(Min,+) Formulas, Max, and Shortest Paths

Meena Mahajan

The Institute of Mathematical Sciences, HBNI, Chennai. 

Workshop on Algebraic Complexity Theory WACT; Paris, France.
5-9 March 2018

Credits

- joint work with
  Anuj Tawari (IMSc)
  Prajakta Nimbhorkar (CMI)
- shaped and influenced by email discussions with Stasys Jukna

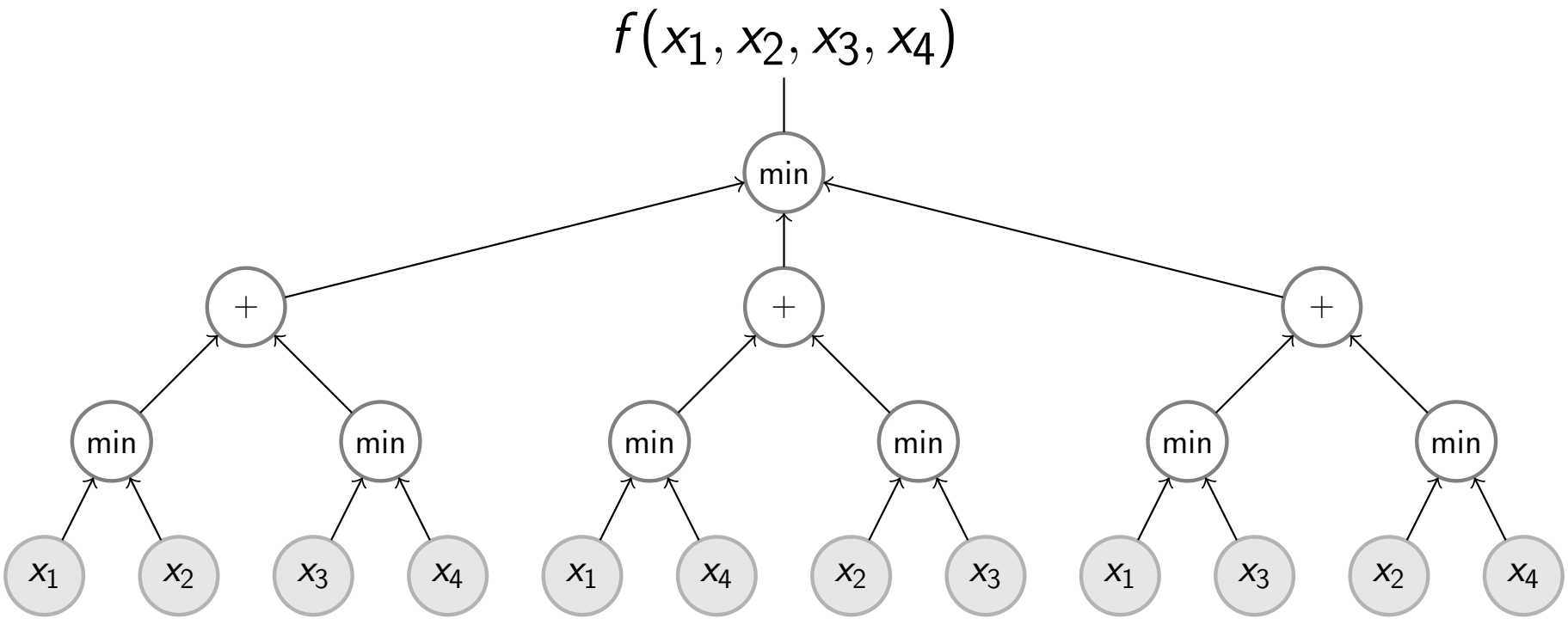
Computation Model

Model: Formulas/Circuits over the tropical semiring Min.

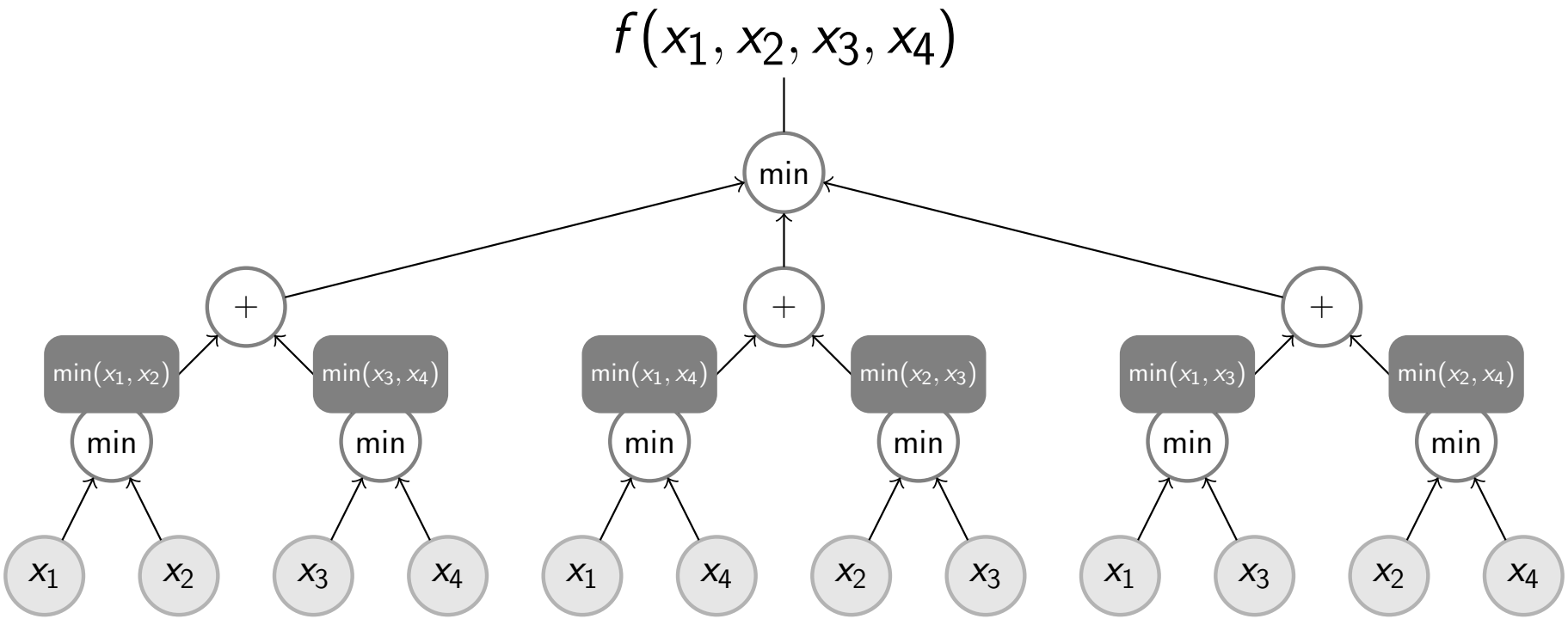
- Elements from the set $\mathbb{N} \cup \{\infty\}$
- Operation min; identity ∞.
- Operation $+$; identity 0.

The tropical semiring Min: $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$.
(An Idempotent (min), Commutative ($+$), semiring.)

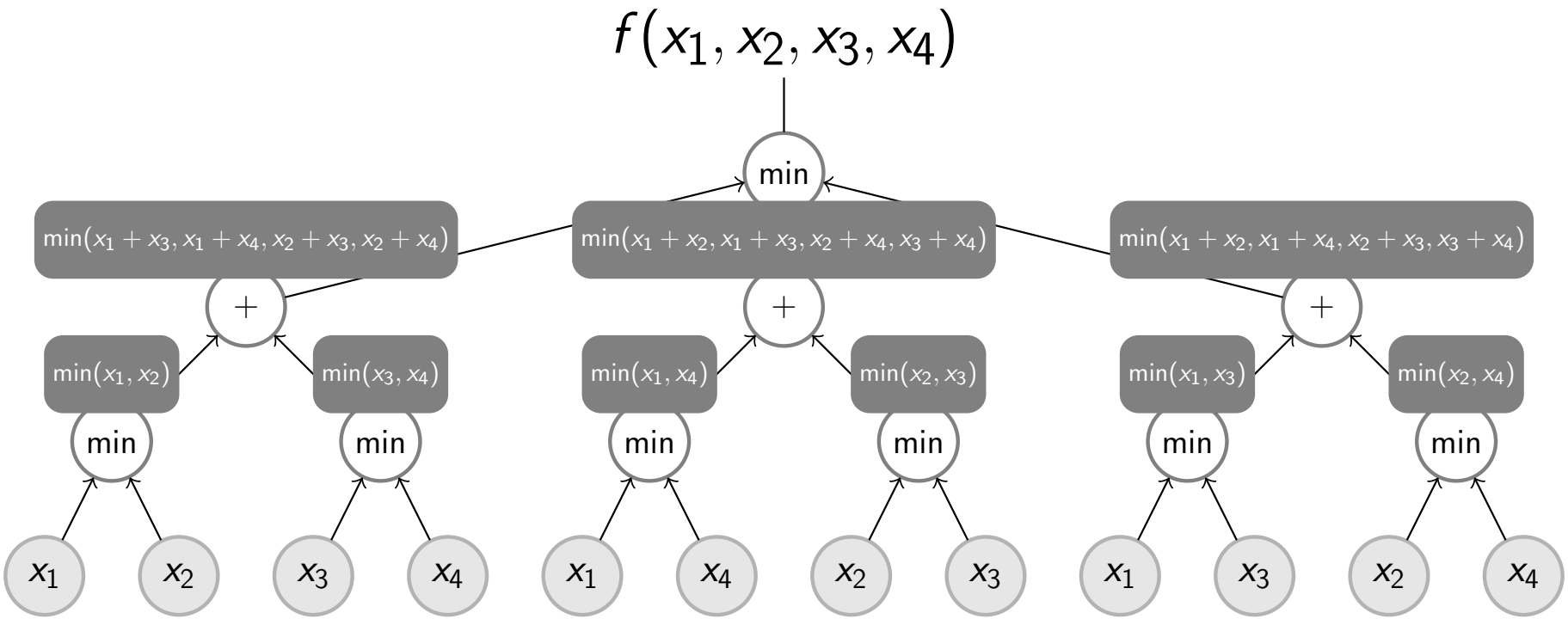
(min, +) formulas



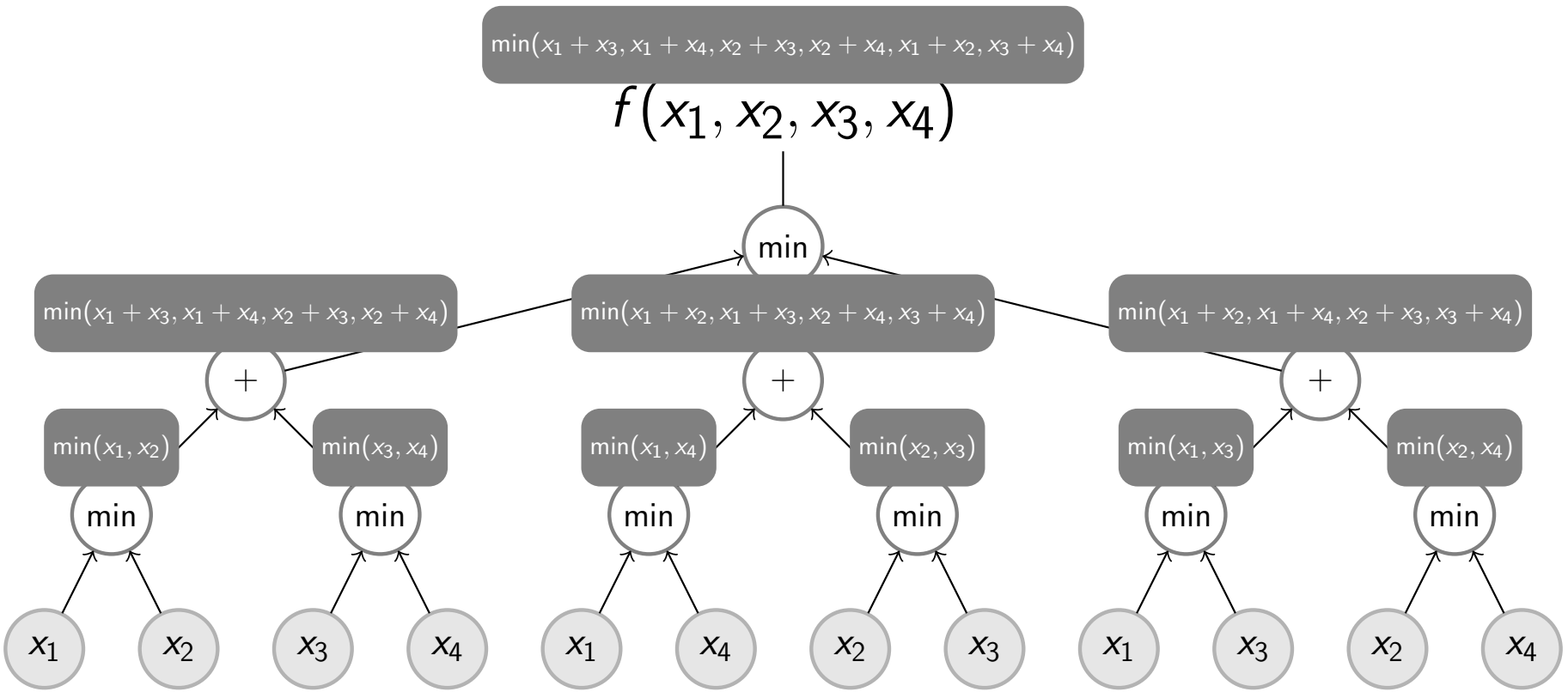
(min, +) formulas



$(\min, +)$ formulas



$(\min, +)$ formulas



What's different from computation over fields

- Neither min nor $+$ have inverses.
  No way to use cancellations for efficiency.

What's different from computation over fields

- Neither min nor $+$ have inverses.
  No way to use cancellations for efficiency.
- Idempotence: $\min(x, x) = x$.

What's different from computation over fields

- Neither min nor $+$ have inverses.
  No way to use cancellations for efficiency.
- Idempotence: $\min(x, x) = x$.
- Absorption: $\min(x, x + y) = x$.
  (as in Boolean semiring $(x \vee (x \wedge y) = x$.)
  Can be exploited to make computation efficient.

What's different from computation over fields

- Neither min nor $+$ have inverses.
  No way to use cancellations for efficiency.
- Idempotence: $\min(x, x) = x$.
- Absorption: $\min(x, x + y) = x$.
  (as in Boolean semiring $(x \vee (x \wedge y) = x$.)
  Can be exploited to make computation efficient.
- Polynomial "produced" by a circuit/formula can be very different from polynomial "computed".
  eg $\min(x, y, x + y, 2 + x, 3 + y)$ computes $\min(x, y)$.

  We can have f, g with different monomials/coefficients, but evaluating to the same values everywhere.

Why study the tropical semiring

- Natural model for many dynamic programming algorithms.
- Can help us understand the power and limitations of dynamic programming.
- Simulates monotone boolean formulas.

$$0 \to \infty; \quad 1 \to 1; \quad \vee \to \min; \quad \wedge \to \wedge$$

Can simulate without using ∞ too, slight blow-up.

$$x \wedge y = \min(x, y); \quad x \vee y = \min(1, x + y).$$

What we can do

Shortest Path computations.

Problem	Upper Bound	Lower bound
Single-Source-Shortest-Path (SSSP)	$O(n^3)$ size $(\min, +)$ circuits [BF62], [M59]	$\Omega(n^3)$ size $(\min, +)$ BP [JS16]
All-Pairs-Shortest-Path (APSP)	$O(n^3)$ size $(\min, +)$ circuits [F62],[W62]	$\Omega(n^3)$ size $(\min, +)$ circuits [J16]
Travelling-Salesman-Problem (TSP)	$O(2^n \cdot n^2)$ size $(\min, +)$ circuits [HK62]	$\Omega(2^n \cdot n^2)$ size $(\min, +)$ circuits [JS82]

What we can't do

$(\min, +)$ formulas not a universal model for monotone functions on $\mathbb{N}$.

- Longest path computations.
- Diameter of a graph.
- The maximum function.

Why we can't compute max

- Min is concave; max is convex; $+$ is linear (convex and concave).
- Suppose f computable by $(\min, +)$ circuit. Then

$$2f(\tilde{x} + \tilde{y}) \geq f(2\tilde{x}) + f(2\tilde{y})$$

- But

$$2\max((0,1) + (1,0)) = 2\max(1,1) = 2 \not\geq 4 = \max(0,2) + \max(2,0)$$

What we can't do (cont'd)

- Computing max over bounded squares?

What we can't do (cont'd)

- Computing max over bounded squares?
- Over $\{0, 1\}$: $\max\{a, b\} = \min\{1, a + b\}$.

What we can't do (cont'd)

- Computing max over bounded squares?
- Over $\{0, 1\}$: $\max\{a, b\} = \min\{1, a + b\}$.
- Over $\{0, 1, 2\}$? $\{0, 1, \ldots, k\}$? Using larger constants?

What we can't do (cont'd)

- Computing max over bounded squares?
- Over $\{0, 1\}$: $\max\{a, b\} = \min\{1, a + b\}$.
- Over $\{0, 1, 2\}$? $\{0, 1, \ldots, k\}$? Using larger constants?
  Not possible: similar convex/concave argument.
- Over $\{0, 1\}$, use of the constant 1 crucial.

Using subtraction to compute max

- Linear size formula: $\max(x_1, \ldots, x_n) = -\min(-x_1, \ldots, -x_n)$.

Using subtraction to compute max

- Linear size formula: $\max(x_1, \ldots, x_n) = -\min(-x_1, \ldots, -x_n)$.
  Not satisfactory: intermediate values negative outside the semiring.

Using subtraction to compute max

- Linear size formula: $\max(x_1, \ldots, x_n) = -\min(-x_1, \ldots, -x_n)$.
  Not satisfactory: intermediate values negative outside the semiring.
- Another formulation:

$$\max(x_1, \ldots, x_n) = \mathrm{Sum}_n - \mathrm{MinSum}_n^{n-1} = \sum_{i \in [n]} x_i - \min_{i \in [n]} \sum_{j \in [n] \setminus \{i\}} x_j$$

  Linear size circuits. Formula size $\Theta(n \log n)$.

Using subtraction to compute max

- Linear size formula: $\max(x_1, \ldots, x_n) = -\min(-x_1, \ldots, -x_n)$.
  Not satisfactory: intermediate values negative outside the semiring.
- Another formulation:

$$\max(x_1, \ldots, x_n) = \mathrm{Sum}_n - \mathrm{MinSum}_n^{n-1} = \sum_{i \in [n]} x_i - \min_{i \in [n]} \sum_{j \in [n] \setminus \{i\}} x_j$$

  Linear size circuits. Formula size $\Theta(n \log n)$.
- How do we know $\Omega(n \log n)$?
  Suppose F computes MinSum_n^{n-1}.
  Transform $F \longrightarrow G$: constants $\to 0$; min $\to \vee$; $+ \to \wedge$.
  Then G computes Th_n^{n-1}.
  Lower bound known for Monotone Boolean formulas for Th_n^{n-1}:
  $\Omega(n \log n)$: [Hansel64], [Krichevskii64]; $n \log n$: [JRadhakrishnan97].

Using subtraction to compute max

- Linear size formula: $\max(x_1, \ldots, x_n) = -\min(-x_1, \ldots, -x_n)$.
  Not satisfactory: intermediate values negative outside the semiring.
- Another formulation:

$$\max(x_1, \ldots, x_n) = \text{Sum}_n - \text{MinSum}_n^{n-1} = \sum_{i \in [n]} x_i - \min_{i \in [n]} \sum_{j \in [n] \setminus \{i\}} x_j$$

  Linear size circuits. Formula size $\Theta(n \log n)$.
- How do we know $\Omega(n \log n)$?
  Suppose F computes MinSum_n^{n-1}.
  Transform $F \longrightarrow G$: constants $\to 0$; min $\to \vee$; $+ \to \wedge$.
  Then G computes Th_n^{n-1}.
  Lower bound known for Monotone Boolean formulas for Th_n^{n-1}:
  $\Omega(n \log n)$: [Hansel64], [Krichevskii64]; $n \log n$: [JRadhakrishnan97].
- Any other formulation?

Expressing max as the difference of $(\min, +)$ formulas

Theorem

Let F_1 and F_2 be $(\min, +)$ formulas such that

$$\textit{for all } \tilde{x} \in \mathbb{N}^n,\ F_1 - F_2 \textit{ computes } \max(x_1, x_2, \ldots, x_n).$$

Then $|F_1| \geq n$, $|F_2| \geq n \log n$.

Proof Sketch

- Remove *common part* of F_1, F_2 without increasing size, to get G_1, G_2. (Find parse tree with only common variable; set these leaves to 0.)

Proof Sketch

- Remove *common part* of F_1, F_2 without increasing size, to get G_1, G_2. (Find parse tree with only common variable; set these leaves to 0.)
- Now max equals $G_1 - G_2$.

Proof Sketch

- Remove *common part* of F_1, F_2 without increasing size, to get G_1, G_2. (Find parse tree with only common variable; set these leaves to 0.)
- Now max equals $G_1 - G_2$.
- Easy to show: $G_1 \geq \mathrm{Sum}_n$.
  i.e. G_1 computes some function of the form $g_1' + \mathrm{Sum}_n$.

Proof Sketch

- Remove *common part* of F_1, F_2 without increasing size, to get G_1, G_2. (Find parse tree with only common variable; set these leaves to 0.)
- Now max equals $G_1 - G_2$.
- Easy to show: $G_1 \geq \mathrm{Sum}_n$.
  i.e. G_1 computes some function of the form $g_1' + \mathrm{Sum}_n$.
- If $g_1' = 0$, then G_2 must compute MinSum_n^{n-1}, and known lower bound applies.

Proof Sketch

- Remove *common part* of F_1, F_2 without increasing size, to get G_1, G_2. (Find parse tree with only common variable; set these leaves to 0.)
- Now max equals $G_1 - G_2$.
- Easy to show: $G_1 \geq \mathrm{Sum}_n$.
  i.e. G_1 computes some function of the form $g_1' + \mathrm{Sum}_n$.
- If $g_1' = 0$, then G_2 must compute MinSum_n^{n-1}, and known lower bound applies.
- g_1' need not be 0.
  Then we don't know what G_2 computes.

Proof Sketch

- Remove *common part* of F_1, F_2 without increasing size, to get G_1, G_2. (Find parse tree with only common variable; set these leaves to 0.)
- Now max equals $G_1 - G_2$.
- Easy to show: $G_1 \geq \mathrm{Sum}_n$.
  i.e. G_1 computes some function of the form $g_1' + \mathrm{Sum}_n$.
- If $g_1' = 0$, then G_2 must compute MinSum_n^{n-1}, and known lower bound applies.
- g_1' need not be 0.
  Then we don't know what G_2 computes.
- $G_2 \longrightarrow H$: zero out all constants at leaves.
  We don't know what H computes.

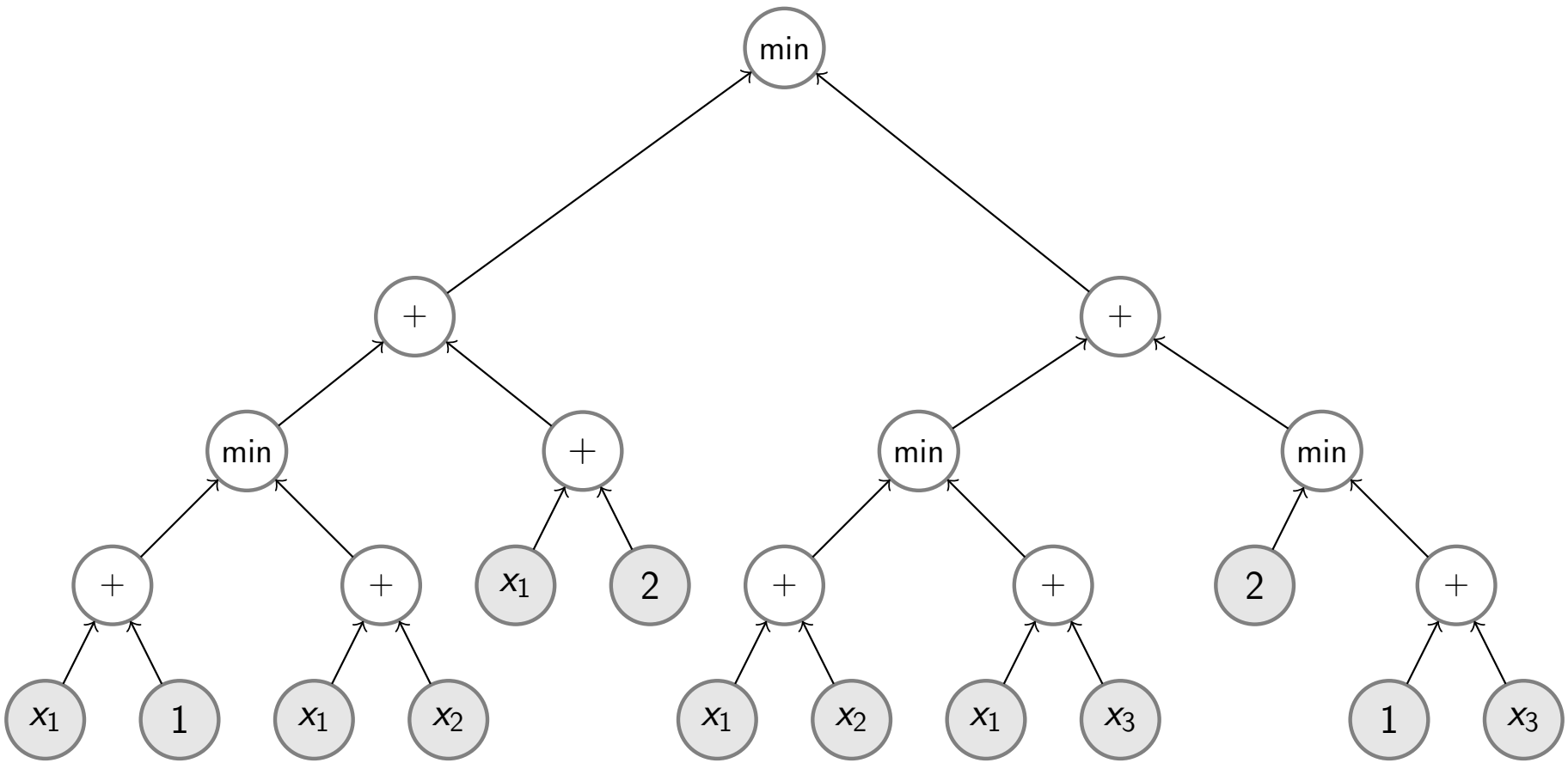
Proof Sketch

- Remove *common part* of F_1, F_2 without increasing size, to get G_1, G_2. (Find parse tree with only common variable; set these leaves to 0.)
- Now max equals $G_1 - G_2$.
- Easy to show: $G_1 \geq \mathrm{Sum}_n$.
  i.e. G_1 computes some function of the form $g_1' + \mathrm{Sum}_n$.
- If $g_1' = 0$, then G_2 must compute MinSum_n^{n-1}, and known lower bound applies.
- g_1' need not be 0.
  Then we don't know what G_2 computes.
- $G_2 \longrightarrow H$: zero out all constants at leaves.
  We don't know what H computes.
- Show nice properties of H, apply a complexity measure.

Removing common part

$$\min\{2x_1 + 3, 2x_1 + x_2 + 2, x_1 + x_2 + 2, x_1 + x_2 + x_3 + 1, x_1 + x_3 + 2, x_1 + 2x_3 + 1\}$$

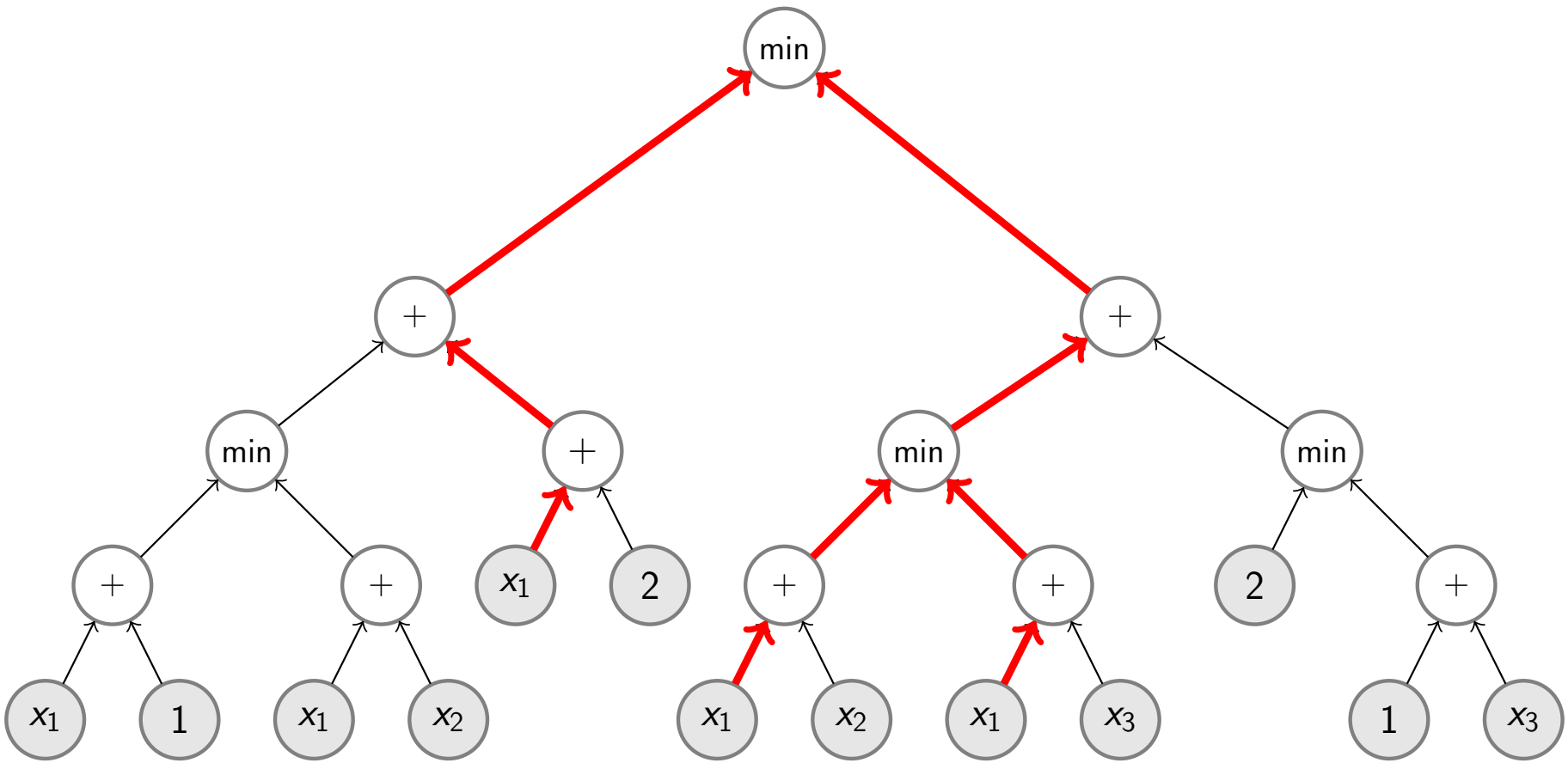
$$= x_1 + \min\{x_1 + 3, x_1 + x_2 + 2, x_2 + 2, x_2 + x_3 + 1, x_3 + 2, 2x_3 + 1\}$$



Removing common part

$$\min\{2x_1 + 3, 2x_1 + x_2 + 2, x_1 + x_2 + 2, x_1 + x_2 + x_3 + 1, x_1 + x_3 + 2, x_1 + 2x_3 + 1\}$$

$$= x_1 + \min\{x_1 + 3, x_1 + x_2 + 2, x_2 + 2, x_2 + x_3 + 1, x_3 + 2, 2x_3 + 1\}$$



Properties of H

$$\max = G_1 - G_2.$$

G_1 and G_2 have no common part.

H is constant-free part of G_2, computes h.

$$\Downarrow$$

Properties of H

$$\max = G_1 - G_2.$$

G_1 and G_2 have no common part.

H is constant-free part of G_2, computes h.

$$\Downarrow$$

$$\forall i \in [n],\ h(e_i) = 0.$$

$$\forall i, j \in [n] \text{ with } i \neq j,\ h(e_i + e_j) \geq 1.$$

Properties of H

$$\max = G_1 - G_2.$$

G_1 and G_2 have no common part.

H is constant-free part of G_2, computes h.

$$\Downarrow$$

$$\forall i \in [n],\ h(e_i) = 0.$$

$$\forall i, j \in [n] \text{ with } i \neq j,\ h(e_i + e_j) \geq 1.$$

Analogous monotone Boolean case:
no singleton minterms, all size-2 minterms present. i.e. $\mathrm{Th}_{2,n}$.

Lower bound known. Mimic Boolean proof.

Properties of H

$$\max = G_1 - G_2.$$

G_1 and G_2 have no common part.

H is constant-free part of G_2, computes h.

$$\Downarrow$$

$$\forall i \in [n],\ h(e_i) = 0.$$

$$\forall i, j \in [n] \text{ with } i \neq j,\ h(e_i + e_j) \geq 1.$$

Analogous monotone Boolean case:
no singleton minterms, all size-2 minterms present. i.e. $\mathrm{Th}_{2,n}$.

Lower bound known. Mimic Boolean proof.
(Jukna: can even transfer Boolean lower bound.)

The complexity measure

$f : \mathbb{N}^n \to \mathbb{N}$ satisfying $f(\tilde{0}) = 0$.

$$\mu(f) = \frac{|\{i \in [n] \mid f(e_i) \geq 1\}|}{n} + \text{Entropy}(\text{Graph}(f))$$

The complexity measure

$f : \mathbb{N}^n \to \mathbb{N}$ satisfying $f(\tilde{0}) = 0$.

$$\mu(f) = \frac{|\{i \in [n] \mid f(e_i) \geq 1\}|}{n} + \text{Entropy}(\text{Graph}(f))$$

- $\text{Graph}(f) = ([n], E)$ where

$$E = \{(i,j) \mid f(e_i + e_j) \geq 1,\ f(e_i) = 0,\ f(e_j) = 0\}$$

The complexity measure

$f : \mathbb{N}^n \to \mathbb{N}$ satisfying $f(\tilde{0}) = 0$.

$$\mu(f) = \frac{|\{i \in [n] \mid f(e_i) \geq 1\}|}{n} + \text{Entropy}(\text{Graph}(f))$$

- $\text{Graph}(f) = ([n], E)$ where

$$E = \{(i,j) \mid f(e_i + e_j) \geq 1,\ f(e_i) = 0,\ f(e_j) = 0\}$$

- Vertex Packing Polytope of a graph $G = (V, E)$:

$$VP(G) = \text{ConvexHull}\,\{\tilde{a} \in \{0,1\}^n \mid \tilde{a} \text{ indicates an independent set}\}$$

- Entropy of a graph $G = (V, E)$ (w.r.t. uniform distribution):

$$H(G) = \min_{\tilde{a} \in VP(G)} \sum_{i=1}^{n} \frac{1}{n} \log \frac{1}{a_i}$$

Graph Entropy

- First defined by [Korner73] with respect to a probability distribution.
- Equivalent characterizations used in [Korner86, Korner-Marton88, Csiszar-Korner-Lovasz-Marton-Simonyi90, Simonyi95, Simonyi01]

Graph Entropy

- First defined by [Korner73] with respect to a probability distribution.
- Equivalent characterizations used in [Korner86, Korner-Marton88, Csiszar-Korner-Lovasz-Marton-Simonyi90, Simonyi95, Simonyi01]
- For graphs F, G on same vertex set,
  - $E(F) \subseteq E(G) \Rightarrow H(F) \leq H(G)$. Monotonicity.
  - $H(F \cup G) \leq H(F) + H(G)$. Sub-additivity.
- $H(V, \emptyset) = 0$.
- $H(K_n) = \log n$.
- If $|V(G)| = n$, and $E(G)$ induces a bipartite graph on m vertices, then $H(G) \leq \frac{m}{n}$. (In particular, $H(K_{n,n}) = 1$.)

Using the Complexity Measure

$$\mu(f) = \frac{|\{i \in [n] \mid f(e_i) \geq 1\}|}{n} + \text{Entropy}(\text{Graph}(f))$$

Using the Complexity Measure

$$\mu(f) = \frac{|\{i \in [n] \mid f(e_i) \geq 1\}|}{n} + \text{Entropy}(\text{Graph}(f))$$

For monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$,

Theorem (NewmanWigderson95)

$$\textit{Monotone-Boolean-Formula-Leaf-size}(f) \geq n \cdot \mu(f)$$

Using the Complexity Measure

$$\mu(f) = \frac{|\{i \in [n] \mid f(e_i) \geq 1\}|}{n} + \text{Entropy}(\text{Graph}(f))$$

For monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$,

Theorem (NewmanWigderson95)

$$\textit{Monotone-Boolean-Formula-Leaf-size}(f) \geq n \cdot \mu(f)$$

Works for $(\min, .+)$ semiring too; for $f : \mathbb{N}^n \to \mathbb{N}$,

Theorem

$$\textit{Constant-free-}(\min, +)\textit{-Formula-Leaf-size}(f) \geq n \cdot \mu(f)$$

Wrapping up Proof

max $= F_1 - F_2$.

- Remove *common part* of F_1, F_2 without increasing size, to get G_1, G_2.
- Now max equals $G_1 - G_2$.
- Easy to show: $G_1 \geq \text{Sum}_n$. Hence Leaf-Size$(G_1) \geq n$.
- $G_2 \longrightarrow H$: zero out all constants at leaves.
- We don't know what H computes. But,
  We can show $h(e_i) = 0$ for all i, and $\text{Graph}(h) = K_n$.
  Hence $\mu(h) = \log n$ and $\text{Leaf-Size}(H) \geq n \log n$.

More about $(\min, +)$ computation

Interpreting a polynomial in different (semi)rings:

Arithmetic	$(\min, +)$	Boolean
$x + yz$	$\min(x, y + z)$	$x \vee yz$
$x^2 + yz$	$\min(2x, y + z)$	$xx \vee yz$ $\equiv x \vee yz$
$x + xy + yz$	$\min(x, x + y, y + z)$ $\equiv \min(x, y + z)$	$x \vee xy \vee yz$ $\equiv x \vee yz$
$x + x^3$	$\min\{x, 3x\} \equiv x$	$x \vee xxx \equiv x$
$x + 2$	$\min\{x, 2\}$	$x \vee 1 \vee 1 \equiv 1$

More about $(\min, +)$ computation

Interpreting a polynomial in different (semi)rings:

Arithmetic	$(\min, +)$	Boolean
$x + yz$	$\min(x, y + z)$	$x \vee yz$
$x^2 + yz$	$\min(2x, y + z)$	$xx \vee yz$ $\equiv x \vee yz$
$x + xy + yz$	$\min(x, x + y, y + z)$ $\equiv \min(x, y + z)$	$x \vee xy \vee yz$ $\equiv x \vee yz$
$x + x^3$	$\min\{x, 3x\} \equiv x$	$x \vee xxx \equiv x$
$x + 2$	$\min\{x, 2\}$	$x \vee 1 \vee 1 \equiv 1$

For multilinear polynomials with 0-1 coefficients, computing cost in monotone-Boolean $\leq (\min, +) \leq$ monotone-arithmetic.

Computing versus Producing

- A circuit syntactically "produces" a polynomial, computes a function.
- Computing and producing equivalent for monotone arithmetic setting.

Computing versus Producing

- A circuit syntactically "produces" a polynomial, computes a function.
- Computing and producing equivalent for monotone arithmetic setting.
- Not so in the tropical semiring; provably huge difference between computing and producing.

$$\text{stconn}_n = \min_{\rho: s \rightsquigarrow t \text{ path}} \sum_{e \in \rho} x_e$$

$$\text{walk}_n = \min_{\rho: s \rightsquigarrow t \text{ walk of length } < n} \sum_{e \in \rho} (\text{multiplicity of } e \text{ in } \rho)\, x_e$$

Equivalent for computing, easy $O(n^3)$.
Very different for producing: walk_n easy;
stconn_n needs size $2^{\Omega(n)}$ (Jukna2015).

Why producing stconn hard

- Cost of producing polynomial same over $(\mathbb{N}, \min, +)$ or $(\mathbb{Q}, +, \times)$.

Why producing stconn hard

- Cost of producing polynomial same over $(\mathbb{N}, \min, +)$ or $(\mathbb{Q}, +, \times)$.
- Consider monotone arithmetic circuit C_1 producing stconn, of size s.

Why producing stconn hard

- Cost of producing polynomial same over $(\mathbb{N}, \min, +)$ or $(\mathbb{Q}, +, \times)$.
- Consider monotone arithmetic circuit C_1 producing stconn, of size s.
- Extract circuit C_2, size $\leq s$, computing max-degree homogeneous part stconn_{he}. (higher envelope; only full length paths.)

Why producing stconn hard

- Cost of producing polynomial same over $(\mathbb{N}, \min, +)$ or $(\mathbb{Q}, +, \times)$.
- Consider monotone arithmetic circuit C_1 producing stconn, of size s.
- Extract circuit C_2, size $\leq s$, computing max-degree homogeneous part stconn_{he}. (higher envelope; only full length paths.)
- Express stconn_{he} as $\sum_{i=1}^{t} g_i \times h_i$, where $t \leq s$ and $n/3 < \text{degree}(g_i) \leq 2n/3$.

Why producing stconn hard

- Cost of producing polynomial same over $(\mathbb{N}, \min, +)$ or $(\mathbb{Q}, +, \times)$.
- Consider monotone arithmetic circuit C_1 producing stconn, of size s.
- Extract circuit C_2, size $\leq s$, computing max-degree homogeneous part stconn_{he}. (higher envelope; only full length paths.)
- Express stconn_{he} as $\sum_{i=1}^{t} g_i \times h_i$, where $t \leq s$ and $n/3 < \text{degree}(g_i) \leq 2n/3$.
- $$n! = \#\text{monomials}(\text{stconn}_{he}) \leq \sum_{i=1}^{t} \#\text{monomials in } g_i \times h_i$$
- A degree d monomial has at most $(n-d)!$ extensions in stconn_{he}. So

$$\#\text{monomials in } g_i \times h_i \leq \max_{n/3 < a \leq 2n/3} (n-a)!a! \leq (n/3)!(2n/3)!)$$

Why producing stconn hard

- Cost of producing polynomial same over $(\mathbb{N}, \min, +)$ or $(\mathbb{Q}, +, \times)$.
- Consider monotone arithmetic circuit C_1 producing stconn, of size s.
- Extract circuit C_2, size $\leq s$, computing max-degree homogeneous part stconn_{he}. (higher envelope; only full length paths.)
- Express stconn_{he} as $\sum_{i=1}^{t} g_i \times h_i$, where $t \leq s$ and $n/3 < \text{degree}(g_i) \leq 2n/3$.
- $$n! = \#\text{monomials}(\text{stconn}_{he}) \leq \sum_{i=1}^{t} \#\text{monomials in } g_i \times h_i$$
- A degree d monomial has at most $(n-d)!$ extensions in stconn_{he}. So $$\#\text{monomials in } g_i \times h_i \leq \max_{n/3 < a \leq 2n/3} (n-a)!a! \leq (n/3)!(2n/3)!)$$
- $s \geq t \geq \binom{n}{n/3} = \exp(\Omega(n))$.

Computing versus Producing ... 2

- Computing stconn_n: easy in $(\min, +)$ semiring,
  hard in monotone arithmetic semiring.
- This is only because stconn_n is not homogeneous.
- Recall, for multilinear polynomials with 0-1 coefficients, computing cost in
  monotone-Boolean $\leq (\min, +) \leq$ monotone-arithmetic.
- For homogeneous multilinear polynomials with 0-1 coefficients, computing cost same in $(\min, +)$ and monotone-arithmetic. (Jukna2015)

Computing versus Producing ... 2

- Computing stconn_n: easy in $(\min, +)$ semiring,
  hard in monotone arithmetic semiring.
- This is only because stconn_n is not homogeneous.
- Recall, for multilinear polynomials with 0-1 coefficients, computing cost in
  monotone-Boolean $\leq (\min, +) \leq$ monotone-arithmetic.
- For homogeneous multilinear polynomials with 0-1 coefficients, computing cost same in $(\min, +)$ and monotone-arithmetic. (Jukna2015)
  (stconn_{he} is HamPath; hard in arithmetic and in $(\min, +)$.)

Computing stconn, revisited

- stcon has $(\min, +)$ circuits of size $O(n^3)$.
  Is $\Omega(n^3)$ necessary? Still open!
  Known to be necessary for tropical skew circuits. (JuknaSchnitger2016)

Computing stconn, revisited

- stcon has $(\min, +)$ circuits of size $O(n^3)$.
  Is $\Omega(n^3)$ necessary? Still open!
  Known to be necessary for tropical skew circuits. (JuknaSchnitger2016)
- stcon has $(\min, +)$ circuits of size $O(n^3)$, alternation depth $\Theta(\log n)$.
  What if we bound alternation depth?

Computing stconn with alternation depth $d \in O(1)$

- Exponential lower bound easy to show:

Computing stconn with alternation depth $d \in O(1)$

- Exponential lower bound easy to show:

$$(\min, +) \text{ depth-}d \text{ size-}s \text{ circuit for } \text{stcon}_{2n}$$
$$\Downarrow$$
$$\text{monotone depth-}d \text{ size-}s \text{ Boolean circuit for } \text{stcon}_{2n}$$
$$\Downarrow$$
$$\text{depth-}d \text{ size-}s \text{ Boolean circuit for } \text{Parity}_n.$$
$$\text{So } s \in \exp(\Omega(n^{\frac{1}{d-1}}))$$

Computing stconn with alternation depth $d \in O(1)$

- Exponential lower bound easy to show:

$$(\min, +) \text{ depth-}d \text{ size-}s \text{ circuit for } \text{stcon}_{2n}$$
$$\Downarrow$$
$$\text{monotone depth-}d \text{ size-}s \text{ Boolean circuit for } \text{stcon}_{2n}$$
$$\Downarrow$$
$$\text{depth-}d \text{ size-}s \text{ Boolean circuit for } \text{Parity}_n.$$
$$\text{So } s \in \exp(\Omega(n^{\frac{1}{d-1}}))$$

- Is this bound tight?
- Upper bound via divide-and-conquer:
  for even d, there is a circuit of size $s \in \exp(O(dn^{\frac{2}{d}} \log n))$.

Computing stconn with alternation depth d (cont'd)

Depth d	easy upper bound	lower bound from parity	lower bound

Computing stconn with alternation depth d (cont'd)

Depth d	easy upper bound	lower bound from parity	lower bound
2	$O(n!) = \exp(O(n \log n))$	$\exp(\Omega(n))$	$\exp(\Omega(n \log n))$

Computing stconn with alternation depth d (cont'd)

Depth d	easy upper bound	lower bound from parity	lower bound
2	$O(n!) = \exp(O(n \log n))$	$\exp(\Omega(n))$	$\exp(\Omega(n \log n))$
3	$O(n!) = \exp(O(n \log n))$	$\exp(\Omega(\sqrt{n}))$	$\exp(\Omega(n \log n))$

Computing stconn with alternation depth d (cont'd)

Depth d	easy upper bound	lower bound from parity	lower bound
2	$O(n!) = \exp(O(n \log n))$	$\exp(\Omega(n))$	$\exp(\Omega(n \log n))$
3	$O(n!) = \exp(O(n \log n))$	$\exp(\Omega(\sqrt{n}))$	$\exp(\Omega(n \log n))$
4	$\exp(O(\sqrt{n} \log n))$	$\exp(\Omega(n^{1/3}))$	???

Computing stconn with alternation depth 4

$\mathcal{G}=$ set of $+$ gates just below the root.
Partial result 1:

Theorem

If at most k gates in $\mathcal{G}$ have fanin > 2, then size $\exp(\Omega(\frac{n}{2^k}\log\frac{n}{2^k}))$.

Computing stconn with alternation depth 4

$\mathcal{G}$= set of $+$ gates just below the root.
Partial result 1:

Theorem

If at most k gates in $\mathcal{G}$ have fanin > 2, then size $\exp(\Omega(\frac{n}{2^k}\log\frac{n}{2^k}))$.

Proof: $L(n,r,k)$ = (leaf-)size of the smallest depth-4 formula that solves stconn_n, and where the number of $+$ gates at the second level with fan-in exceeding r is at most k. Show

- For $k \geq 1$, $L(n,r,k) \geq L(n(\frac{r-1}{r}), r, k-1)$. In particular,

$$L(n,2,k) \geq L\left(\frac{n}{2}, 2, k-1\right).$$

- If all gates in $\mathcal{G}$ have fanin ≤ 2, then size $\exp(\Omega(n\log n))$. ie

$$L(n,2,0) \geq 2^{\Omega(n\log n)}.$$

(no better than depth 2)

Computing stconn with alternation depth 4

Partial result 2:

Theorem

If F is a depth 4 formula for stconn with size $L(F)$, and all second level $+$ gates have fanin at most t, then

$$L(F) \geq \exp\left(\Omega\left(\frac{n \log n}{t}\right)\right)$$

Computing stconn with alternation depth 4

Partial result 2:

Theorem

If F is a depth 4 formula for stconn with size $L(F)$, and all second level $+$ gates have fanin at most t, then

$$L(F) \geq \exp\left(\Omega\left(\frac{n \log n}{t}\right)\right)$$

Lower bound tight at two extremes: $t = \sqrt{n}$, and $t = 2$.

Open Questions re tropical computation

- Tight lower bounds for problems with low tropical complexity.
- How much of machinery from monotone arithmetic circuits (over fields) can be ported?
- How much from monotone Boolean circuits?

Thank you