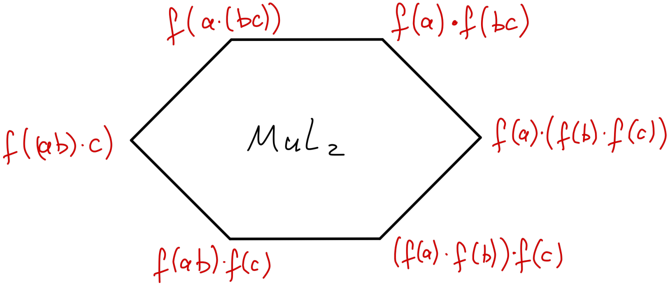
Prism over Δ_n is $\text{prism}(\Delta_n) = \Delta_n \times [0,1] \subset \mathbb{R}^{n+2}$.

Generic objective function $c' = (c_1 < c_2 < \cdots < c_{n+1} < d_1)$.

Number of max-slope arborescences of $(\text{prism}(\Delta_n), c')$ for $n \geq 0$

$$1, 2, 6, 21, 80, 322, \ldots$$

Sloane database (OEIS): Number of vertices of the multiplihedron Mul_n.
Number of ways to evaluate $f(a_1 \cdot a_2 \cdots a_{n+1})$, where f is multiplicative.



Prisms over simplices

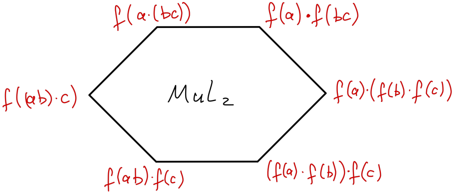
Prism over Δ_n is $\text{prism}(\Delta_n) = \Delta_n \times [0,1] \subset \mathbb{R}^{n+2}$.

Generic objective function $c' = (c_1 < c_2 < \cdots < c_{n+1} < d_1)$.

Number of max-slope arborescences of $(\text{prism}(\Delta_n), c')$ for $n \geq 0$

$$1, 2, 6, 21, 80, 322, \ldots$$

Sloane database (OEIS): Number of vertices of the multiplihedron Mul_n.
Number of ways to evaluate $f(a_1 \cdot a_2 \cdots a_{n+1})$, where f is multiplicative.



Theorem (Black-Lütjeharms-S.'24)

Max-slope pivot polytope of $(\text{prism}(\Delta_n), c')$ *is isom. to multiplihedron* Mul_n.

Prisms over simplices

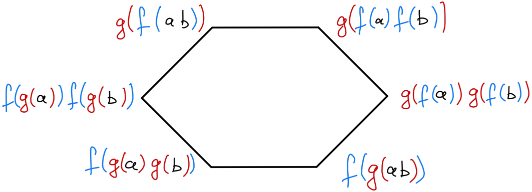
Prism over Δ_n is $\text{prism}(\Delta_n) = \Delta_n \times [0,1] \subset \mathbb{R}^{n+2}$.

Generic objective function $c' = (c_1 < c_2 < \cdots < c_{n+1} < d_1)$.

Number of max-slope arborescences of $(\text{prism}(\Delta_n), c')$ for $n \geq 0$

$$1, 2, 6, 21, 80, 322, \ldots$$

Sloane database (OEIS): Number of vertices of the multiplihedron Mul_n.
Number of ways to evaluate $f(a_1 \cdot a_2 \cdots a_{n+1})$, where f is multiplicative.



Theorem (Black-Lütjeharms-S.'24)

Max-slope pivot polytope of $(\text{prism}^k(\Delta_n), c')$ *is isom. to* k*-multiplihedron* Mul_n^k.

Products of simplices

$$\text{prism}^k(\Delta_n) = \Delta_n \times \underbrace{\Delta_1 \times \cdots \times \Delta_1}_{k}$$

Max-slope pivot rule polytopes of $\Delta_n \times \Delta_m$

Theorem (Black, Lütjeharms, S.'24)

$\Pi(\Delta_n \times \Delta_m, c)$ *is combinatorially isomorphic to the*

Products of simplices

$$\text{prism}^k(\Delta_n) = \Delta_n \times \underbrace{\Delta_1 \times \cdots \times \Delta_1}_{k}$$

Max-slope pivot rule polytopes of $\Delta_n \times \Delta_m$

Theorem (Black, Lütjeharms, S.'24)

$\Pi(\Delta_n \times \Delta_m, c)$ *is combinatorially isomorphic to the* (m, n)*-constrainahedron.*

Constrainahedra [Bottman–Poliakova'22, Chapoton–Pilaud'22] capture the combinatorics of particles to parallel lines in $\mathbb{R}^2$.

Products of simplices

$$\text{prism}^k(\Delta_n) = \Delta_n \times \underbrace{\Delta_1 \times \cdots \times \Delta_1}_{k}$$

Max-slope pivot rule polytopes of $\Delta_n \times \Delta_m$

Theorem (Black, Lütjeharms, S.'24)

$\Pi(\Delta_n \times \Delta_m, c)$ *is combinatorially isomorphic to the* (m, n)*-constrainahedron.*

Constrainahedra [Bottman–Poliakova'22, Chapoton–Pilaud'22] capture the combinatorics of particles to parallel lines in $\mathbb{R}^2$.

What is the connection between max-slope pivot rules and particles?

Products of simplices

$$\text{prism}^k(\Delta_n) = \Delta_n \times \underbrace{\Delta_1 \times \cdots \times \Delta_1}_{k}$$

Max-slope pivot rule polytopes of $\Delta_n \times \Delta_m$

Theorem (Black, Lütjeharms, S.'24)

$\Pi(\Delta_n \times \Delta_m, c)$ *is combinatorially isomorphic to the* (m, n)*-constrainahedron.*

Constrainahedra [Bottman–Poliakova'22, Chapoton–Pilaud'22] capture the combinatorics of particles to parallel lines in $\mathbb{R}^2$.

What is the connection between max-slope pivot rules and particles?

Theorem (Pilaud–Poullot'25)

$\Pi(\Delta_{n_1} \times \cdots \times \Delta_{n_l}, c)$ *is comb. isomorphic to the* $(n_1, \ldots, n_l)$*-constrainahedron.*

Piecewise-linear homeomorphism between normal fans of $\Pi(\Delta_n, c)$ and Loday's associahedron Asso_{n-1}. Extends to shuffle products of deformed permutahedra.

Max-slope pivot rules on simplices

simplex $\Delta_n = \text{conv}(e_1, e_2, \ldots, e_{n+1})$ obj. function $c = (c_1 < c_2 < \cdots < c_{n+1})$

Max-slope pivot rules on simplices

simplex $\Delta_n = \text{conv}(e_1, e_2, \ldots, e_{n+1})$ obj. function $c = (c_1 < c_2 < \cdots < c_{n+1})$

Arborescences correspond to maps $A : [n] \to [n+1]$ with $A(i) > i$ for all i

$\longrightarrow$ $n!$ many arborescences

Max-slope pivot rules on simplices

simplex $\Delta_n = \text{conv}(e_1, e_2, \ldots, e_{n+1})$ obj. function $c = (c_1 < c_2 < \cdots < c_{n+1})$

Arborescences correspond to maps $A : [n] \to [n+1]$ with $A(i) > i$ for all i
$\longrightarrow$ $n!$ many arborescences

Max-slope arborescence for $\omega = (\omega_1, \ldots, \omega_n)$

$$\mathcal{A}^\omega(i) \;=\; \text{argmax} \left\{ \frac{\omega_j - \omega_i}{c_j - c_i} : j > i \right\}$$

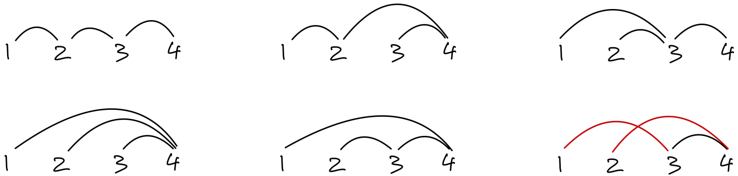
Max-slope pivot rules on simplices

simplex $\Delta_n = \mathrm{conv}(e_1, e_2, \ldots, e_{n+1})$ obj. function $c = (c_1 < c_2 < \cdots < c_{n+1})$

Arborescences correspond to maps $A : [n] \to [n+1]$ with $A(i) > i$ for all i
$\longrightarrow$ $n!$ many arborescences

Max-slope arborescence for $\omega = (\omega_1, \ldots, \omega_n)$

$$\mathcal{A}^\omega(i) \;=\; \mathrm{argmax}\left\{ \frac{\omega_j - \omega_i}{c_j - c_i} : j > i \right\}$$



Non-crossing arborescences

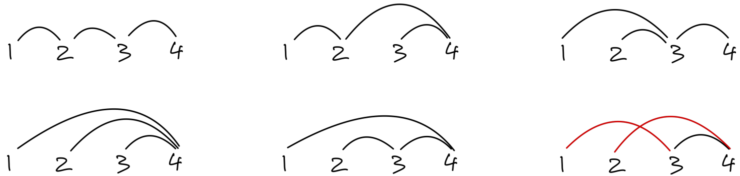
Max-slope pivot rules on simplices

simplex $\Delta_n = \operatorname{conv}(e_1, e_2, \ldots, e_{n+1})$ obj. function $c = (c_1 < c_2 < \cdots < c_{n+1})$

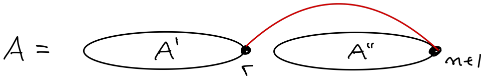
Arborescences correspond to maps $A : [n] \to [n+1]$ with $A(i) > i$ for all i
$\longrightarrow$ $n!$ many arborescences

Max-slope arborescence for $\omega = (\omega_1, \ldots, \omega_n)$

$$\mathcal{A}^\omega(i) \;=\; \operatorname{argmax}\left\{ \frac{\omega_j - \omega_i}{c_j - c_i} : j > i \right\}$$



Non-crossing arborescences – Catalan recurrence!



Particles with locations and velocities

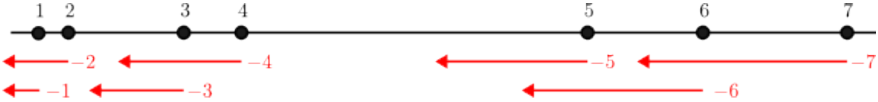
Consider $n \geq 2$ labelled particles on a real line.

Particles with locations and velocities

Consider $n \geq 2$ labelled particles on a real line.

At time $t = 0$, they are at locations $-\omega_1 \leq -\omega_2 \leq \cdots \leq -\omega_n$.

For $t > 0$ they travel at constant velocities $0 > -c_1 > -c_2 > \cdots > -c_n$



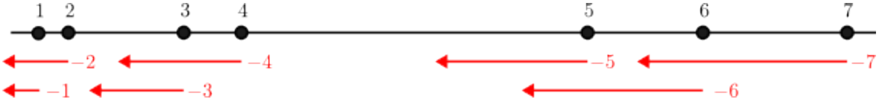
If $i < j$ collide, then i is absorbed by the faster j, which continues at speed $-c_j$.

Particles with locations and velocities

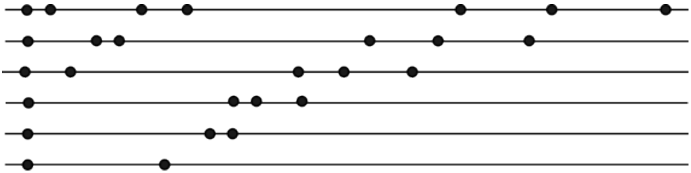
Consider $n \geq 2$ labelled particles on a real line.

At time $t = 0$, they are at locations $-\omega_1 \leq -\omega_2 \leq \cdots \leq -\omega_n$.

For $t > 0$ they travel at constant velocities $0 > -c_1 > -c_2 > \cdots > -c_n$



If $i < j$ collide, then i is absorbed by the faster j, which continues at speed $-c_j$.

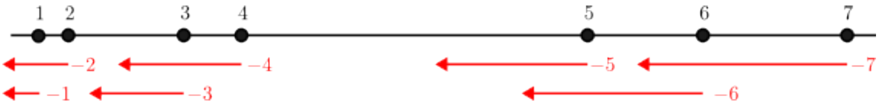


Particles with locations and velocities

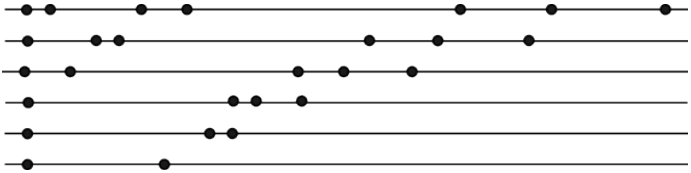
Consider $n \geq 2$ labelled particles on a real line.

At time $t = 0$, they are at locations $-\omega_1 \leq -\omega_2 \leq \cdots \leq -\omega_n$.

For $t > 0$ they travel at constant velocities $0 > -c_1 > -c_2 > \cdots > -c_n$



If $i < j$ collide, then i is absorbed by the faster j, which continues at speed $-c_j$.



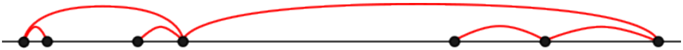
Define $A : [n-1] \to [n]$ such that particle i is absorbed by particle $A(i)$

Particles with locations and velocities

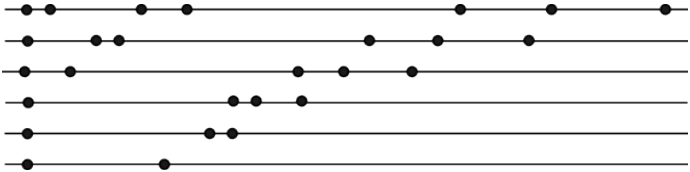
Consider $n \geq 2$ labelled particles on a real line.

At time $t = 0$, they are at locations $-\omega_1 \leq -\omega_2 \leq \cdots \leq -\omega_n$.

For $t > 0$ they travel at constant velocities $0 > -c_1 > -c_2 > \cdots > -c_n$



If $i < j$ collide, then i is absorbed by the faster j, which continues at speed $-c_j$.



Define $A : [n-1] \to [n]$ such that particle i is absorbed by particle $A(i)$

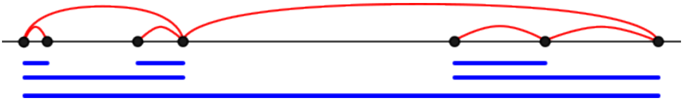
- A is precisely the max-slope arborescence for (c, ω).

Particles with locations and velocities

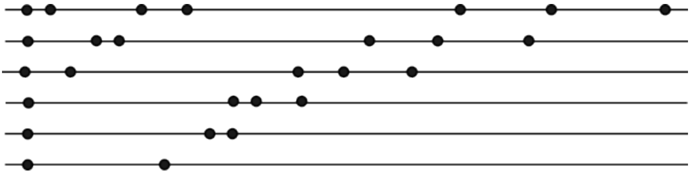
Consider $n \geq 2$ labelled particles on a real line.

At time $t = 0$, they are at locations $-\omega_1 \leq -\omega_2 \leq \cdots \leq -\omega_n$.

For $t > 0$ they travel at constant velocities $0 > -c_1 > -c_2 > \cdots > -c_n$



If $i < j$ collide, then i is absorbed by the faster j, which continues at speed $-c_j$.

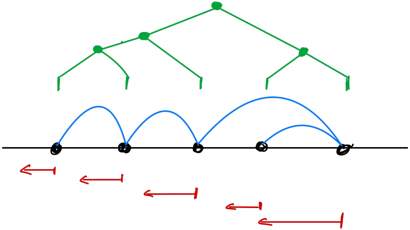


Define $A : [n-1] \to [n]$ such that particle i is absorbed by particle $A(i)$

- A is precisely the max-slope arborescence for (c, ω).
- the associated bracketing yields the isomorphism of face lattices

□

From linear programming to particle collisions



The Polyhedral Geometry of Pivot Rules and Monotone Paths
(Black, De Loera, Lütjeharms, S.), SIAGA 2023, arXiv:2201.05134

From linear programming to colliding particles
(A. Black, N. Lütjeharms, S.) arXiv:2405.08506

Applications of max-slope polytopes to (flag) matroids and flag varieties

Flag Polymatroids
(Black, S.), Adv. Math. 2024, arXiv:2207.12221