Algorithms for Submodular Function Minimization (SFMin)

S. Thomas McCormick

Sauder School of Business, UBC

Cargese Workshop on Combinatorial Optimization,
Sept–Oct 2013

Outline

The Greedy Algorithm

- Order the elements such that $w_1 \geq w_2 \geq \cdots \geq w_n$.

The Greedy Algorithm

- Order the elements such that $w_1 \geq w_2 \geq \cdots \geq w_n$.
  1. Make x_1 as large as possible: $x_1 \leftarrow f(\{e_1\}) - f(\emptyset)$.

The Greedy Algorithm

- Order the elements such that $w_1 \geq w_2 \geq \cdots \geq w_n$.
  1. Make x_1 as large as possible: $x_1 \leftarrow f(\{e_1\}) - f(\emptyset)$.
  2. Make x_2 as large as possible: $x_2 \leftarrow f(\{e_1, e_2\}) - f(\{e_1\})$.

The Greedy Algorithm

- Order the elements such that $w_1 \geq w_2 \geq \cdots \geq w_n$.
  1. Make x_1 as large as possible: $x_1 \leftarrow f(\{e_1\}) - f(\emptyset)$.
  2. Make x_2 as large as possible: $x_2 \leftarrow f(\{e_1, e_2\}) - f(\{e_1\})$.
  3. Make x_3 as large as possible: $x_3 \leftarrow f(\{e_1, e_2, e_3\}) - f(\{e_1, e_2\})$

The Greedy Algorithm

- Order the elements such that $w_1 \geq w_2 \geq \cdots \geq w_n$.
  1. Make x_1 as large as possible: $x_1 \leftarrow f(\{e_1\}) - f(\emptyset)$.
  2. Make x_2 as large as possible: $x_2 \leftarrow f(\{e_1, e_2\}) - f(\{e_1\})$.
  3. Make x_3 as large as possible: $x_3 \leftarrow f(\{e_1, e_2, e_3\}) - f(\{e_1, e_2\})$
  4. Etc, etc ...

The Greedy Algorithm

- Order the elements such that $w_1 \geq w_2 \geq \cdots \geq w_n$.
  1. Make x_1 as large as possible: $x_1 \leftarrow f(\{e_1\}) - f(\emptyset)$.
  2. Make x_2 as large as possible: $x_2 \leftarrow f(\{e_1, e_2\}) - f(\{e_1\})$.
  3. Make x_3 as large as possible: $x_3 \leftarrow f(\{e_1, e_2, e_3\}) - f(\{e_1, e_2\})$
  4. Etc, etc ...
- Notice that this Greedy Algorithm depends only on the input linear order. We derived the order from w, but we could apply the same algorithm to any order $\prec$.

The Greedy Algorithm

- Order the elements such that $w_1 \geq w_2 \geq \cdots \geq w_n$.
    1. Make x_1 as large as possible: $x_1 \leftarrow f(\{e_1\}) - f(\emptyset)$.
    2. Make x_2 as large as possible: $x_2 \leftarrow f(\{e_1, e_2\}) - f(\{e_1\})$.
    3. Make x_3 as large as possible: $x_3 \leftarrow f(\{e_1, e_2, e_3\}) - f(\{e_1, e_2\})$
    4. Etc, etc ...
- Notice that this Greedy Algorithm depends only on the input linear order. We derived the order from w, but we could apply the same algorithm to any order $\prec$.
- Given linear order $\prec$ and $e \in E$, define $e^{\prec} = \{g \in E \mid g \prec e\}$.

The Greedy Algorithm

- Order the elements such that $w_1 \geq w_2 \geq \cdots \geq w_n$.
  1. Make x_1 as large as possible: $x_1 \leftarrow f(\{e_1\}) - f(\emptyset)$.
  2. Make x_2 as large as possible: $x_2 \leftarrow f(\{e_1, e_2\}) - f(\{e_1\})$.
  3. Make x_3 as large as possible: $x_3 \leftarrow f(\{e_1, e_2, e_3\}) - f(\{e_1, e_2\})$
  4. Etc, etc ...
- Notice that this Greedy Algorithm depends only on the input linear order. We derived the order from w, but we could apply the same algorithm to any order $\prec$.
- Given linear order $\prec$ and $e \in E$, define $e^{\prec} = \{g \in E \mid g \prec e\}$.
  - E.g., suppose that
    $\prec_1$ is $3 \prec_1 1 \prec_1 4 \prec_1 5 \prec_1 2$ and
    $\prec_2$ is $1 \prec_2 2 \prec_2 3 \prec_2 4 \prec_2 5$.

The Greedy Algorithm

- Order the elements such that $w_1 \geq w_2 \geq \cdots \geq w_n$.
  1. Make x_1 as large as possible: $x_1 \leftarrow f(\{e_1\}) - f(\emptyset)$.
  2. Make x_2 as large as possible: $x_2 \leftarrow f(\{e_1, e_2\}) - f(\{e_1\})$.
  3. Make x_3 as large as possible: $x_3 \leftarrow f(\{e_1, e_2, e_3\}) - f(\{e_1, e_2\})$
  4. Etc, etc ...
- Notice that this Greedy Algorithm depends only on the input linear order. We derived the order from w, but we could apply the same algorithm to any order $\prec$.
- Given linear order $\prec$ and $e \in E$, define $e^{\prec} = \{g \in E \mid g \prec e\}$.
  - E.g., suppose that
    $\prec_1$ is $3 \prec_1 1 \prec_1 4 \prec_1 5 \prec_1 2$ and
    $\prec_2$ is $1 \prec_2 2 \prec_2 3 \prec_2 4 \prec_2 5$.
  - Then $3^{\prec_1} = \emptyset$, $3^{\prec_2} = \{1, 2\}$,
    and $2^{\prec_1} = \{1, 3, 4, 5\}$, $2^{\prec_2} = \{1\}$.

The Greedy Algorithm

- Order the elements such that $w_1 \geq w_2 \geq \cdots \geq w_n$.
  1. Make x_1 as large as possible: $x_1 \leftarrow f(\{e_1\}) - f(\emptyset)$.
  2. Make x_2 as large as possible: $x_2 \leftarrow f(\{e_1, e_2\}) - f(\{e_1\})$.
  3. Make x_3 as large as possible: $x_3 \leftarrow f(\{e_1, e_2, e_3\}) - f(\{e_1, e_2\})$
  4. Etc, etc ...
- Notice that this Greedy Algorithm depends only on the input linear order. We derived the order from w, but we could apply the same algorithm to any order $\prec$.
- Given linear order $\prec$ and $e \in E$, define $e^{\prec} = \{g \in E \mid g \prec e\}$.
  - E.g., suppose that
    $\prec_1$ is $3 \prec_1 1 \prec_1 4 \prec_1 5 \prec_1 2$ and
    $\prec_2$ is $1 \prec_2 2 \prec_2 3 \prec_2 4 \prec_2 5$.
  - Then $3^{\prec_1} = \emptyset$, $3^{\prec_2} = \{1, 2\}$,
    and $2^{\prec_1} = \{1, 3, 4, 5\}$, $2^{\prec_2} = \{1\}$.
- In this notation we can re-express the main step of Greedy on the ith element in $\prec$ as
  "Make $x_{e_i} \leftarrow f(e_i^{\prec} + e_i) - f(e_i^{\prec})$."

The Greedy Algorithm produces a feasible x

- We now prove that the x computed by Greedy belongs to $B(f)$ as follows:

The Greedy Algorithm produces a feasible x

- We now prove that the x computed by Greedy belongs to $B(f)$ as follows:
  - Index the elements such that $\prec$ is $e_1 \prec e_2 \prec \cdots \prec e_n$. First, $x(E) = \sum_{e_i \in E}[f(e_i^{\prec} + e_i) - f(e_i^{\prec})] = f(E) - f(\emptyset) = f(E)$.

The Greedy Algorithm produces a feasible x

- We now prove that the x computed by Greedy belongs to $B(f)$ as follows:
  - Index the elements such that $\prec$ is $e_1 \prec e_2 \prec \cdots \prec e_n$. First, $x(E) = \sum_{e_i \in E}[f(e_i^{\prec} + e_i) - f(e_i^{\prec})] = f(E) - f(\emptyset) = f(E)$.
  - Now for any $\emptyset \subset S \subset E$ we need to verify that $x(S) \leq f(S)$. Define k as the largest index such that $e_k \in S$, and use induction on k.

The Greedy Algorithm produces a feasible x

- We now prove that the x computed by Greedy belongs to $B(f)$ as follows:
  - Index the elements such that $\prec$ is $e_1 \prec e_2 \prec \cdots \prec e_n$. First, $x(E) = \sum_{e_i \in E}[f(e_i^{\prec} + e_i) - f(e_i^{\prec})] = f(E) - f(\emptyset) = f(E)$.
  - Now for any $\emptyset \subset S \subset E$ we need to verify that $x(S) \leq f(S)$. Define k as the largest index such that $e_k \in S$, and use induction on k.
  - If $k = 1$ then $S = \{e_1\}$ and $x_1 = f(e_1^{\prec} + e_1) - f(e_1^{\prec}) = f(\{e_1\}) - f(\emptyset) = f(S)$.

The Greedy Algorithm produces a feasible x

- We now prove that the x computed by Greedy belongs to $B(f)$ as follows:
  - Index the elements such that $\prec$ is $e_1 \prec e_2 \prec \cdots \prec e_n$. First, $x(E) = \sum_{e_i \in E}[f(e_i^{\prec} + e_i) - f(e_i^{\prec})] = f(E) - f(\emptyset) = f(E)$.
  - Now for any $\emptyset \subset S \subset E$ we need to verify that $x(S) \leq f(S)$. Define k as the largest index such that $e_k \in S$, and use induction on k.
  - If $k = 1$ then $S = \{e_1\}$ and $x_1 = f(e_1^{\prec} + e_1) - f(e_1^{\prec}) = f(\{e_1\}) - f(\emptyset) = f(S)$.
  - If $k > 1$, then $S \cup e_k^{\prec} = e_{k+1}^{\prec}$ and $S \cap e_k^{\prec} = S - e_k$. Then submodularity implies that $f(S) \geq f(S \cup e_k^{\prec}) + f(S \cap e_k^{\prec}) - f(e_k^{\prec}) = f(e_{k+1}^{\prec}) + f(S - e_k) - f(e_k^{\prec})$.

The Greedy Algorithm produces a feasible x

- We now prove that the x computed by Greedy belongs to $B(f)$ as follows:
  - Index the elements such that $\prec$ is $e_1 \prec e_2 \prec \cdots \prec e_n$. First, $x(E) = \sum_{e_i \in E}[f(e_i^{\prec} + e_i) - f(e_i^{\prec})] = f(E) - f(\emptyset) = f(E)$.
  - Now for any $\emptyset \subset S \subset E$ we need to verify that $x(S) \leq f(S)$. Define k as the largest index such that $e_k \in S$, and use induction on k.
  - If $k = 1$ then $S = \{e_1\}$ and $x_1 = f(e_1^{\prec} + e_1) - f(e_1^{\prec}) = f(\{e_1\}) - f(\emptyset) = f(S)$.
  - If $k > 1$, then $S \cup e_k^{\prec} = e_{k+1}^{\prec}$ and $S \cap e_k^{\prec} = S - e_k$. Then submodularity implies that $f(S) \geq f(S \cup e_k^{\prec}) + f(S \cap e_k^{\prec}) - f(e_k^{\prec}) =$ $f(e_{k+1}^{\prec}) + f(S - e_k) - f(e_k^{\prec})$.
  - The largest e_i in $S - e_k$ is smaller than k, so induction applies to $S - e_k$ and we get $x(S) - x_{e_k} = x(S - e_k) \leq f(S - e_k)$, or $x(S) \leq f(S - e_k) + x_{e_k} = f(S - e_k) + (f(e_k^{\prec} + e_k) - f(e_k^{\prec}))$.

The Greedy Algorithm produces a feasible x

- We now prove that the x computed by Greedy belongs to $B(f)$ as follows:
  - Index the elements such that $\prec$ is $e_1 \prec e_2 \prec \cdots \prec e_n$. First, $x(E) = \sum_{e_i \in E}[f(e_i^{\prec} + e_i) - f(e_i^{\prec})] = f(E) - f(\emptyset) = f(E)$.
  - Now for any $\emptyset \subset S \subset E$ we need to verify that $x(S) \leq f(S)$. Define k as the largest index such that $e_k \in S$, and use induction on k.
  - If $k = 1$ then $S = \{e_1\}$ and $x_1 = f(e_1^{\prec} + e_1) - f(e_1^{\prec}) = f(\{e_1\}) - f(\emptyset) = f(S)$.
  - If $k > 1$, then $S \cup e_k^{\prec} = e_{k+1}^{\prec}$ and $S \cap e_k^{\prec} = S - e_k$. Then submodularity implies that $f(S) \geq f(S \cup e_k^{\prec}) + f(S \cap e_k^{\prec}) - f(e_k^{\prec}) = f(e_{k+1}^{\prec}) + f(S - e_k) - f(e_k^{\prec})$.
  - The largest e_i in $S - e_k$ is smaller than k, so induction applies to $S - e_k$ and we get $x(S) - x_{e_k} = x(S - e_k) \leq f(S - e_k)$, or $x(S) \leq f(S - e_k) + x_{e_k} = f(S - e_k) + (f(e_k^{\prec} + e_k) - f(e_k^{\prec}))$.
  - Thus $x(S) \leq f(S - e_k) + (f(e_k^{\prec} + e_k) - f(e_k^{\prec})) = f(e_{k+1}^{\prec}) + f(S - e_k) - f(e_k^{\prec}) \leq f(S)$.

Is Greedy's solution optimal?

- Recall that we are trying to solve $\max_{x \in \mathbb{R}^E} w^T x$ s.t. $x \in B(f)$.

Is Greedy's solution optimal?

- Recall that we are trying to solve $\max_{x \in \mathbb{R}^E} w^T x$ s.t. $x \in B(f)$.
- This is a linear program (LP):

$$
\begin{array}{rcll}
\max w^T x & & & \\
\text{s.t. } x(S) & \leq & f(S) & \text{for all } \emptyset \subset S \subset E \\
x(E) & = & f(E) & \\
x & & \text{free.} &
\end{array}
$$

Is Greedy's solution optimal?

- Recall that we are trying to solve $\max_{x \in \mathbb{R}^E} w^T x$ s.t. $x \in B(f)$.
- This is a linear program (LP):

$$
\begin{array}{rcll}
\max w^T x & & & \\
\text{s.t. } x(S) & \leq & f(S) & \text{for all } \emptyset \subset S \subset E \\
x(E) & = & f(E) & \\
x & & \text{free.} &
\end{array}
$$

- This LP has 2^n constraints, one for each S.

Is Greedy's solution optimal?

- Recall that we are trying to solve $\max_{x \in \mathbb{R}^E} w^T x$ s.t. $x \in B(f)$.
- This is a linear program (LP):

$$\begin{array}{rcll} \max w^T x & & & \\ \text{s.t. } x(S) & \leq & f(S) & \text{for all } \emptyset \subset S \subset E \\ x(E) & = & f(E) & \\ x & & \text{free.} & \end{array}$$

- This LP has 2^n constraints, one for each S.
- Optimality is proven via duality. Put dual variable π_S on constraint $x(S) \leq f(S)$ to get the dual:

$$\begin{array}{rcll} \min \sum_{S \subseteq E} f(S)\pi_S & & & \\ \text{s.t. } \sum_{S \ni e} \pi_S & = & w_e & \text{for all } e \in E \\ \pi_S & \geq & 0 & \text{for all } S \subset E \\ \pi_E & & \text{free.} & \end{array}$$

Is Greedy's solution optimal?

- Recall that we are trying to solve $\max_{x \in \mathbb{R}^E} w^T x$ s.t. $x \in B(f)$.
- This is a linear program (LP):

$$\begin{array}{rcll} \max w^T x & & & \\ \text{s.t. } x(S) & \leq & f(S) & \text{for all } \emptyset \subset S \subset E \\ x(E) & = & f(E) & \\ x & & \text{free.} & \end{array}$$

- This LP has 2^n constraints, one for each S.
- Optimality is proven via duality. Put dual variable π_S on constraint $x(S) \leq f(S)$ to get the dual:

$$\begin{array}{rcll} \min \sum_{S \subseteq E} f(S)\pi_S & & & \\ \text{s.t. } \sum_{S \ni e} \pi_S & = & w_e & \text{for all } e \in E \\ \pi_S & \geq & 0 & \text{for all } S \subset E \\ \pi_E & & \text{free.} & \end{array}$$

- In order to show optimality of the x coming from Greedy, we construct a dual optimal solution.

Dual feasibility

- Here are the dual LPs:

$$\begin{array}{rcll} \max w^T x & & & \\ \text{s.t. } x(S) & \leq & f(S) & \forall S \\ x(E) & = & f(E) & \\ x & & \text{free.} & \end{array} \qquad \begin{array}{rcll} \min \sum_{S \subseteq E} f(S) \pi_S & & & \\ \text{s.t. } \sum_{S \ni e} \pi_S & = & w_e & \\ \pi_S & \geq & 0 & S \neq E \\ \pi_E & & \text{free.} & \end{array}$$

Dual feasibility

- Here are the dual LPs:

$$\begin{array}{rcll} \max w^T x & & & \\ \text{s.t. } x(S) & \leq & f(S) & \forall S \\ x(E) & = & f(E) & \\ x & & \text{free.} & \end{array} \qquad \begin{array}{rcll} \min \sum_{S \subseteq E} f(S) \pi_S & & & \\ \text{s.t. } \sum_{S \ni e} \pi_S & = & w_e & \\ \pi_S & \geq & 0 & S \neq E \\ \pi_E & & \text{free.} & \end{array}$$

- Define π_S like this: Put $\pi_S = w_{e_{i-1}} - w_{e_i}$ if $S = e_i^{\prec}$, $\pi_E = w_{e_n} - 0$ (using "$w_{e_{n+1}} = 0$"), and $\pi_S = 0$ otherwise.

Dual feasibility

- Here are the dual LPs:

$$\begin{array}{rcll} \max w^T x & & & \\ \text{s.t. } x(S) & \leq & f(S) & \forall S \\ x(E) & = & f(E) & \\ x & & \text{free.} & \end{array} \qquad \begin{array}{rcll} \min \sum_{S \subseteq E} f(S)\pi_S & & & \\ \text{s.t. } \sum_{S \ni e} \pi_S & = & w_e & \\ \pi_S & \geq & 0 & S \neq E \\ \pi_E & & \text{free.} & \end{array}$$

- Define π_S like this: Put $\pi_S = w_{e_{i-1}} - w_{e_i}$ if $S = e_i^{\prec}$, $\pi_E = w_{e_n} - 0$ (using "$w_{e_{n+1}} = 0$"), and $\pi_S = 0$ otherwise.
- First, note that this π_S is feasible for the dual LP:

Dual feasibility

- Here are the dual LPs:

$$\begin{array}{rcll} \max w^T x & & & \\ \text{s.t. } x(S) & \le & f(S) & \forall S \\ x(E) & = & f(E) & \\ x & & \text{free.} & \end{array} \qquad \begin{array}{rcll} \min \sum_{S \subseteq E} f(S)\pi_S & & & \\ \text{s.t. } \sum_{S \ni e} \pi_S & = & w_e & \\ \pi_S & \ge & 0 & S \neq E \\ \pi_E & & \text{free.} & \end{array}$$

- Define π_S like this: Put $\pi_S = w_{e_{i-1}} - w_{e_i}$ if $S = e_i^{\prec}$, $\pi_E = w_{e_n} - 0$ (using "$w_{e_{n+1}} = 0$"), and $\pi_S = 0$ otherwise.
- First, note that this π_S is feasible for the dual LP:
  - We chose $\prec$ s.t. $w_{e_{i-1}} - w_{e_i} \ge 0$, and so $\pi_S \ge 0$.

Dual feasibility

- Here are the dual LPs:

$$\begin{array}{rcll} \max w^T x & & & \\ \text{s.t. } x(S) & \leq & f(S) & \forall S \\ x(E) & = & f(E) & \\ x & & \text{free.} & \end{array} \qquad \begin{array}{rcll} \min \sum_{S \subseteq E} f(S)\pi_S & & & \\ \text{s.t. } \sum_{S \ni e} \pi_S & = & w_e & \\ \pi_S & \geq & 0 & S \neq E \\ \pi_E & & \text{free.} & \end{array}$$

- Define π_S like this: Put $\pi_S = w_{e_{i-1}} - w_{e_i}$ if $S = e_i^{\prec}$, $\pi_E = w_{e_n} - 0$ (using "$w_{e_{n+1}} = 0$"), and $\pi_S = 0$ otherwise.
- First, note that this π_S is feasible for the dual LP:
  - We chose $\prec$ s.t. $w_{e_{i-1}} - w_{e_i} \geq 0$, and so $\pi_S \geq 0$.
  - Now $\sum_{S \ni e_k} \pi_S = \sum_{i=k+1}^{n+1} (w_{e_{i-1}} - w_{e_i})$
    $= w_{e_k} - w_{e_{n+1}} = w_{e_k}$, as desired.

Optimality from duality

- For any $x \in B(f)$ and π feasible for the dual, note that

$$\begin{array}{rcl} w^T x & = & \sum_{e \in E} (\sum_{S \ni e} \pi_S) x_e \\ & = & \sum_{S \subseteq E} \pi_S \sum_{e \in S} x_e \\ & = & \sum_{S \subseteq E} \pi_S x(S) \\ & \leq & \sum_{S \subseteq E} \pi_S f(S). \end{array}$$

Optimality from duality

- For any $x \in B(f)$ and π feasible for the dual, note that

$$\begin{array}{rcl} w^T x & = & \sum_{e \in E} (\sum_{S \ni e} \pi_S) x_e \\ & = & \sum_{S \subseteq E} \pi_S \sum_{e \in S} x_e \\ & = & \sum_{S \subseteq E} \pi_S x(S) \\ & \leq & \sum_{S \subseteq E} \pi_S f(S). \end{array}$$

- Since we already proved that the Greedy output $x \in B(f)$ and our π is feasible, we only need to show that $w^T x = \sum_{S \subseteq E} \pi_S f(S)$.

Optimality from duality

- For any $x \in B(f)$ and π feasible for the dual, note that

$$\begin{aligned} w^T x &= \sum_{e \in E} (\sum_{S \ni e} \pi_S) x_e \\ &= \sum_{S \subseteq E} \pi_S \sum_{e \in S} x_e \\ &= \sum_{S \subseteq E} \pi_S x(S) \\ &\leq \sum_{S \subseteq E} \pi_S f(S). \end{aligned}$$

- Since we already proved that the Greedy output $x \in B(f)$ and our π is feasible, we only need to show that $w^T x = \sum_{S \subseteq E} \pi_S f(S)$.
- Consider the above display. The only place there's an inequality is $\sum_{S \subseteq E} \pi_S x(S) \leq \sum_{S \subseteq E} \pi_S f(S)$.

Optimality from duality

- For any $x \in B(f)$ and π feasible for the dual, note that

$$\begin{array}{rcl} w^T x & = & \sum_{e \in E} (\sum_{S \ni e} \pi_S) x_e \\ & = & \sum_{S \subseteq E} \pi_S \sum_{e \in S} x_e \\ & = & \sum_{S \subseteq E} \pi_S x(S) \\ & \leq & \sum_{S \subseteq E} \pi_S f(S). \end{array}$$

- Since we already proved that the Greedy output $x \in B(f)$ and our π is feasible, we only need to show that $w^T x = \sum_{S \subseteq E} \pi_S f(S)$.
- Consider the above display. The only place there's an inequality is $\sum_{S \subseteq E} \pi_S x(S) \leq \sum_{S \subseteq E} \pi_S f(S)$.
  - If $\pi_S = 0$ then both sides are zero.

Optimality from duality

- For any $x \in B(f)$ and π feasible for the dual, note that

$$\begin{array}{rcl} w^T x & = & \sum_{e \in E}(\sum_{S \ni e} \pi_S) x_e \\ & = & \sum_{S \subseteq E} \pi_S \sum_{e \in S} x_e \\ & = & \sum_{S \subseteq E} \pi_S x(S) \\ & \leq & \sum_{S \subseteq E} \pi_S f(S). \end{array}$$

- Since we already proved that the Greedy output $x \in B(f)$ and our π is feasible, we only need to show that $w^T x = \sum_{S \subseteq E} \pi_S f(S)$.
- Consider the above display. The only place there's an inequality is $\sum_{S \subseteq E} \pi_S x(S) \leq \sum_{S \subseteq E} \pi_S f(S)$.
  - If $\pi_S = 0$ then both sides are zero.
  - If $\pi_S \neq 0$, then S is $e_k^{\prec}$ for some k.

Optimality from duality

- For any $x \in B(f)$ and π feasible for the dual, note that

$$\begin{array}{rcl} w^T x & = & \sum_{e \in E} (\sum_{S \ni e} \pi_S) x_e \\ & = & \sum_{S \subseteq E} \pi_S \sum_{e \in S} x_e \\ & = & \sum_{S \subseteq E} \pi_S x(S) \\ & \leq & \sum_{S \subseteq E} \pi_S f(S). \end{array}$$

- Since we already proved that the Greedy output $x \in B(f)$ and our π is feasible, we only need to show that $w^T x = \sum_{S \subseteq E} \pi_S f(S)$.
- Consider the above display. The only place there's an inequality is $\sum_{S \subseteq E} \pi_S x(S) \leq \sum_{S \subseteq E} \pi_S f(S)$.
  - If $\pi_S = 0$ then both sides are zero.
  - If $\pi_S \neq 0$, then S is $e_k^{\prec}$ for some k.
  - But then $x(S) = \sum_{i<k} x_{e_i} = \sum_{i<k} (f(e_i^{\prec} + e_i) - f(e_i^{\prec})) = f(e_{k-1}^{\prec} + e_{k-1}) - f(\emptyset) = f(e_k^{\prec}) = f(S)$.

Optimality from duality

- For any $x \in B(f)$ and π feasible for the dual, note that

$$\begin{array}{rcl} w^T x & = & \sum_{e \in E} (\sum_{S \ni e} \pi_S) x_e \\ & = & \sum_{S \subseteq E} \pi_S \sum_{e \in S} x_e \\ & = & \sum_{S \subseteq E} \pi_S x(S) \\ & \leq & \sum_{S \subseteq E} \pi_S f(S). \end{array}$$

- Since we already proved that the Greedy output $x \in B(f)$ and our π is feasible, we only need to show that $w^T x = \sum_{S \subseteq E} \pi_S f(S)$.
- Consider the above display. The only place there's an inequality is $\sum_{S \subseteq E} \pi_S x(S) \leq \sum_{S \subseteq E} \pi_S f(S)$.
  - If $\pi_S = 0$ then both sides are zero.
  - If $\pi_S \neq 0$, then S is $e_k^{\prec}$ for some k.
  - But then $x(S) = \sum_{i<k} x_{e_i} = \sum_{i<k} (f(e_i^{\prec} + e_i) - f(e_i^{\prec})) = f(e_{k-1}^{\prec} + e_{k-1}) - f(\emptyset) = f(e_k^{\prec}) = f(S)$.
  - Thus we get equality, and so x is (primal) optimal (and π is dual optimal).

Notes about the Greedy Algorithm

- The Greedy Algorithm takes $O(n\mathrm{EO} + n \log n)$ time:

Notes about the Greedy Algorithm

- The Greedy Algorithm takes $O(n\mathrm{EO} + n \log n)$ time:
  - It takes $O(n \log n)$ time to sort the w_e.

Notes about the Greedy Algorithm

- The Greedy Algorithm takes $O(n\mathrm{EO} + n \log n)$ time:
  - It takes $O(n \log n)$ time to sort the w_e.
  - There are n calls to $\mathcal{E}$ that cost $O(n\mathrm{EO})$.

Notes about the Greedy Algorithm

- The Greedy Algorithm takes $O(n\mathrm{EO} + n \log n)$ time:
  - It takes $O(n \log n)$ time to sort the w_e.
  - There are n calls to $\mathcal{E}$ that cost $O(n\mathrm{EO})$.
- It can be shown (see below) that the output x of Greedy is in fact a vertex of $B(f)$.

Notes about the Greedy Algorithm

- The Greedy Algorithm takes $O(n\mathrm{EO} + n \log n)$ time:
  - It takes $O(n \log n)$ time to sort the w_e.
  - There are n calls to $\mathcal{E}$ that cost $O(n\mathrm{EO})$.
- It can be shown (see below) that the output x of Greedy is in fact a vertex of $B(f)$.
  - When the input to Greedy is linear order $\prec$, we denote the output x by $v^{\prec}$.

Notes about the Greedy Algorithm

- The Greedy Algorithm takes $O(n\mathrm{EO} + n \log n)$ time:
  - It takes $O(n \log n)$ time to sort the w_e.
  - There are n calls to $\mathcal{E}$ that cost $O(n\mathrm{EO})$.
- It can be shown (see below) that the output x of Greedy is in fact a vertex of $B(f)$.
  - When the input to Greedy is linear order $\prec$, we denote the output x by $v^{\prec}$.
  - We have shown that $w^T x$ is maximized at $v^{\prec}$ for an order $\prec$ consistent with w, and so in fact these Greedy vertices are *all* the vertices of $B(f)$. Thus there are at most $n!$ vertices of $B(f)$.

Notes about the Greedy Algorithm

- The Greedy Algorithm takes $O(n\mathrm{EO} + n \log n)$ time:
  - It takes $O(n \log n)$ time to sort the w_e.
  - There are n calls to $\mathcal{E}$ that cost $O(n\mathrm{EO})$.
- It can be shown (see below) that the output x of Greedy is in fact a vertex of $B(f)$.
  - When the input to Greedy is linear order $\prec$, we denote the output x by $v^{\prec}$.
  - We have shown that $w^T x$ is maximized at $v^{\prec}$ for an order $\prec$ consistent with w, and so in fact these Greedy vertices are *all* the vertices of $B(f)$. Thus there are at most $n!$ vertices of $B(f)$.
  - Although $B(f)$ has 2^n constraints, the linear order $\prec$ is a succinct certificate that $v^{\prec} \in B(f)$.

Notes about the Greedy Algorithm

- The Greedy Algorithm takes $O(n\mathrm{EO} + n \log n)$ time:
  - It takes $O(n \log n)$ time to sort the w_e.
  - There are n calls to $\mathcal{E}$ that cost $O(n\mathrm{EO})$.
- It can be shown (see below) that the output x of Greedy is in fact a vertex of $B(f)$.
  - When the input to Greedy is linear order $\prec$, we denote the output x by $v^{\prec}$.
  - We have shown that $w^T x$ is maximized at $v^{\prec}$ for an order $\prec$ consistent with w, and so in fact these Greedy vertices are *all* the vertices of $B(f)$. Thus there are at most $n!$ vertices of $B(f)$.
  - Although $B(f)$ has 2^n constraints, the linear order $\prec$ is a succinct certificate that $v^{\prec} \in B(f)$.
  - This proves that $B(f) \neq \emptyset$.

Notes about the Greedy Algorithm

- The Greedy Algorithm takes $O(n\mathrm{EO} + n \log n)$ time:
  - It takes $O(n \log n)$ time to sort the w_e.
  - There are n calls to $\mathcal{E}$ that cost $O(n\mathrm{EO})$.
- It can be shown (see below) that the output x of Greedy is in fact a vertex of $B(f)$.
  - When the input to Greedy is linear order $\prec$, we denote the output x by $v^{\prec}$.
  - We have shown that $w^T x$ is maximized at $v^{\prec}$ for an order $\prec$ consistent with w, and so in fact these Greedy vertices are *all* the vertices of $B(f)$. Thus there are at most $n!$ vertices of $B(f)$.
  - Although $B(f)$ has 2^n constraints, the linear order $\prec$ is a succinct certificate that $v^{\prec} \in B(f)$.
  - This proves that $B(f) \neq \emptyset$.
  - Greedy works on $B(f)$ for *any* w; it works on $P(f)$ if $w \geq 0$.

Understanding the basis matrix for Greedy

- The basis matrix M for an LP is the submatrix induced by the columns of the variables not at their bounds, and the rows whose constraints are tight (satisfied with equality).

Understanding the basis matrix for Greedy

- The basis matrix M for an LP is the submatrix induced by the columns of the variables not at their bounds, and the rows whose constraints are tight (satisfied with equality).
  - Here all the x_e are free (do not have bounds) and so M includes columns for every $e \in E$.

Understanding the basis matrix for Greedy

- The basis matrix M for an LP is the submatrix induced by the columns of the variables not at their bounds, and the rows whose constraints are tight (satisfied with equality).
  - Here all the x_e are free (do not have bounds) and so M includes columns for every $e \in E$.
  - As we saw in the proof, the constraint for $S = e_k^{\prec}$ is tight for each $e_k \in E$.

Understanding the basis matrix for Greedy

- The basis matrix M for an LP is the submatrix induced by the columns of the variables not at their bounds, and the rows whose constraints are tight (satisfied with equality).
  - Here all the x_e are free (do not have bounds) and so M includes columns for every $e \in E$.
  - As we saw in the proof, the constraint for $S = e_k^{\prec}$ is tight for each $e_k \in E$.
- Therefore M is the lower triangular matrix:

$$M = \begin{array}{c} \\ e_2^{\prec} \\ e_3^{\prec} \\ \vdots \\ e_{n+1}^{\prec} \end{array} \begin{array}{c} \begin{array}{cccc} e_1 & e_2 & \dots & e_n \end{array} \\ \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix} \end{array}$$

More Greedy basis matrix

- Recall that M is the lower triangular matrix:

$$M = \begin{array}{c c} & \begin{array}{cccc} e_1 & e_2 & \dots & e_n \end{array} \\ \begin{array}{c} e_2^{\prec} \\ e_3^{\prec} \\ \vdots \\ e_{n+1}^{\prec} \end{array} & \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix} \end{array}$$

More Greedy basis matrix

- Recall that M is the lower triangular matrix:

$$M = \begin{array}{c c} & \begin{array}{cccc} e_1 & e_2 & \dots & e_n \end{array} \\ \begin{array}{c} e_2^{\prec} \\ e_3^{\prec} \\ \vdots \\ e_{n+1}^{\prec} \end{array} & \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix} \end{array}$$

- Let $b^{\prec}$ be the RHS $(f(e_2^{\prec}), f(e_3^{\prec}), \dots, f(e_{n+1}^{\prec}))$.

More Greedy basis matrix

- Recall that M is the lower triangular matrix:

$$M = \begin{array}{c c} & \begin{array}{cccc} e_1 & e_2 & \dots & e_n \end{array} \\ \begin{array}{c} e_2^{\prec} \\ e_3^{\prec} \\ \vdots \\ e_{n+1}^{\prec} \end{array} & \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix} \end{array}$$

- Let $b^{\prec}$ be the RHS $(f(e_2^{\prec}), f(e_3^{\prec}), \dots, f(e_{n+1}^{\prec}))$.
- Then our Greedy primal vector $v^{\prec}$ solves $Mv^{\prec} = b^{\prec}$.

More Greedy basis matrix

- Recall that M is the lower triangular matrix:

$$M = \begin{array}{c} \\ e_2^{\prec} \\ e_3^{\prec} \\ \vdots \\ e_{n+1}^{\prec} \end{array} \begin{array}{c} \begin{array}{cccc} e_1 & e_2 & \dots & e_n \end{array} \\ \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix} \end{array}$$

- Let $b^{\prec}$ be the RHS $(f(e_2^{\prec}), f(e_3^{\prec}), \dots, f(e_{n+1}^{\prec}))$.
- Then our Greedy primal vector $v^{\prec}$ solves $Mv^{\prec} = b^{\prec}$.
- Triangular systems like this are easy to solve, and indeed gives that $x_{e_i} = f(e_i^{\prec} + e_i) - f(e_i^{\prec})$.

More Greedy basis matrix

- Recall that M is the lower triangular matrix:

$$M = \begin{array}{c} \\ e_2^{\prec} \\ e_3^{\prec} \\ \vdots \\ e_{n+1}^{\prec} \end{array} \begin{array}{c} \begin{array}{cccc} e_1 & e_2 & \dots & e_n \end{array} \\ \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix} \end{array}$$

- Let $b^{\prec}$ be the RHS $(f(e_2^{\prec}), f(e_3^{\prec}), \dots, f(e_{n+1}^{\prec}))$.
- Then our Greedy primal vector $v^{\prec}$ solves $Mv^{\prec} = b^{\prec}$.
- Triangular systems like this are easy to solve, and indeed gives that $x_{e_i} = f(e_i^{\prec} + e_i) - f(e_i^{\prec})$.
- Duality says that the dual has the same basis matrix, and π restricted to the $e_i^{\prec}$ solves $\pi^T M = w^T$.

More Greedy basis matrix

- Recall that M is the lower triangular matrix:

$$M = \begin{array}{c} \\ e_2^{\prec} \\ e_3^{\prec} \\ \vdots \\ e_{n+1}^{\prec} \end{array} \begin{array}{c} \begin{array}{cccc} e_1 & e_2 & \dots & e_n \end{array} \\ \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix} \end{array}$$

- Let $b^{\prec}$ be the RHS $(f(e_2^{\prec}), f(e_3^{\prec}), \dots, f(e_{n+1}^{\prec}))$.
- Then our Greedy primal vector $v^{\prec}$ solves $Mv^{\prec} = b^{\prec}$.
- Triangular systems like this are easy to solve, and indeed gives that $x_{e_i} = f(e_i^{\prec} + e_i) - f(e_i^{\prec})$.
- Duality says that the dual has the same basis matrix, and π restricted to the $e_i^{\prec}$ solves $\pi^T M = w^T$.
- Again this triangular system easily solves to $\pi_{e_i^{\prec}} = w_{i-1} - w_i$.

More Greedy basis matrix

- Recall that M is the lower triangular matrix:

$$M = \begin{array}{c c} & \begin{array}{cccc} e_1 & e_2 & \dots & e_n \end{array} \\ \begin{array}{c} e_2^{\prec} \\ e_3^{\prec} \\ \vdots \\ e_{n+1}^{\prec} \end{array} & \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix} \end{array}$$

- Let $b^{\prec}$ be the RHS $(f(e_2^{\prec}), f(e_3^{\prec}), \dots, f(e_{n+1}^{\prec}))$.
- Then our Greedy primal vector $v^{\prec}$ solves $Mv^{\prec} = b^{\prec}$.
- Triangular systems like this are easy to solve, and indeed gives that $x_{e_i} = f(e_i^{\prec} + e_i) - f(e_i^{\prec})$.
- Duality says that the dual has the same basis matrix, and π restricted to the $e_i^{\prec}$ solves $\pi^T M = w^T$.
- Again this triangular system easily solves to $\pi_{e_i^{\prec}} = w_{i-1} - w_i$.
- This also shows that $v^{\prec}$ is a vertex, as it follows from M being nonsingular.

Outline

From vertices of $B(f)$ to edges of $B(f)$

- We now understand the vertices of $B(f)$ via Greedy.

From vertices of $B(f)$ to edges of $B(f)$

- We now understand the vertices of $B(f)$ via Greedy.
- To be able to move around in $B(f)$ we also need to understand its edges.

From vertices of $B(f)$ to edges of $B(f)$

- We now understand the vertices of $B(f)$ via Greedy.
- To be able to move around in $B(f)$ we also need to understand its edges.
- Suppose that $\prec$ looks like

$$e_1 e_2 \ldots e_i l k e_{i+3} \ldots e_n,$$

and $\prec'$ looks like

$$e_1 e_2 \ldots e_i k l e_{i+3} \ldots e_n;$$

we say that (l, k) are *consecutive* in $\prec$.

From vertices of $B(f)$ to edges of $B(f)$

- We now understand the vertices of $B(f)$ via Greedy.
- To be able to move around in $B(f)$ we also need to understand its edges.
- Suppose that $\prec$ looks like

$$e_1 e_2 \dots e_i l k e_{i+3} \dots e_n,$$

  and $\prec'$ looks like

$$e_1 e_2 \dots e_i k l e_{i+3} \dots e_n;$$

  we say that (l, k) are *consecutive* in $\prec$.
- For $e \in E$ define $\chi(e) \in \{0,1\}^E$ by $\chi(e)_e = 1$ and $\chi(e)_g = 0$ for $g \neq e$.

From vertices of $B(f)$ to edges of $B(f)$

- We now understand the vertices of $B(f)$ via Greedy.
- To be able to move around in $B(f)$ we also need to understand its edges.
- Suppose that $\prec$ looks like

$$e_1 e_2 \dots e_i l k e_{i+3} \dots e_n,$$

  and $\prec'$ looks like

$$e_1 e_2 \dots e_i k l e_{i+3} \dots e_n;$$

  we say that (l, k) are *consecutive* in $\prec$.
- For $e \in E$ define $\chi(e) \in \{0,1\}^E$ by $\chi(e)_e = 1$ and $\chi(e)_g = 0$ for $g \neq e$.
- We are going to show that $v^{\prec'} - v^{\prec} = \alpha(\chi_k - \chi_l)$ for a step length α.

Stepping along an edge

- Recall that $v^{\prec}$ comes from $e_1 e_2 \dots e_i {\color{red}l}{\color{blue}k} e_{i+3} \dots e_n$, and $\prec'$ comes from $e_1 e_2 \dots e_i {\color{blue}k}{\color{red}l} e_{i+3} \dots e_n$.

Stepping along an edge

- Recall that $v^{\prec}$ comes from $e_1 e_2 \ldots e_i \textcolor{red}{l}\textcolor{blue}{k} e_{i+3} \ldots e_n$, and $\prec'$ comes from $e_1 e_2 \ldots e_i \textcolor{blue}{k}\textcolor{red}{l} e_{i+3} \ldots e_n$.
- Notice that for $e \neq k, l$ we have that $e^{\prec} = e^{\prec'}$.

Stepping along an edge

- Recall that $v^{\prec}$ comes from $e_1 e_2 \dots e_i l k e_{i+3} \dots e_n$, and $\prec'$ comes from $e_1 e_2 \dots e_i k l e_{i+3} \dots e_n$.
- Notice that for $e \neq k, l$ we have that $e^{\prec} = e^{\prec'}$.
  - Thus for $e \neq k, l$ we have that $v_e^{\prec} = f(e^{\prec} + e) - f(e^{\prec}) = f(e^{\prec'} + e) - f(e^{\prec'}) = v_e^{\prec'}$.

Stepping along an edge

- Recall that $v^{\prec}$ comes from $e_1 e_2 \dots e_i l k e_{i+3} \dots e_n$, and $\prec'$ comes from $e_1 e_2 \dots e_i k l e_{i+3} \dots e_n$.
- Notice that for $e \neq k, l$ we have that $e^{\prec} = e^{\prec'}$.
  - Thus for $e \neq k, l$ we have that $v_e^{\prec} = f(e^{\prec} + e) - f(e^{\prec}) = f(e^{\prec'} + e) - f(e^{\prec'}) = v_e^{\prec'}$.
- For $e = k$ we have $v_k^{\prec} = f(k^{\prec} + k) - f(k^{\prec}) = f(l^{\prec} + k + l) - f(l^{\prec} + l)$ and $v_k^{\prec'} = f(k^{\prec'} + k) - f(k^{\prec'}) = f(l^{\prec} + k) - f(l^{\prec})$.

Stepping along an edge

- Recall that $v^{\prec}$ comes from $e_1 e_2 \dots e_i l k e_{i+3} \dots e_n$, and $\prec'$ comes from $e_1 e_2 \dots e_i k l e_{i+3} \dots e_n$.
- Notice that for $e \neq k, l$ we have that $e^{\prec} = e^{\prec'}$.
  - Thus for $e \neq k, l$ we have that $v_e^{\prec} = f(e^{\prec} + e) - f(e^{\prec}) = f(e^{\prec'} + e) - f(e^{\prec'}) = v_e^{\prec'}$.
- For $e = k$ we have $v_k^{\prec} = f(k^{\prec} + k) - f(k^{\prec}) = f(l^{\prec} + k + l) - f(l^{\prec} + l)$ and $v_k^{\prec'} = f(k^{\prec'} + k) - f(k^{\prec'}) = f(l^{\prec} + k) - f(l^{\prec})$.
- For $e = l$ we have $v_l^{\prec} = f(l^{\prec} + l) - f(l^{\prec}) = f(l^{\prec} + l) - f(l^{\prec})$ and $v_l^{\prec'} = f(l^{\prec'} + l) - f(l^{\prec'}) = f(l^{\prec} + k + l) - f(l^{\prec} + k)$.

Stepping along an edge

- Recall that $v^{\prec}$ comes from $e_1 e_2 \dots e_i l k e_{i+3} \dots e_n$, and $\prec'$ comes from $e_1 e_2 \dots e_i k l e_{i+3} \dots e_n$.
- Notice that for $e \neq k, l$ we have that $e^{\prec} = e^{\prec'}$.
  - Thus for $e \neq k, l$ we have that $v_e^{\prec} = f(e^{\prec} + e) - f(e^{\prec}) = f(e^{\prec'} + e) - f(e^{\prec'}) = v_e^{\prec'}$.
- For $e = k$ we have $v_k^{\prec} = f(k^{\prec} + k) - f(k^{\prec}) = f(l^{\prec} + k + l) - f(l^{\prec} + l)$ and $v_k^{\prec'} = f(k^{\prec'} + k) - f(k^{\prec'}) = f(l^{\prec} + k) - f(l^{\prec})$.
- For $e = l$ we have $v_l^{\prec} = f(l^{\prec} + l) - f(l^{\prec}) = f(l^{\prec} + l) - f(l^{\prec})$ and $v_l^{\prec'} = f(l^{\prec'} + l) - f(l^{\prec'}) = f(l^{\prec} + k + l) - f(l^{\prec} + k)$.
  - Define $\alpha = [f(l^{\prec} + l) - f(l^{\prec})] - [f(l^{\prec} + k + l) - f(l^{\prec} + k)]$.

Stepping along an edge

- Recall that $v^{\prec}$ comes from $e_1e_2\dots e_i lk e_{i+3}\dots e_n$, and $\prec'$ comes from $e_1e_2\dots e_i kl e_{i+3}\dots e_n$.
- Notice that for $e \neq k, l$ we have that $e^{\prec} = e^{\prec'}$.
  - Thus for $e \neq k, l$ we have that $v_e^{\prec} = f(e^{\prec} + e) - f(e^{\prec}) = f(e^{\prec'} + e) - f(e^{\prec'}) = v_e^{\prec'}$.
- For $e = k$ we have $v_k^{\prec} = f(k^{\prec} + k) - f(k^{\prec}) = f(l^{\prec} + k + l) - f(l^{\prec} + l)$ and $v_k^{\prec'} = f(k^{\prec'} + k) - f(k^{\prec'}) = f(l^{\prec} + k) - f(l^{\prec})$.
- For $e = l$ we have $v_l^{\prec} = f(l^{\prec} + l) - f(l^{\prec}) = f(l^{\prec} + l) - f(l^{\prec})$ and $v_l^{\prec'} = f(l^{\prec'} + l) - f(l^{\prec'}) = f(l^{\prec} + k + l) - f(l^{\prec} + k)$.
  - Define $\alpha = [f(l^{\prec} + l) - f(l^{\prec})] - [f(l^{\prec} + k + l) - f(l^{\prec} + k)]$.
  - By submodularity, $\alpha \geq 0$.

Stepping along an edge

- Recall that $v^{\prec}$ comes from $e_1 e_2 \dots e_i l k e_{i+3} \dots e_n$, and $\prec'$ comes from $e_1 e_2 \dots e_i k l e_{i+3} \dots e_n$.
- Notice that for $e \neq k, l$ we have that $e^{\prec} = e^{\prec'}$.
  - Thus for $e \neq k, l$ we have that $v_e^{\prec} = f(e^{\prec} + e) - f(e^{\prec}) = f(e^{\prec'} + e) - f(e^{\prec'}) = v_e^{\prec'}$.
- For $e = k$ we have $v_k^{\prec} = f(k^{\prec} + k) - f(k^{\prec}) = f(l^{\prec} + k + l) - f(l^{\prec} + l)$ and $v_k^{\prec'} = f(k^{\prec'} + k) - f(k^{\prec'}) = f(l^{\prec} + k) - f(l^{\prec})$.
- For $e = l$ we have $v_l^{\prec} = f(l^{\prec} + l) - f(l^{\prec}) = f(l^{\prec} + l) - f(l^{\prec})$ and $v_l^{\prec'} = f(l^{\prec'} + l) - f(l^{\prec'}) = f(l^{\prec} + k + l) - f(l^{\prec} + k)$.
  - Define $\alpha = [f(l^{\prec} + l) - f(l^{\prec})] - [f(l^{\prec} + k + l) - f(l^{\prec} + k)]$.
  - By submodularity, $\alpha \geq 0$.
  - Then we see that $v_l^{\prec'} = v_l^{\prec} - \alpha$, and $v_k^{\prec'} = v_k^{\prec} + \alpha$.

Stepping along an edge

- Recall that $v^{\prec}$ comes from $e_1 e_2 \ldots e_i l k e_{i+3} \ldots e_n$, and $\prec'$ comes from $e_1 e_2 \ldots e_i k l e_{i+3} \ldots e_n$.
- Notice that for $e \neq k, l$ we have that $e^{\prec} = e^{\prec'}$.
  - Thus for $e \neq k, l$ we have that $v_e^{\prec} = f(e^{\prec} + e) - f(e^{\prec}) = f(e^{\prec'} + e) - f(e^{\prec'}) = v_e^{\prec'}$.
- For $e = k$ we have $v_k^{\prec} = f(k^{\prec} + k) - f(k^{\prec}) = f(l^{\prec} + k + l) - f(l^{\prec} + l)$ and $v_k^{\prec'} = f(k^{\prec'} + k) - f(k^{\prec'}) = f(l^{\prec} + k) - f(l^{\prec})$.
- For $e = l$ we have $v_l^{\prec} = f(l^{\prec} + l) - f(l^{\prec}) = f(l^{\prec} + l) - f(l^{\prec})$ and $v_l^{\prec'} = f(l^{\prec'} + l) - f(l^{\prec'}) = f(l^{\prec} + k + l) - f(l^{\prec} + k)$.
  - Define $\alpha = [f(l^{\prec} + l) - f(l^{\prec})] - [f(l^{\prec} + k + l) - f(l^{\prec} + k)]$.
  - By submodularity, $\alpha \geq 0$.
  - Then we see that $v_l^{\prec'} = v_l^{\prec} - \alpha$, and $v_k^{\prec'} = v_k^{\prec} + \alpha$.
  - Intuition: as we move k earlier in $\prec$, $v_k^{\prec}$ gets bigger; as we move k later in $\prec$, $v_k^{\prec}$ gets smaller.

Exchange capacities

- We call this step length $\alpha = [f(l^{\prec} + l) - f(l^{\prec})] - [f(l^{\prec} + k + l) - f(l^{\prec} + k)]$ the exchange capacity of the consecutive pair (l, k), and denote it as $c(k, l; v^{\prec})$.

Exchange capacities

- We call this step length $\alpha = [f(l^{\prec} + l) - f(l^{\prec})] - [f(l^{\prec} + k + l) - f(l^{\prec} + k)]$ the exchange capacity of the consecutive pair (l, k), and denote it as $c(k, l; v^{\prec})$.
  - Since $x(E) = f(E)$ is a constraint of $B(f)$, all $x \in B(f)$ have the constant sum $f(E)$. Thus it is not a surprise that $|v_k^{\prec} - v_k^{\prec'}| = |v_l^{\prec} - v_l^{\prec'}| = c(k, l; v^{\prec})$.

Exchange capacities

- We call this step length $\alpha = [f(l^{\prec} + l) - f(l^{\prec})] - [f(l^{\prec} + k + l) - f(l^{\prec} + k)]$ the exchange capacity of the consecutive pair (l, k), and denote it as $c(k, l; v^{\prec})$.
  - Since $x(E) = f(E)$ is a constraint of $B(f)$, all $x \in B(f)$ have the constant sum $f(E)$. Thus it is not a surprise that $|v_k^{\prec} - v_k^{\prec'}| = |v_l^{\prec} - v_l^{\prec'}| = c(k, l; v^{\prec})$.
  - We have indeed shown that when (l, k) is consecutive in $\prec$, then $v^{\prec'} - v^{\prec} = c(k, l; v^{\prec})(\chi_k - \chi_l)$.

Exchange capacities

- We call this step length $\alpha = [f(l^{\prec} + l) - f(l^{\prec})] - [f(l^{\prec} + k + l) - f(l^{\prec} + k)]$ the exchange capacity of the consecutive pair (l, k), and denote it as $c(k, l; v^{\prec})$.
  - Since $x(E) = f(E)$ is a constraint of $B(f)$, all $x \in B(f)$ have the constant sum $f(E)$. Thus it is not a surprise that $|v_k^{\prec} - v_k^{\prec'}| = |v_l^{\prec} - v_l^{\prec'}| = c(k, l; v^{\prec})$.
  - We have indeed shown that when (l, k) is consecutive in $\prec$, then $v^{\prec'} - v^{\prec} = c(k, l; v^{\prec})(\chi_k - \chi_l)$.
- It turns out that all the edges of $B(f)$ come from consecutive exchanges like this.

Exchange capacities

- We call this step length $\alpha = [f(l^{\prec} + l) - f(l^{\prec})] - [f(l^{\prec} + k + l) - f(l^{\prec} + k)]$ the exchange capacity of the consecutive pair (l, k), and denote it as $c(k, l; v^{\prec})$.
    - Since $x(E) = f(E)$ is a constraint of $B(f)$, all $x \in B(f)$ have the constant sum $f(E)$. Thus it is not a surprise that $|v_k^{\prec} - v_k^{\prec'}| = |v_l^{\prec} - v_l^{\prec'}| = c(k, l; v^{\prec})$.
    - We have indeed shown that when (l, k) is consecutive in $\prec$, then $v^{\prec'} - v^{\prec} = c(k, l; v^{\prec})(\chi_k - \chi_l)$.
- It turns out that all the edges of $B(f)$ come from consecutive exchanges like this.
- Given some $x \in B(f)$ and $k, l \in E$, it is natural to wonder if we can compute the more general exchange capacity $c(k, l; x)$, which is the largest α such that $x + \alpha(\chi_k - \chi_l) \in B(f)$.

Exchange capacities

- We call this step length $\alpha = [f(l^{\prec} + l) - f(l^{\prec})] - [f(l^{\prec} + k + l) - f(l^{\prec} + k)]$ the exchange capacity of the consecutive pair (l, k), and denote it as $c(k, l; v^{\prec})$.
  - Since $x(E) = f(E)$ is a constraint of $B(f)$, all $x \in B(f)$ have the constant sum $f(E)$. Thus it is not a surprise that $|v_k^{\prec} - v_k^{\prec'}| = |v_l^{\prec} - v_l^{\prec'}| = c(k, l; v^{\prec})$.
  - We have indeed shown that when (l, k) is consecutive in $\prec$, then $v^{\prec'} - v^{\prec} = c(k, l; v^{\prec})(\chi_k - \chi_l)$.
- It turns out that all the edges of $B(f)$ come from consecutive exchanges like this.
- Given some $x \in B(f)$ and $k, l \in E$, it is natural to wonder if we can compute the more general exchange capacity $c(k, l; x)$, which is the largest α such that $x + \alpha(\chi_k - \chi_l) \in B(f)$.
  - Unfortunately it turns out that computing $c(k, l; x)$ is provably as difficult as SFMin.

Outline

An algorithmic framework for SFMin

- We now start to develop a framework for algorithms for SFMin (due to Cunningham) that resembles the Max Flow / Min Cut algorithms.

An algorithmic framework for SFMin

- We now start to develop a framework for algorithms for SFMin (due to Cunningham) that resembles the Max Flow / Min Cut algorithms.
- The framework starts by showing that SFMin can be modeled using a dual pair of linear program (due to Edmonds).

An algorithmic framework for SFMin

- We now start to develop a framework for algorithms for SFMin (due to Cunningham) that resembles the Max Flow / Min Cut algorithms.
- The framework starts by showing that SFMin can be modeled using a dual pair of linear program (due to Edmonds).
- However, the first weakly and strongly polynomial algorithms for SFMin came from a very different viewpoint.

An algorithmic framework for SFMin

- We now start to develop a framework for algorithms for SFMin (due to Cunningham) that resembles the Max Flow / Min Cut algorithms.
- The framework starts by showing that SFMin can be modeled using a dual pair of linear program (due to Edmonds).
- However, the first weakly and strongly polynomial algorithms for SFMin came from a very different viewpoint.
  - There is an equivalence between Separation and Optimization via the Ellipsoid Algorithm due to Grötschel, Lovász, and Schrijver.

An algorithmic framework for SFMin

- We now start to develop a framework for algorithms for SFMin (due to Cunningham) that resembles the Max Flow / Min Cut algorithms.
- The framework starts by showing that SFMin can be modeled using a dual pair of linear program (due to Edmonds).
- However, the first weakly and strongly polynomial algorithms for SFMin came from a very different viewpoint.
  - There is an equivalence between Separation and Optimization via the Ellipsoid Algorithm due to Grötschel, Lovász, and Schrijver.
  - For a certain polymatroid, its Separation problem is equivalent to SFMin.

An algorithmic framework for SFMin

- We now start to develop a framework for algorithms for SFMin (due to Cunningham) that resembles the Max Flow / Min Cut algorithms.
- The framework starts by showing that SFMin can be modeled using a dual pair of linear program (due to Edmonds).
- However, the first weakly and strongly polynomial algorithms for SFMin came from a very different viewpoint.
  - There is an equivalence between Separation and Optimization via the Ellipsoid Algorithm due to Grötschel, Lovász, and Schrijver.
  - For a certain polymatroid, its Separation problem is equivalent to SFMin.
  - The polymatroid's Optimization problem is equivalent to the LP we solved via Greedy.

An algorithmic framework for SFMin

- We now start to develop a framework for algorithms for SFMin (due to Cunningham) that resembles the Max Flow / Min Cut algorithms.
- The framework starts by showing that SFMin can be modeled using a dual pair of linear program (due to Edmonds).
- However, the first weakly and strongly polynomial algorithms for SFMin came from a very different viewpoint.
  - There is an equivalence between Separation and Optimization via the Ellipsoid Algorithm due to Grötschel, Lovász, and Schrijver.
  - For a certain polymatroid, its Separation problem is equivalent to SFMin.
  - The polymatroid's Optimization problem is equivalent to the LP we solved via Greedy.
  - Therefore Ellipsoid says that SFMin is (weakly) polynomial.

An algorithmic framework for SFMin

- We now start to develop a framework for algorithms for SFMin (due to Cunningham) that resembles the Max Flow / Min Cut algorithms.
- The framework starts by showing that SFMin can be modeled using a dual pair of linear program (due to Edmonds).
- However, the first weakly and strongly polynomial algorithms for SFMin came from a very different viewpoint.
  - There is an equivalence between Separation and Optimization via the Ellipsoid Algorithm due to Grötschel, Lovász, and Schrijver.
  - For a certain polymatroid, its Separation problem is equivalent to SFMin.
  - The polymatroid's Optimization problem is equivalent to the LP we solved via Greedy.
  - Therefore Ellipsoid says that SFMin is (weakly) polynomial.
  - GLS then extend this to show a strongly polynomial running time.

Edmonds' LP formulation of SFMin

- Recall that SFMin is $\min_{S \subseteq E} f(S)$. It is very unclear whether this can be formulated as an LP.

Edmonds' LP formulation of SFMin

- Recall that SFMin is $\min_{S \subseteq E} f(S)$. It is very unclear whether this can be formulated as an LP.
- Let's modify the dual LPs we used for Greedy by relaxing $x(E) = f(E)$ to just $x(E) \leq f(E)$, putting an upper bound u on x in the primal, and replacing w by the all-ones vector $\mathbb{1}$:

$$\begin{array}{rcl} \max \mathbb{1}^T x & & \\ \text{s.t. } x(S) & \leq & f(S) \\ x & \leq & u \\ x & & \text{free.} \end{array} \qquad \begin{array}{rcl} \min u^T \sigma + \sum_{S \subseteq E} f(S) \pi_S & & \\ \text{s.t. } \sigma_e + \sum_{S \ni e} \pi_S & = & 1 \\ \sigma, \pi & \geq & 0 \end{array}$$

Edmonds' LP formulation of SFMin

- Recall that SFMin is $\min_{S \subseteq E} f(S)$. It is very unclear whether this can be formulated as an LP.
- Let's modify the dual LPs we used for Greedy by relaxing $x(E) = f(E)$ to just $x(E) \leq f(E)$, putting an upper bound u on x in the primal, and replacing w by the all-ones vector $\mathbb{1}$:

$$
\begin{array}{rcl}
\max \mathbb{1}^T x & & \\
\text{s.t. } x(S) & \leq & f(S) \\
x & \leq & u \\
x & & \text{free.}
\end{array}
\qquad
\begin{array}{rcl}
\min u^T \sigma + \sum_{S \subseteq E} f(S) \pi_S & & \\
\text{s.t. } \sigma_e + \sum_{S \ni e} \pi_S & = & 1 \\
\sigma, \pi & \geq & 0
\end{array}
$$

- These kinds of "combinatorial" LPs often have 0–1 optimal solutions.

Edmonds' LP formulation of SFMin

- Recall that SFMin is $\min_{S \subseteq E} f(S)$. It is very unclear whether this can be formulated as an LP.
- Let's modify the dual LPs we used for Greedy by relaxing $x(E) = f(E)$ to just $x(E) \leq f(E)$, putting an upper bound u on x in the primal, and replacing w by the all-ones vector $\mathbb{1}$:

$$\begin{array}{rcl} \max \mathbb{1}^T x & & \\ \text{s.t. } x(S) & \leq & f(S) \\ x & \leq & u \\ x & & \text{free.} \end{array} \qquad \begin{array}{rcl} \min u^T \sigma + \sum_{S \subseteq E} f(S) \pi_S & & \\ \text{s.t. } \sigma_e + \sum_{S \ni e} \pi_S & = & 1 \\ \sigma, \pi & \geq & 0 \end{array}$$

- These kinds of "combinatorial" LPs often have 0–1 optimal solutions.
- Even better, we guess (see below) that there exists an optimal solution to the dual where only one π_S is positive, say $\pi_{S^*} = 1$.

LP optimal structure

- We believe that there exists an optimal solution to the dual where only one π_S is positive, say $\pi_{S^*} = 1$.

LP optimal structure

- We believe that there exists an optimal solution to the dual where only one π_S is positive, say $\pi_{S^*} = 1$.
- Then the constraints $\sigma_e + \sum_{S \ni e} \pi_S = 1$ would force that $\sigma = \chi(E - S^*)$, and so the dual objective value would be $u(E - S^*) + f(S^*)$. Let's prove this.

LP optimal structure

- We believe that there exists an optimal solution to the dual where only one π_S is positive, say $\pi_{S^*} = 1$.
- Then the constraints $\sigma_e + \sum_{S \ni e} \pi_S = 1$ would force that $\sigma = \chi(E - S^*)$, and so the dual objective value would be $u(E - S^*) + f(S^*)$. Let's prove this.
  - (Weak duality:) $\mathbb{1}^T x = x(E) = x(S) + x(E - S) \leq f(S) + u(E - S)$. Thus we only have to show that this is satisfied with equality.

LP optimal structure

- We believe that there exists an optimal solution to the dual where only one π_S is positive, say $\pi_{S^*} = 1$.
- Then the constraints $\sigma_e + \sum_{S \ni e} \pi_S = 1$ would force that $\sigma = \chi(E - S^*)$, and so the dual objective value would be $u(E - S^*) + f(S^*)$. Let's prove this.
  - (Weak duality:) $\mathbb{1}^T x = x(E) = x(S) + x(E - S) \leq f(S) + u(E - S)$. Thus we only have to show that this is satisfied with equality.
  - Suppose that x^* is primal optimal, and S and T are both x^*-tight, i.e., $x^*(S) = f(S)$ and $x^*(T) = f(T)$. Then (homework) both $S \cap T$ and $S \cup T$ are also x^*-tight.

LP optimal structure

- We believe that there exists an optimal solution to the dual where only one π_S is positive, say $\pi_{S^*} = 1$.
- Then the constraints $\sigma_e + \sum_{S \ni e} \pi_S = 1$ would force that $\sigma = \chi(E - S^*)$, and so the dual objective value would be $u(E - S^*) + f(S^*)$. Let's prove this.
  - (Weak duality:) $\mathbb{1}^T x = x(E) = x(S) + x(E - S) \leq f(S) + u(E - S)$. Thus we only have to show that this is satisfied with equality.
  - Suppose that x^* is primal optimal, and S and T are both x^*-tight, i.e., $x^*(S) = f(S)$ and $x^*(T) = f(T)$. Then (homework) both $S \cap T$ and $S \cup T$ are also x^*-tight.
  - Thus we can take the union of all x^*-tight sets to get S^*, which is also x^*-tight.

LP optimal structure

- We believe that there exists an optimal solution to the dual where only one π_S is positive, say $\pi_{S^*} = 1$.
- Then the constraints $\sigma_e + \sum_{S \ni e} \pi_S = 1$ would force that $\sigma = \chi(E - S^*)$, and so the dual objective value would be $u(E - S^*) + f(S^*)$. Let's prove this.
  - (Weak duality:) $\mathbb{1}^T x = x(E) = x(S) + x(E - S) \leq f(S) + u(E - S)$. Thus we only have to show that this is satisfied with equality.
  - Suppose that x^* is primal optimal, and S and T are both x^*-tight, i.e., $x^*(S) = f(S)$ and $x^*(T) = f(T)$. Then (homework) both $S \cap T$ and $S \cup T$ are also x^*-tight.
  - Thus we can take the union of all x^*-tight sets to get S^*, which is also x^*-tight.
  - If $x_e^* < u_e$ then we must have that $e \in S^*$; if not, then we could feasibly increase x_e^*, contradicting optimality. Thus $x_e^* = u_e$ for all $e \notin S^*$.

LP optimal structure

- We believe that there exists an optimal solution to the dual where only one π_S is positive, say $\pi_{S^*} = 1$.
- Then the constraints $\sigma_e + \sum_{S \ni e} \pi_S = 1$ would force that $\sigma = \chi(E - S^*)$, and so the dual objective value would be $u(E - S^*) + f(S^*)$. Let's prove this.
  - (Weak duality:) $\mathbb{1}^T x = x(E) = x(S) + x(E - S) \leq f(S) + u(E - S)$. Thus we only have to show that this is satisfied with equality.
  - Suppose that x^* is primal optimal, and S and T are both x^*-tight, i.e., $x^*(S) = f(S)$ and $x^*(T) = f(T)$. Then (homework) both $S \cap T$ and $S \cup T$ are also x^*-tight.
  - Thus we can take the union of all x^*-tight sets to get S^*, which is also x^*-tight.
  - If $x_e^* < u_e$ then we must have that $e \in S^*$; if not, then we could feasibly increase x_e^*, contradicting optimality. Thus $x_e^* = u_e$ for all $e \notin S^*$.
  - Thus $x^*(E) = x^*(S^*) + x^*(E - S^*) = f(S^*) + u(E - S^*)$, proving that S^* induces a dual optimal solution.

Specialize the LP to get SFMin

- Our LP strong duality says that $\max_{x \in P(f): x \leq u} x(E) = \min_{S \subseteq E}(f(S) + u(E - S))$.

Specialize the LP to get SFMin

- Our LP strong duality says that $\max_{x \in P(f) : x \le u} x(E) = \min_{S \subseteq E} (f(S) + u(E - S))$.
- If we choose $u = 0$ then we get $\max_{x \in P(f) : x \le 0} x(E) = \min_{S \subseteq E} f(S) + 0$. This dual LP is just SFMin!

Specialize the LP to get SFMin

- Our LP strong duality says that $\max_{x \in P(f): x \leq u} x(E) = \min_{S \subseteq E}(f(S) + u(E - S))$.
- If we choose $u = 0$ then we get $\max_{x \in P(f): x \leq 0} x(E) = \min_{S \subseteq E} f(S) + 0$. This dual LP is just SFMin!
- For $y \in \mathbb{R}^E$ define $y^- \in \mathbb{R}^E$ via $y_e^- = \min(y_e, 0) \leq 0$.

Specialize the LP to get SFMin

- Our LP strong duality says that $\max_{x \in P(f): x \leq u} x(E) = \min_{S \subseteq E}(f(S) + u(E - S))$.
- If we choose $u = 0$ then we get $\max_{x \in P(f): x \leq 0} x(E) = \min_{S \subseteq E} f(S) + 0$. This dual LP is just SFMin!
- For $y \in \mathbb{R}^E$ define $y^- \in \mathbb{R}^E$ via $y_e^- = \min(y_e, 0) \leq 0$.
- If $y \in B(f)$ then $y^- \leq 0$ and $y^- \in P(F)$, so it is primal feasible.

Specialize the LP to get SFMin

- Our LP strong duality says that $\max_{x \in P(f): x \le u} x(E) = \min_{S \subseteq E}(f(S) + u(E - S))$.
- If we choose $u = 0$ then we get $\max_{x \in P(f): x \le 0} x(E) = \min_{S \subseteq E} f(S) + 0$. This dual LP is just SFMin!
- For $y \in \mathbb{R}^E$ define $y^- \in \mathbb{R}^E$ via $y_e^- = \min(y_e, 0) \le 0$.
- If $y \in B(f)$ then $y^- \le 0$ and $y^- \in P(F)$, so it is primal feasible.
- We now want to show the converse, that if $x \in P(f)$ and $x \le 0$, then there is some $y \in B(f)$ with $y \ge x$ and $y^- = x$.

Specialize the LP to get SFMin

- Our LP strong duality says that $\max_{x \in P(f): x \le u} x(E) = \min_{S \subseteq E}(f(S) + u(E - S))$.
- If we choose $u = 0$ then we get $\max_{x \in P(f): x \le 0} x(E) = \min_{S \subseteq E} f(S) + 0$. This dual LP is just SFMin!
- For $y \in \mathbb{R}^E$ define $y^- \in \mathbb{R}^E$ via $y_e^- = \min(y_e, 0) \le 0$.
- If $y \in B(f)$ then $y^- \le 0$ and $y^- \in P(F)$, so it is primal feasible.
- We now want to show the converse, that if $x \in P(f)$ and $x \le 0$, then there is some $y \in B(f)$ with $y \ge x$ and $y^- = x$.
- We know that an optimal $x^* \in P(f)$ with $x^* \le 0$ looks like:

$$(\underbrace{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0}_{\text{not in any } x^*\text{-tight set}}\ \underbrace{0\ 0\ 0\ 0\ -\ -\ -\ -\ -\ -\ -\ -}_{S^* = \text{ biggest } x^*\text{-tight set}})$$

Moving from $P(f)$ to $B(f)$

- We know that an optimal $x^* \in P(f)$ with $x^* \leq 0$ looks like:

$$(\underbrace{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0}_{\text{not in any } x^*\text{-tight set}}\ \underbrace{0\ 0\ 0\ 0\ -\ -\ -\ -\ -\ -\ -\ -}_{S^* = \text{ biggest } x^*\text{-tight set}})$$

Moving from $P(f)$ to $B(f)$

- We know that an optimal $x^* \in P(f)$ with $x^* \leq 0$ looks like:

$$(\underbrace{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0}_{\text{not in any } x^*\text{-tight set}}\ \underbrace{0\ 0\ 0\ 0\ -\ -\ -\ -\ -\ -\ -\ -}_{S^* = \text{ biggest } x^*\text{-tight set}})$$

- Set $y = x^*$ and pick some $e \notin S^*$ and increase y_e (making it positive) until it becomes tight (there is an exponential but finite number of constraints to check).

Moving from $P(f)$ to $B(f)$

- We know that an optimal $x^* \in P(f)$ with $x^* \leq 0$ looks like:

$$(\underbrace{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0}_{\text{not in any } x^*\text{-tight set}}\ \underbrace{0\ 0\ 0\ 0\ -\ -\ -\ -\ -\ -\ -\ -}_{S^* =\ \text{biggest } x^*\text{-tight set}})$$

- Set $y = x^*$ and pick some $e \notin S^*$ and increase y_e (making it positive) until it becomes tight (there is an exponential but finite number of constraints to check).
- Continue until every e is contained in an y-tight set.

Moving from $P(f)$ to $B(f)$

- We know that an optimal $x^* \in P(f)$ with $x^* \leq 0$ looks like:

$$(\underbrace{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0}_{\text{not in any } x^*\text{-tight set}}\ \underbrace{0\ 0\ 0\ 0\ -\ -\ -\ -\ -\ -\ -\ -}_{S^* = \text{ biggest } x^*\text{-tight set}})$$

- Set $y = x^*$ and pick some $e \notin S^*$ and increase y_e (making it positive) until it becomes tight (there is an exponential but finite number of constraints to check).
- Continue until every e is contained in an y-tight set.
- Now every e is in an y-tight set, and so E is tight, so the new y is in $B(f)$. It looks like:

$$(\underbrace{+\ +\ +\ +\ +\ +\ +\ +}_{\text{increased elements}}\ 0\ 0\ 0\ 0\ \underbrace{0\ 0\ 0\ 0\ -\ -\ -\ -\ -\ -\ -\ -}_{S^* \text{ is still } y\text{-tight}})$$

Moving from $P(f)$ to $B(f)$

- We know that an optimal $x^* \in P(f)$ with $x^* \leq 0$ looks like:

$$(\underbrace{0\;0\;0\;0\;0\;0\;0\;0\;0\;0\;0\;0}_{\text{not in any } x^*\text{-tight set}}\;\underbrace{0\;0\;0\;0\;-\;-\;-\;-\;-\;-\;-\;-}_{S^* = \text{ biggest } x^*\text{-tight set}})$$

- Set $y = x^*$ and pick some $e \notin S^*$ and increase y_e (making it positive) until it becomes tight (there is an exponential but finite number of constraints to check).
- Continue until every e is contained in an y-tight set.
- Now every e is in an y-tight set, and so E is tight, so the new y is in $B(f)$. It looks like:

$$(\underbrace{+\;+\;+\;+\;+\;+\;+\;+}_{\text{increased elements}}\;0\;0\;0\;0\;\underbrace{0\;0\;0\;0\;-\;-\;-\;-\;-\;-\;-\;-}_{S^* \text{ is still } y\text{-tight}})$$

- Thus we can use the modified primal LP $\max_{y \in B(f)} y^-(E)$.

Moving from $P(f)$ to $B(f)$

- We know that an optimal $x^* \in P(f)$ with $x^* \leq 0$ looks like:

$$(\underbrace{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0}_{\text{not in any } x^*\text{-tight set}}\ \underbrace{0\ 0\ 0\ 0\ -\ -\ -\ -\ -\ -\ -\ -}_{S^* = \text{ biggest } x^*\text{-tight set}})$$

- Set $y = x^*$ and pick some $e \notin S^*$ and increase y_e (making it positive) until it becomes tight (there is an exponential but finite number of constraints to check).
- Continue until every e is contained in an y-tight set.
- Now every e is in an y-tight set, and so E is tight, so the new y is in $B(f)$. It looks like:

$$(\underbrace{+\ +\ +\ +\ +\ +\ +\ +}_{\text{increased elements}}\ 0\ 0\ 0\ 0\ \underbrace{0\ 0\ 0\ 0\ -\ -\ -\ -\ -\ -\ -\ -}_{S^* \text{ is still } y\text{-tight}})$$

- Thus we can use the modified primal LP $\max_{y \in B(f)} y^-(E)$.
  - This is the form of the LP that we will use.

Moving from $P(f)$ to $B(f)$

- We know that an optimal $x^* \in P(f)$ with $x^* \leq 0$ looks like:

$$(\underbrace{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0}_{\text{not in any } x^*\text{-tight set}}\ \underbrace{0\ 0\ 0\ 0\ -\ -\ -\ -\ -\ -\ -\ -}_{S^* =\ \text{biggest } x^*\text{-tight set}})$$

- Set $y = x^*$ and pick some $e \notin S^*$ and increase y_e (making it positive) until it becomes tight (there is an exponential but finite number of constraints to check).
- Continue until every e is contained in an y-tight set.
- Now every e is in an y-tight set, and so E is tight, so the new y is in $B(f)$. It looks like:

$$(\underbrace{+\ +\ +\ +\ +\ +\ +\ +}_{\text{increased elements}}\ 0\ 0\ 0\ 0\ \underbrace{0\ 0\ 0\ 0\ -\ -\ -\ -\ -\ -\ -\ -}_{S^*\ \text{is still } y\text{-tight}})$$

- Thus we can use the modified primal LP $\max_{y \in B(f)} y^-(E)$.
  - This is the form of the LP that we will use.
  - This LP is quite close to the Greedy LP, except that the objective is the piecewise linear $y^-(E)$ instead of $x(E)$, and this makes solving the problem *much* harder.

Outline

SFMin weak duality, complementary slackness

- Here is weak duality for these LPs:

$$
\begin{aligned}
y^-(E) &\leq y^-(S) && \text{tight if } y_e < 0 \implies e \in S \\
&\leq y(S) && \text{tight if } e \in S \implies y_e \leq 0 \\
&\leq f(S) && \text{tight if } S \text{ is } y\text{-tight.}
\end{aligned}
$$

Complementary slackness is equivalent to the tightness conditions that ensure that each inequality is an equality.

SFMin weak duality, complementary slackness

- Here is weak duality for these LPs:

$$\begin{array}{rcll} y^-(E) & \leq & y^-(S) & \text{tight if } y_e < 0 \implies e \in S \\ & \leq & y(S) & \text{tight if } e \in S \implies y_e \leq 0 \\ & \leq & f(S) & \text{tight if } S \text{ is } y\text{-tight.} \end{array}$$

  Complementary slackness is equivalent to the tightness conditions that ensure that each inequality is an equality.

- Therefore an optimal y and S look like:

$$y = (\underbrace{-\ -\ -\ -\ 0\,0\,0\,0\,0\,0}_{S \text{ includes all } -, \text{ no } +}\ 0\,0\,0\,0\ +\ +\ +\ +\ +\ +\ +)$$

SFMin weak duality, complementary slackness

- Here is weak duality for these LPs:

$$\begin{array}{rcll} y^-(E) & \le & y^-(S) & \text{tight if } y_e < 0 \implies e \in S \\ & \le & y(S) & \text{tight if } e \in S \implies y_e \le 0 \\ & \le & f(S) & \text{tight if } S \text{ is } y\text{-tight.} \end{array}$$

  Complementary slackness is equivalent to the tightness conditions that ensure that each inequality is an equality.

- Therefore an optimal y and S look like:

$$y = (\underbrace{- \ - \ - \ - \ 0\,0\,0\,0\,0\,0}_{S \text{ includes all } -, \text{ no } +}\ 0\,0\,0\,0\ + \ + \ + \ + \ + \ + +)$$

- If we can achieve this picture along with $y(S) = f(S)$, it *proves* that y and S jointly solve SFMin.

SFMin weak duality, complementary slackness

- Here is weak duality for these LPs:

$$\begin{array}{rcll} y^-(E) & \leq & y^-(S) & \text{tight if } y_e < 0 \implies e \in S \\ & \leq & y(S) & \text{tight if } e \in S \implies y_e \leq 0 \\ & \leq & f(S) & \text{tight if } S \text{ is } y\text{-tight.} \end{array}$$

  Complementary slackness is equivalent to the tightness conditions that ensure that each inequality is an equality.

- Therefore an optimal y and S look like:

$$y = (\underbrace{-\ -\ -\ -\ 0\ 0\ 0\ 0\ 0\ 0}_{S \text{ includes all } -,\text{ no } +}\ 0\ 0\ 0\ 0\ +\ +\ +\ +\ +\ +\ +)$$

- If we can achieve this picture along with $y(S) = f(S)$, it *proves* that y and S jointly solve SFMin.
- Or does it? What is missing? ....

How do we know that $y \in B(f)$?

- How can we verify that $y \in B(f)$? There are 2^n inequalities to check.

How do we know that $y \in B(f)$?

- How can we verify that $y \in B(f)$? There are 2^n inequalities to check.
- Here is a clever way to do it (Cunningham):

How do we know that $y \in B(f)$?

- How can we verify that $y \in B(f)$? There are 2^n inequalities to check.
- Here is a clever way to do it (Cunningham):
  1. $B(f)$ is bounded, and so it is the convex hull of its vertices, i.e., $y \in B(f)$ iff y is a convex combination of vertices of $B(f)$.

How do we know that $y \in B(f)$?

- How can we verify that $y \in B(f)$? There are 2^n inequalities to check.
- Here is a clever way to do it (Cunningham):
  1. $B(f)$ is bounded, and so it is the convex hull of its vertices, i.e., $y \in B(f)$ iff y is a convex combination of vertices of $B(f)$.
  2. We know that all vertices of $B(f)$ come from Greedy applied to linear orders, which have succinct certificates.

How do we know that $y \in B(f)$?

- How can we verify that $y \in B(f)$? There are 2^n inequalities to check.
- Here is a clever way to do it (Cunningham):
  1. $B(f)$ is bounded, and so it is the convex hull of its vertices, i.e., $y \in B(f)$ iff y is a convex combination of vertices of $B(f)$.
  2. We know that all vertices of $B(f)$ come from Greedy applied to linear orders, which have succinct certificates.
  3. Carathéodory's Theorem says that in fact there is always a convex hull representation of y using at most n vertices.

How do we know that $y \in B(f)$?

- How can we verify that $y \in B(f)$? There are 2^n inequalities to check.
- Here is a clever way to do it (Cunningham):
  1. $B(f)$ is bounded, and so it is the convex hull of its vertices, i.e., $y \in B(f)$ iff y is a convex combination of vertices of $B(f)$.
  2. We know that all vertices of $B(f)$ come from Greedy applied to linear orders, which have succinct certificates.
  3. Carathéodory's Theorem says that in fact there is always a convex hull representation of y using at most n vertices.
- Therefore the algorithms will keep a representation of y like this:

How do we know that $y \in B(f)$?

- How can we verify that $y \in B(f)$? There are 2^n inequalities to check.
- Here is a clever way to do it (Cunningham):
  1. $B(f)$ is bounded, and so it is the convex hull of its vertices, i.e., $y \in B(f)$ iff y is a convex combination of vertices of $B(f)$.
  2. We know that all vertices of $B(f)$ come from Greedy applied to linear orders, which have succinct certificates.
  3. Carathéodory's Theorem says that in fact there is always a convex hull representation of y using at most n vertices.
- Therefore the algorithms will keep a representation of y like this:
  - We have an index set $\mathcal{I}$ of size $O(n)$.

How do we know that $y \in B(f)$?

- How can we verify that $y \in B(f)$? There are 2^n inequalities to check.
- Here is a clever way to do it (Cunningham):
  1. $B(f)$ is bounded, and so it is the convex hull of its vertices, i.e., $y \in B(f)$ iff y is a convex combination of vertices of $B(f)$.
  2. We know that all vertices of $B(f)$ come from Greedy applied to linear orders, which have succinct certificates.
  3. Carathéodory's Theorem says that in fact there is always a convex hull representation of y using at most n vertices.
- Therefore the algorithms will keep a representation of y like this:
  - We have an index set $\mathcal{I}$ of size $O(n)$.
  - For each $i \in \mathcal{I}$ we have a linear order $\prec_i$ with associated Greedy vertex v^i.

How do we know that $y \in B(f)$?

- How can we verify that $y \in B(f)$? There are 2^n inequalities to check.
- Here is a clever way to do it (Cunningham):
  1. $B(f)$ is bounded, and so it is the convex hull of its vertices, i.e., $y \in B(f)$ iff y is a convex combination of vertices of $B(f)$.
  2. We know that all vertices of $B(f)$ come from Greedy applied to linear orders, which have succinct certificates.
  3. Carathéodory's Theorem says that in fact there is always a convex hull representation of y using at most n vertices.
- Therefore the algorithms will keep a representation of y like this:
  - We have an index set $\mathcal{I}$ of size $O(n)$.
  - For each $i \in \mathcal{I}$ we have a linear order $\prec_i$ with associated Greedy vertex v^i.
  - We keep multipliers $\lambda_i \geq 0$ for $i \in \mathcal{I}$ satisfying $\sum_{i \in \mathcal{I}} \lambda_i = 1$.

How do we know that $y \in B(f)$?

- How can we verify that $y \in B(f)$? There are 2^n inequalities to check.
- Here is a clever way to do it (Cunningham):
  1. $B(f)$ is bounded, and so it is the convex hull of its vertices, i.e., $y \in B(f)$ iff y is a convex combination of vertices of $B(f)$.
  2. We know that all vertices of $B(f)$ come from Greedy applied to linear orders, which have succinct certificates.
  3. Carathéodory's Theorem says that in fact there is always a convex hull representation of y using at most n vertices.
- Therefore the algorithms will keep a representation of y like this:
  - We have an index set $\mathcal{I}$ of size $O(n)$.
  - For each $i \in \mathcal{I}$ we have a linear order $\prec_i$ with associated Greedy vertex v^i.
  - We keep multipliers $\lambda_i \geq 0$ for $i \in \mathcal{I}$ satisfying $\sum_{i \in \mathcal{I}} \lambda_i = 1$.
  - Then $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$ is a succinct certificate proving that $y \in B(f)$.

Keeping $|\mathcal{I}| = O(n)$

- As the algorithms proceed, they will add new indices to $\mathcal{I}$ (and possibly delete some old indices), and so $|\mathcal{I}|$ grows over time.

Keeping $|\mathcal{I}| = O(n)$

- As the algorithms proceed, they will add new indices to $\mathcal{I}$ (and possibly delete some old indices), and so $|\mathcal{I}|$ grows over time.
- When $\mathcal{I}$ becomes too large, from time to time we need to "Carathéodory-ize" it and bring its size back down to n.

Keeping $|\mathcal{I}| = O(n)$

- As the algorithms proceed, they will add new indices to $\mathcal{I}$ (and possibly delete some old indices), and so $|\mathcal{I}|$ grows over time.
- When $\mathcal{I}$ becomes too large, from time to time we need to "Carathéodory-ize" it and bring its size back down to n.
- Let V be the matrix with $|E| + 1$ rows and $|\mathcal{I}|$ columns which has a row of all ones at the top, and whose column i otherwise is v^i.

Keeping $|\mathcal{I}| = O(n)$

- As the algorithms proceed, they will add new indices to $\mathcal{I}$ (and possibly delete some old indices), and so $|\mathcal{I}|$ grows over time.
- When $\mathcal{I}$ becomes too large, from time to time we need to "Carathéodory-ize" it and bring its size back down to n.
- Let V be the matrix with $|E| + 1$ rows and $|\mathcal{I}|$ columns which has a row of all ones at the top, and whose column i otherwise is v^i.
- Therefore we keep the equation $V\lambda = (1 \quad y)$.

Keeping $|\mathcal{I}| = O(n)$

- As the algorithms proceed, they will add new indices to $\mathcal{I}$ (and possibly delete some old indices), and so $|\mathcal{I}|$ grows over time.
- When $\mathcal{I}$ becomes too large, from time to time we need to "Carathéodory-ize" it and bring its size back down to n.
- Let V be the matrix with $|E| + 1$ rows and $|\mathcal{I}|$ columns which has a row of all ones at the top, and whose column i otherwise is v^i.
- Therefore we keep the equation $V\lambda = (1 \quad y)$.
- The task of subroutine $\textsc{ReduceV}$ is to eliminate redundant columns of V while maintaining $V\lambda = (1 \quad y)$ and $\lambda \geq 0$.

Keeping $|\mathcal{I}| = O(n)$

- As the algorithms proceed, they will add new indices to $\mathcal{I}$ (and possibly delete some old indices), and so $|\mathcal{I}|$ grows over time.
- When $\mathcal{I}$ becomes too large, from time to time we need to "Carathéodory-ize" it and bring its size back down to n.
- Let V be the matrix with $|E| + 1$ rows and $|\mathcal{I}|$ columns which has a row of all ones at the top, and whose column i otherwise is v^i.
- Therefore we keep the equation $V\lambda = (1 \quad y)$.
- The task of subroutine $\textsc{ReduceV}$ is to eliminate redundant columns of V while maintaining $V\lambda = (1 \quad y)$ and $\lambda \geq 0$.
- This can be done with standard linear algebra techniques in $O(n^3)$ time.

Outline of a generic SFMin algorithm

- We keep linear orders $\prec_i$ with associated v^i, and $y \in B(f)$ as $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$.

Outline of a generic SFMin algorithm

- We keep linear orders $\prec_i$ with associated v^i, and $y \in B(f)$ as $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$.
- Suppose that y looks like:

$$y = (\underbrace{- \ - \ - \ - \ - \ -}_{S^-(y)} \ \underbrace{0\ 0\ 0\ 0\ 0}_{S^0(y)} \ \underbrace{+ \ + \ + \ +}_{S^+(y)})$$

Outline of a generic SFMin algorithm

- We keep linear orders $\prec_i$ with associated v^i, and $y \in B(f)$ as $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$.
- Suppose that y looks like:

$$y = (\underbrace{- \; - \; - \; - \; - \; -}_{S^-(y)} \; \underbrace{0\;0\;0\;0\;0}_{S^0(y)} \; \underbrace{+ \; + \; + \; +}_{S^+(y)})$$

- To maximize $y^-(E)$ ($\iff \min_S y^+(E)$), we want to increase y_e for some $e \in S^-(y)$ (or decrease y_e for some $e \in S^+(y)$).

Outline of a generic SFMin algorithm

- We keep linear orders $\prec_i$ with associated v^i, and $y \in B(f)$ as $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$.
- Suppose that y looks like:

$$y = (\underbrace{- \;\; - \;\; - \;\; - \;\; - \; -}_{S^-(y)} \;\; \underbrace{0\;0\;0\;0\;0}_{S^0(y)} \;\; \underbrace{+ \;\; + \;\; + \; +}_{S^+(y)})$$

- To maximize $y^-(E)$ ($\iff \min_S y^+(E)$), we want to increase y_e for some $e \in S^-(y)$ (or decrease y_e for some $e \in S^+(y)$).
  - We know that y_e increases if we move e to the left in some $\prec_i$, and y_e decreases if we move e to the right in some $\prec_i$.

Outline of a generic SFMin algorithm

- We keep linear orders $\prec_i$ with associated v^i, and $y \in B(f)$ as $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$.
- Suppose that y looks like:

$$y = (\underbrace{-\ -\ -\ -\ -\ -}_{S^-(y)}\ \underbrace{0\ 0\ 0\ 0\ 0}_{S^0(y)}\ \underbrace{+\ +\ +\ +}_{S^+(y)})$$

- To maximize $y^-(E)$ ($\iff \min_S y^+(E)$), we want to increase y_e for some $e \in S^-(y)$ (or decrease y_e for some $e \in S^+(y)$).
  - We know that y_e increases if we move e to the left in some $\prec_i$, and y_e decreases if we move e to the right in some $\prec_i$.
  - This suggests we find some $k \in S^-(y)$ and $l \in S^+(y)$ and compute $c(k, l; y)$, then set $y' \leftarrow y + \alpha(\chi_k - \chi_l)$ for some $\alpha \le c(k, l; y)$.

Outline of a generic SFMin algorithm

- We keep linear orders $\prec_i$ with associated v^i, and $y \in B(f)$ as $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$.
- Suppose that y looks like:

$$y = (\underbrace{- \ - \ - \ - \ - \ -}_{S^-(y)} \ \underbrace{0\ 0\ 0\ 0\ 0}_{S^0(y)} \ \underbrace{+ \ + \ + \ +}_{S^+(y)})$$

- To maximize $y^-(E)$ ($\iff \min_S y^+(E)$), we want to increase y_e for some $e \in S^-(y)$ (or decrease y_e for some $e \in S^+(y)$).
  - We know that y_e increases if we move e to the left in some $\prec_i$, and y_e decreases if we move e to the right in some $\prec_i$.
  - This suggests we find some $k \in S^-(y)$ and $l \in S^+(y)$ and compute $c(k, l; y)$, then set $y' \leftarrow y + \alpha(\chi_k - \chi_l)$ for some $\alpha \leq c(k, l; y)$.
  - But unfortunately computing $c(k, l; y)$ is as hard as SFMin.

Outline of a generic SFMin algorithm

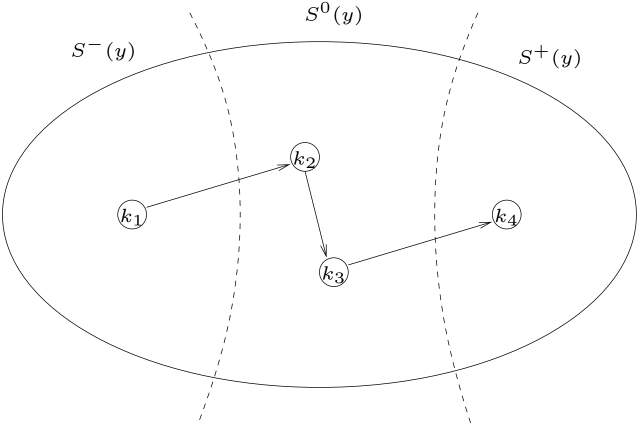
- We keep linear orders $\prec_i$ with associated v^i, and $y \in B(f)$ as $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$.
- Suppose that y looks like:

$$y = (\underbrace{- \;\; - \;\; - \;\; - \;\; - \; -}_{S^-(y)} \; \underbrace{0\;0\;0\;0\;0}_{S^0(y)} \; \underbrace{+ \;\; + \;\; + \; +}_{S^+(y)})$$

- To maximize $y^-(E)$ ($\iff \min_S y^+(E)$), we want to increase y_e for some $e \in S^-(y)$ (or decrease y_e for some $e \in S^+(y)$).
  - We know that y_e increases if we move e to the left in some $\prec_i$, and y_e decreases if we move e to the right in some $\prec_i$.
  - This suggests we find some $k \in S^-(y)$ and $l \in S^+(y)$ and compute $c(k, l; y)$, then set $y' \leftarrow y + \alpha(\chi_k - \chi_l)$ for some $\alpha \leq c(k, l; y)$.
  - But unfortunately computing $c(k, l; y)$ is as hard as SFMin.
  - And if we don't have any $\prec_i$ with (l, k) consecutive in $\prec_i$, then how can we change the representation $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$ to track this $\chi_k - \chi_l$ direction?

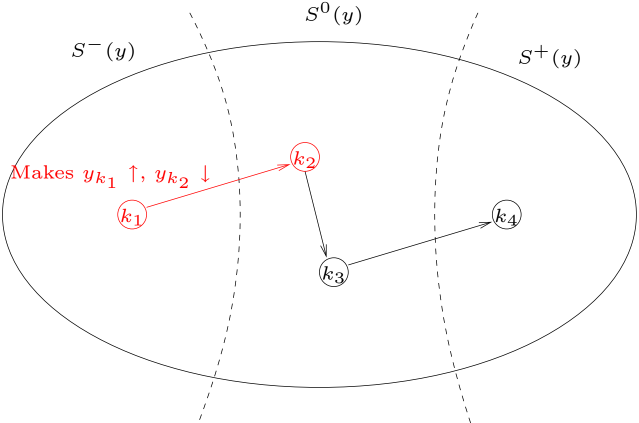
SFMin augmenting paths

Assume that we have the situation as in the picture below, where (k_2, k_1) is consecutive in $\prec_1$, (k_3, k_2) is consecutive in $\prec_2$, and (k_4, k_3) is consecutive in $\prec_3$.



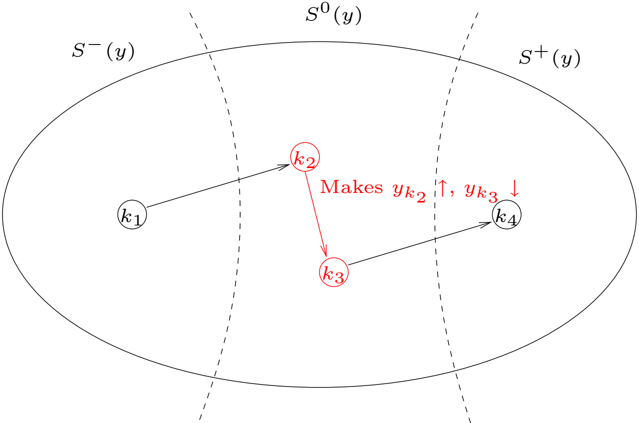
SFMin augmenting paths

Assume that we have the situation as in the picture below, where (k_2, k_1) is consecutive in $\prec_1$, (k_3, k_2) is consecutive in $\prec_2$, and (k_4, k_3) is consecutive in $\prec_3$.
We could swap k_2 and k_1 in $\prec_1$ to $\uparrow y_{k_1}$ and $\downarrow y_{k_2}$, but this wouldn't increase $y^-(E)$.



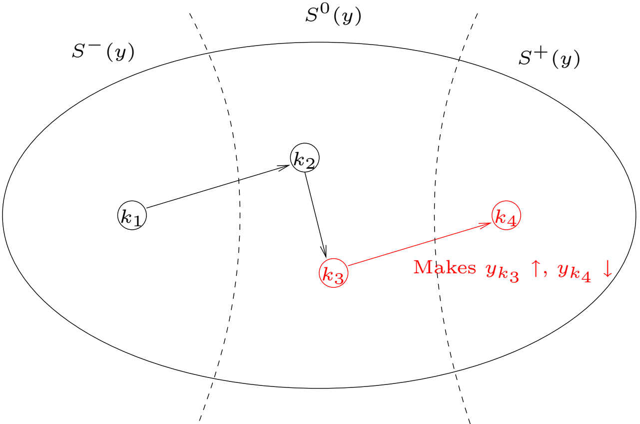
SFMin augmenting paths

Assume that we have the situation as in the picture below, where (k_2, k_1) is consecutive in $\prec_1$, (k_3, k_2) is consecutive in $\prec_2$, and (k_4, k_3) is consecutive in $\prec_3$.
We could swap k_3 and k_2 in $\prec_2$ to $\uparrow y_{k_2}$ and $\downarrow y_{k_3}$, but this wouldn't increase $y^-(E)$.



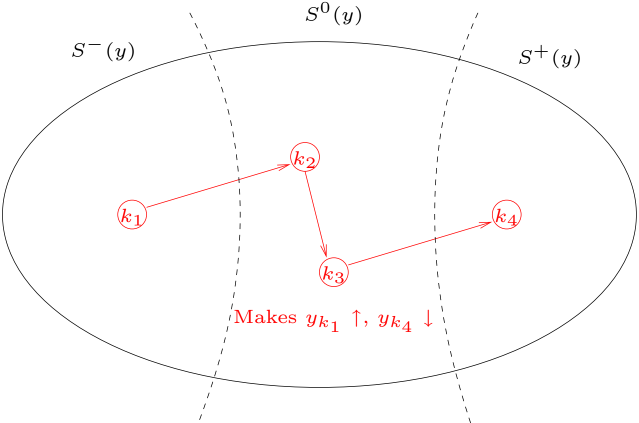
SFMin augmenting paths

Assume that we have the situation as in the picture below, where (k_2, k_1) is consecutive in $\prec_1$, (k_3, k_2) is consecutive in $\prec_2$, and (k_4, k_3) is consecutive in $\prec_3$.
We could swap k_4 and k_3 in $\prec_3$ to $\uparrow y_{k_3}$ and $\downarrow y_{k_4}$, but this wouldn't increase $y^-(E)$.



SFMin augmenting paths

Assume that we have the situation as in the picture below, where (k_2, k_1) is consecutive in $\prec_1$, (k_3, k_2) is consecutive in $\prec_2$, and (k_4, k_3) is consecutive in $\prec_3$.
But if we do all three swaps at the same time this would $\uparrow y_{k_1}$ and $\downarrow y_{k_4}$, and this would increase $y^-(E)$.



SFMin is like Max Flow / Min Cut

- This suggests a rudimentary algorithm:

SFMin is like Max Flow / Min Cut

- This suggests a rudimentary algorithm:
  1. Make a network with nodes E, and arc $e \to g$ whenever (g, e) is consecutive in some $\prec_i$.

SFMin is like Max Flow / Min Cut

- This suggests a rudimentary algorithm:
  1. Make a network with nodes E, and arc $e \to g$ whenever (g, e) is consecutive in some $\prec_i$.
  2. If there is a directed path from $S^-(y)$ to $S^+(y)$ then augment along it; repeat until no such path remains.

SFMin is like Max Flow / Min Cut

- This suggests a rudimentary algorithm:
  1. Make a network with nodes E, and arc $e \to g$ whenever (g, e) is consecutive in some $\prec_i$.
  2. If there is a directed path from $S^-(y)$ to $S^+(y)$ then augment along it; repeat until no such path remains.
  3. If $\nexists$ a directed path from $S^-(y)$ to $S^+(y)$, define $S^* = \{e \in E \mid \exists$ augmenting path from $S^-(y)$ up to $e\}$. Then we show below that S^* solves SFMin.

SFMin is like Max Flow / Min Cut

- This suggests a rudimentary algorithm:
  1. Make a network with nodes E, and arc $e \to g$ whenever (g, e) is consecutive in some $\prec_i$.
  2. If there is a directed path from $S^-(y)$ to $S^+(y)$ then augment along it; repeat until no such path remains.
  3. If $\nexists$ a directed path from $S^-(y)$ to $S^+(y)$, define $S^* = \{e \in E \mid \exists$ augmenting path from $S^-(y)$ up to $e\}$. Then we show below that S^* solves SFMin.
- Note that $S^-(y) \subseteq S^* \subseteq E - S^+(y)$, so S^* satisfies two of the three complementary slackness conditions.

SFMin is like Max Flow / Min Cut

- This suggests a rudimentary algorithm:
  1. Make a network with nodes E, and arc $e \to g$ whenever (g, e) is consecutive in some $\prec_i$.
  2. If there is a directed path from $S^-(y)$ to $S^+(y)$ then augment along it; repeat until no such path remains.
  3. If $\nexists$ a directed path from $S^-(y)$ to $S^+(y)$, define $S^* = \{e \in E \mid \exists$ augmenting path from $S^-(y)$ up to $e\}$. Then we show below that S^* solves SFMin.
- Note that $S^-(y) \subseteq S^* \subseteq E - S^+(y)$, so S^* satisfies two of the three complementary slackness conditions.
- I claim that S^* is at the left of every $\prec_i$.

SFMin is like Max Flow / Min Cut

- This suggests a rudimentary algorithm:
  1. Make a network with nodes E, and arc $e \to g$ whenever (g, e) is consecutive in some $\prec_i$.
  2. If there is a directed path from $S^-(y)$ to $S^+(y)$ then augment along it; repeat until no such path remains.
  3. If $\nexists$ a directed path from $S^-(y)$ to $S^+(y)$, define $S^* = \{e \in E \mid \exists$ augmenting path from $S^-(y)$ up to $e\}$. Then we show below that S^* solves SFMin.
- Note that $S^-(y) \subseteq S^* \subseteq E - S^+(y)$, so S^* satisfies two of the three complementary slackness conditions.
- I claim that S^* is at the left of every $\prec_i$.
  - Suppose that there is some $\prec_i$ with $l \notin S^*$ to the left of some $k \in S^*$.

SFMin is like Max Flow / Min Cut

- This suggests a rudimentary algorithm:
  1. Make a network with nodes E, and arc $e \to g$ whenever (g, e) is consecutive in some $\prec_i$.
  2. If there is a directed path from $S^-(y)$ to $S^+(y)$ then augment along it; repeat until no such path remains.
  3. If $\nexists$ a directed path from $S^-(y)$ to $S^+(y)$, define $S^* = \{e \in E \mid \exists$ augmenting path from $S^-(y)$ up to $e\}$. Then we show below that S^* solves SFMin.
- Note that $S^-(y) \subseteq S^* \subseteq E - S^+(y)$, so S^* satisfies two of the three complementary slackness conditions.
- I claim that S^* is at the left of every $\prec_i$.
  - Suppose that there is some $\prec_i$ with $l \notin S^*$ to the left of some $k \in S^*$.
  - Then there must be such a pair (l, k) that is consecutive in $\prec_i$.

SFMin is like Max Flow / Min Cut

- This suggests a rudimentary algorithm:
  1. Make a network with nodes E, and arc $e \to g$ whenever (g, e) is consecutive in some $\prec_i$.
  2. If there is a directed path from $S^-(y)$ to $S^+(y)$ then augment along it; repeat until no such path remains.
  3. If $\nexists$ a directed path from $S^-(y)$ to $S^+(y)$, define $S^* = \{e \in E \mid \exists$ augmenting path from $S^-(y)$ up to $e\}$. Then we show below that S^* solves SFMin.
- Note that $S^-(y) \subseteq S^* \subseteq E - S^+(y)$, so S^* satisfies two of the three complementary slackness conditions.
- I claim that S^* is at the left of every $\prec_i$.
  - Suppose that there is some $\prec_i$ with $l \notin S^*$ to the left of some $k \in S^*$.
  - Then there must be such a pair (l, k) that is consecutive in $\prec_i$.
  - But then we could extend the augmenting path to k along arc $k \to l$ coming from consecutive pair (l, k), contradicting that $l \notin S^*$.

SFMin is like Max Flow / Min Cut

- I just showed that S^* is at the left of every $\prec_i$.

SFMin is like Max Flow / Min Cut

- I just showed that S^* is at the left of every $\prec_i$.
- Now S^* at the left of $\prec_i$ implies that $v^i(S^*) = f(S^*)$.

SFMin is like Max Flow / Min Cut

- I just showed that S^* is at the left of every $\prec_i$.
- Now S^* at the left of $\prec_i$ implies that $v^i(S^*) = f(S^*)$.
- Then $y(S^*) = \sum_{i \in \mathcal{I}} \lambda_i v^i(S^*) = \sum_{i \in \mathcal{I}} \lambda_i f(S^*) = f(S^*) \sum_{i \in \mathcal{I}} \lambda_i = f(S^*)$.

SFMin is like Max Flow / Min Cut

- I just showed that S^* is at the left of every $\prec_i$.
- Now S^* at the left of $\prec_i$ implies that $v^i(S^*) = f(S^*)$.
- Then $y(S^*) = \sum_{i \in \mathcal{I}} \lambda_i v^i(S^*) = \sum_{i \in \mathcal{I}} \lambda_i f(S^*) = f(S^*) \sum_{i \in \mathcal{I}} \lambda_i = f(S^*)$.
- Thus S^* is y-tight, the third complementary slackness condition, and so S^* is indeed optimal for SFMin.

SFMin is like Max Flow / Min Cut

- I just showed that S^* is at the left of every $\prec_i$.
- Now S^* at the left of $\prec_i$ implies that $v^i(S^*) = f(S^*)$.
- Then $y(S^*) = \sum_{i \in \mathcal{I}} \lambda_i v^i(S^*) = \sum_{i \in \mathcal{I}} \lambda_i f(S^*) = f(S^*) \sum_{i \in \mathcal{I}} \lambda_i = f(S^*)$.
- Thus S^* is y-tight, the third complementary slackness condition, and so S^* is indeed optimal for SFMin.
- This proof is very much in the same spirit as the Max Flow / Min Cut augmenting path proof.

SFMin is like Max Flow / Min Cut

- I just showed that S^* is at the left of every $\prec_i$.
- Now S^* at the left of $\prec_i$ implies that $v^i(S^*) = f(S^*)$.
- Then $y(S^*) = \sum_{i\in\mathcal{I}} \lambda_i v^i(S^*) = \sum_{i\in\mathcal{I}} \lambda_i f(S^*) = f(S^*) \sum_{i\in\mathcal{I}} \lambda_i = f(S^*)$.
- Thus S^* is y-tight, the third complementary slackness condition, and so S^* is indeed optimal for SFMin.
- This proof is very much in the same spirit as the Max Flow / Min Cut augmenting path proof.
- The same proof works with a more general definition of arcs: Put $e \to g \in A$ whenever $g \prec_i e$ for some $i \in \mathcal{I}$.

SFMin is like Max Flow / Min Cut

- I just showed that S^* is at the left of every $\prec_i$.
- Now S^* at the left of $\prec_i$ implies that $v^i(S^*) = f(S^*)$.
- Then $y(S^*) = \sum_{i \in \mathcal{I}} \lambda_i v^i(S^*) = \sum_{i \in \mathcal{I}} \lambda_i f(S^*) = f(S^*) \sum_{i \in \mathcal{I}} \lambda_i = f(S^*)$.
- Thus S^* is y-tight, the third complementary slackness condition, and so S^* is indeed optimal for SFMin.
- This proof is very much in the same spirit as the Max Flow / Min Cut augmenting path proof.
- The same proof works with a more general definition of arcs: Put $e \to g \in A$ whenever $g \prec_i e$ for some $i \in \mathcal{I}$.
- The "only" remaining thing to do is to find some way to arrange augmentations so there is only a polynomial number of them.

SFMin is *not* like Max Flow / Min Cut

- The set of arcs changes dynamically as $\mathcal{I}$ changes and y changes.

SFMin is *not* like Max Flow / Min Cut

- The set of arcs changes dynamically as $\mathcal{I}$ changes and y changes.
- The "capacity" of arcs changes dynamically.

SFMin is *not* like Max Flow / Min Cut

- The set of arcs changes dynamically as $\mathcal{I}$ changes and y changes.
- The "capacity" of arcs changes dynamically.
- One augmenting path could contain several arcs coming from the same $\prec_i$, implying that computing the augmentation amount is quite complicated.

SFMin is *not* like Max Flow / Min Cut

- The set of arcs changes dynamically as $\mathcal{I}$ changes and y changes.
- The "capacity" of arcs changes dynamically.
- One augmenting path could contain several arcs coming from the same $\prec_i$, implying that computing the augmentation amount is quite complicated.
- Augmentation amounts depend on the λ_i, which can be arbitrarily small.

SFMin is *not* like Max Flow / Min Cut

- The set of arcs changes dynamically as $\mathcal{I}$ changes and y changes.
- The "capacity" of arcs changes dynamically.
- One augmenting path could contain several arcs coming from the same $\prec_i$, implying that computing the augmentation amount is quite complicated.
- Augmentation amounts depend on the λ_i, which can be arbitrarily small.
- These are some of the reasons why it took many, many years to figure out how to get a combinatorial SFMin algorithm, and why Cunningham's SFMin algorithm was only pseudo-polynomial.

Current state of the art in SFMin

(Taken from S. T. McCormick (2006). Submodular Function Minimization. Chapter 7 in the *Handbook on Discrete Optimization*, Elsevier, K. Aardal, G. Nemhauser, and R. Weismantel, eds, 321–391.; see my webpage for updated version.)

	Cunningham for General SFM [13], Sec. 3.1	Schrijver [76, 84], Schrijver-PR [22], Sec. 3.2	Iwata, Fleischer, and Fujishige [49, 45], Sec. 3.3	Iwata Hybrid [47], Sec. 3.3.4	Orlin [71], Sec. 3.4.1	Iwata and Orlin [51], Sec. 3.4.2
Pseudo-polyn. running time	$O(Mn^6 \log(Mn) \cdot$ EO)					
Weakly polyn. running time			$O(n^5 \text{EO} \log M)$ [49], Sec. 3.3.1	$O((n^4\text{EO} + n^5) \cdot \log M)$ **(*)**		$O((n^4\text{EO} + n^5) \cdot \log(nM))$
Strongly polyn. running time		$O(n^7\text{EO} + n^8)$ [22, 84]	$O(n^7 \text{EO} \log n)$ [49], Sec. 3.3.2	$O((n^6\text{EO} + n^7) \cdot \log n)$	$O(n^5\text{EO} + n^6)$ **(*)**	$O((n^5\text{EO} + n^6) \log n)$
Fully comb. running time			$O(n^9 \text{EO} \log^2 n)$ [45], Sec. 3.3.3	$O(n^8 \text{EO} \log^2 n)$		$O((n^7\text{EO} + n^8) \log n)$ **(*)**

Outline

Why is Carathéodory bad?

1. The Carathéodory subroutine $\textsc{ReduceV}$ is a bottleneck in the worst-case running time of the fastest SFMin algorithms.

Why is Carathéodory bad?

1. The Carathéodory subroutine $\textsc{ReduceV}$ is a bottleneck in the worst-case running time of the fastest SFMin algorithms.
2. The linear algebra involved in $\textsc{ReduceV}$ is ugly and produces highly fractional λ_i in general.

Why is Carathéodory bad?

1. The Carathéodory subroutine $\textsc{ReduceV}$ is a bottleneck in the worst-case running time of the fastest SFMin algorithms.
2. The linear algebra involved in $\textsc{ReduceV}$ is ugly and produces highly fractional λ_i in general.
   - With integral $f(S)$, it is easy to prove that there is always an integral y^* that solves the dual of SFMin.

Why is Carathéodory bad?

1. The Carathéodory subroutine $\textsc{ReduceV}$ is a bottleneck in the worst-case running time of the fastest SFMin algorithms.
2. The linear algebra involved in $\textsc{ReduceV}$ is ugly and produces highly fractional λ_i in general.
   - With integral $f(S)$, it is easy to prove that there is always an integral y^* that solves the dual of SFMin.
3. The linear algebra of $\textsc{ReduceV}$ is also a bottleneck in the empirical running time of SFMin algorithms (Iwata).

Why is Carathéodory bad?

1. The Carathéodory subroutine $\textsc{ReduceV}$ is a bottleneck in the worst-case running time of the fastest SFMin algorithms.
2. The linear algebra involved in $\textsc{ReduceV}$ is ugly and produces highly fractional λ_i in general.
   - With integral $f(S)$, it is easy to prove that there is always an integral y^* that solves the dual of SFMin.
3. The linear algebra of $\textsc{ReduceV}$ is also a bottleneck in the empirical running time of SFMin algorithms (Iwata).
4. Potentially, replacing $\textsc{ReduceV}$ by something more combinatorial would be a way to get a faster SFMin algorithm.

Why is Carathéodory bad?

1. The Carathéodory subroutine $\textsc{ReduceV}$ is a bottleneck in the worst-case running time of the fastest SFMin algorithms.
2. The linear algebra involved in $\textsc{ReduceV}$ is ugly and produces highly fractional λ_i in general.
   - With integral $f(S)$, it is easy to prove that there is always an integral y^* that solves the dual of SFMin.
3. The linear algebra of $\textsc{ReduceV}$ is also a bottleneck in the empirical running time of SFMin algorithms (Iwata).
4. Potentially, replacing $\textsc{ReduceV}$ by something more combinatorial would be a way to get a faster SFMin algorithm.

- **Challenge:** Find a linear algebra-free way to prove that the current y belongs to $B(f)$. This new method should be:

Why is Carathéodory bad?

1. The Carathéodory subroutine $\textsc{ReduceV}$ is a bottleneck in the worst-case running time of the fastest SFMin algorithms.
2. The linear algebra involved in $\textsc{ReduceV}$ is ugly and produces highly fractional λ_i in general.
   - With integral $f(S)$, it is easy to prove that there is always an integral y^* that solves the dual of SFMin.
3. The linear algebra of $\textsc{ReduceV}$ is also a bottleneck in the empirical running time of SFMin algorithms (Iwata).
4. Potentially, replacing $\textsc{ReduceV}$ by something more combinatorial would be a way to get a faster SFMin algorithm.

- **Challenge:** Find a linear algebra-free way to prove that the current y belongs to $B(f)$. This new method should be:
  1. Efficient: Calling $\textsc{ReduceV}$ costs $O(n^3)$ time, so the new method needs to be at least this fast.

Why is Carathéodory bad?

1. The Carathéodory subroutine $\textsc{ReduceV}$ is a bottleneck in the worst-case running time of the fastest SFMin algorithms.
2. The linear algebra involved in $\textsc{ReduceV}$ is ugly and produces highly fractional λ_i in general.
   - With integral $f(S)$, it is easy to prove that there is always an integral y^* that solves the dual of SFMin.
3. The linear algebra of $\textsc{ReduceV}$ is also a bottleneck in the empirical running time of SFMin algorithms (Iwata).
4. Potentially, replacing $\textsc{ReduceV}$ by something more combinatorial would be a way to get a faster SFMin algorithm.

- **Challenge:** Find a linear algebra-free way to prove that the current y belongs to $B(f)$. This new method should be:
  1. Efficient: Calling $\textsc{ReduceV}$ costs $O(n^3)$ time, so the new method needs to be at least this fast.
  2. Integral: It should work without using any multiplication or division, i.e., no linear algebra, and it should be able to maintain that y is always integral.

Outline

- Optimizing submodular functions
  - The Greedy Algorithm
  - Edges of $B(f)$
- SFMin algorithms
  - An algorithmic framework
  - Algorithm-izing the dual LPs
- Combinatorial Hull
  - Carathéodory is a bottleneck
  - Avoiding linear algebra
  - Combinatorial hull and membership
  - Algorithmic ideas for combinatorial hull

A useful theorem?

- Recall that if y, $y' \in B(f)$, then $y(E) = y'(E) = f(E)$.

A useful theorem?

- Recall that if y, $y' \in B(f)$, then $y(E) = y'(E) = f(E)$.
  - Thus we can't have $y \geq y'$ with $y \neq y'$.

A useful theorem?

- Recall that if y, $y' \in B(f)$, then $y(E) = y'(E) = f(E)$.
  - Thus we can't have $y \geq y'$ with $y \neq y'$.
- Use tilde to represent projecting out the first component, so that $\tilde{E} = E - \{1\}$ and $\tilde{y} = (y_2, y_3, \ldots, y_n)$.

A useful theorem?

- Recall that if y, $y' \in B(f)$, then $y(E) = y'(E) = f(E)$.
  - Thus we can't have $y \geq y'$ with $y \neq y'$.
- Use tilde to represent projecting out the first component, so that $\tilde{E} = E - \{1\}$ and $\tilde{y} = (y_2, y_3, \ldots, y_n)$.
  - Now we can have $\tilde{y} \geq \tilde{y}'$ with $\tilde{y} \neq \tilde{y}'$.

A useful theorem?

- Recall that if y, $y' \in B(f)$, then $y(E) = y'(E) = f(E)$.
  - Thus we can't have $y \geq y'$ with $y \neq y'$.
- Use tilde to represent projecting out the first component, so that $\tilde{E} = E - \{1\}$ and $\tilde{y} = (y_2, y_3, \ldots, y_n)$.
  - Now we can have $\tilde{y} \geq \tilde{y}'$ with $\tilde{y} \neq \tilde{y}'$.
- Suppose that we have x, $z \in B(f)$, $y(E) = f(E)$, and $\tilde{x} \leq \tilde{y} \leq \tilde{z}$.

A useful theorem?

- Recall that if y, $y' \in B(f)$, then $y(E) = y'(E) = f(E)$.
  - Thus we can't have $y \geq y'$ with $y \neq y'$.
- Use tilde to represent projecting out the first component, so that $\tilde{E} = E - \{1\}$ and $\tilde{y} = (y_2, y_3, \ldots, y_n)$.
  - Now we can have $\tilde{y} \geq \tilde{y}'$ with $\tilde{y} \neq \tilde{y}'$.
- Suppose that we have x, $z \in B(f)$, $y(E) = f(E)$, and $\tilde{x} \leq \tilde{y} \leq \tilde{z}$.
- **Theorem (Fujishige):** Then $y \in B(f)$.

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \leq f(S)$.

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \leq f(S)$.
- Case 1: Suppose that $1 \notin S$, so that $\tilde{S} = S$.

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \leq f(S)$.
- Case 1: Suppose that $1 \notin S$, so that $\tilde{S} = S$.
  - Then $y(S) = y(\tilde{S}) \leq z(\tilde{S}) = z(S) \leq f(S)$.

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \leq f(S)$.
- Case 1: Suppose that $1 \notin S$, so that $\tilde{S} = S$.
  - Then $y(S) = y(\tilde{S}) \leq z(\tilde{S}) = z(S) \leq f(S)$.
- Case 2: Suppose that $1 \in S$, so that $\tilde{S} = S - \{1\}$.

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \leq f(S)$.
- Case 1: Suppose that $1 \notin S$, so that $\tilde{S} = S$.
  - Then $y(S) = y(\tilde{S}) \leq z(\tilde{S}) = z(S) \leq f(S)$.
- Case 2: Suppose that $1 \in S$, so that $\tilde{S} = S - \{1\}$.
  - Then $y(E) = f(E)$ implies that $y_1 = f(E) - \tilde{y}(\tilde{E})$.

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \le f(S)$.
- Case 1: Suppose that $1 \notin S$, so that $\tilde{S} = S$.
  - Then $y(S) = y(\tilde{S}) \le z(\tilde{S}) = z(S) \le f(S)$.
- Case 2: Suppose that $1 \in S$, so that $\tilde{S} = S - \{1\}$.
  - Then $y(E) = f(E)$ implies that $y_1 = f(E) - \tilde{y}(\tilde{E})$.
  - Thus $y(S) = \tilde{y}(\tilde{S}) + y_1 = \tilde{y}(\tilde{S}) + f(E) - \tilde{y}(\tilde{E}) = f(E) - \tilde{y}(\tilde{E} - \tilde{S}) \le f(E) - \tilde{x}(\tilde{E} - \tilde{S}) = \tilde{x}(\tilde{S}) + f(E) - \tilde{x}(\tilde{E}) = x(\tilde{S}) + x_1 = x(S) \le f(S)$.

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \le f(S)$.
- Case 1: Suppose that $1 \notin S$, so that $\tilde{S} = S$.
  - Then $y(S) = y(\tilde{S}) \le z(\tilde{S}) = z(S) \le f(S)$.
- Case 2: Suppose that $1 \in S$, so that $\tilde{S} = S - \{1\}$.
  - Then $y(E) = f(E)$ implies that $y_1 = f(E) - \tilde{y}(\tilde{E})$.
  - Thus $y(S) = \tilde{y}(\tilde{S}) + y_1 = \tilde{y}(\tilde{S}) + f(E) - \tilde{y}(\tilde{E}) = f(E) - \tilde{y}(\tilde{E} - \tilde{S}) \le f(E) - \tilde{x}(\tilde{E} - \tilde{S}) = \tilde{x}(\tilde{S}) + f(E) - \tilde{x}(\tilde{\tilde{E}}) = x(\tilde{S}) + x_1 = x(S) \le f(S)$.
- These show that $y \in B(f)$ iff $y(E) = f(E)$ and

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \leq f(S)$.
- Case 1: Suppose that $1 \notin S$, so that $\tilde{S} = S$.
  - Then $y(S) = y(\tilde{S}) \leq z(\tilde{S}) = z(S) \leq f(S)$.
- Case 2: Suppose that $1 \in S$, so that $\tilde{S} = S - \{1\}$.
  - Then $y(E) = f(E)$ implies that $y_1 = f(E) - \tilde{y}(\tilde{E})$.
  - Thus $y(S) = \tilde{y}(\tilde{S}) + y_1 = \tilde{y}(\tilde{S}) + f(E) - \tilde{y}(\tilde{E}) = f(E) - \tilde{y}(\tilde{E} - \tilde{S}) \leq f(E) - \tilde{x}(\tilde{E} - \tilde{S}) = \tilde{x}(\tilde{S}) + f(E) - \tilde{x}(\tilde{\tilde{E}}) = x(\tilde{S}) + x_1 = x(S) \leq f(S)$.
- These show that $y \in B(f)$ iff $y(E) = f(E)$ and
  - $\tilde{y}(S) \leq \tilde{f}(S)\ \forall\ S \subseteq \tilde{E}$, and

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \le f(S)$.
- Case 1: Suppose that $1 \notin S$, so that $\tilde{S} = S$.
  - Then $y(S) = y(\tilde{S}) \le z(\tilde{S}) = z(S) \le f(S)$.
- Case 2: Suppose that $1 \in S$, so that $\tilde{S} = S - \{1\}$.
  - Then $y(E) = f(E)$ implies that $y_1 = f(E) - \tilde{y}(\tilde{E})$.
  - Thus $y(S) = \tilde{y}(\tilde{S}) + y_1 = \tilde{y}(\tilde{S}) + f(E) - \tilde{y}(\tilde{E}) = f(E) - \tilde{y}(\tilde{E} - \tilde{S}) \le f(E) - \tilde{x}(\tilde{E} - \tilde{S}) = \tilde{x}(\tilde{S}) + f(E) - \tilde{x}(\tilde{\tilde{E}}) = x(\tilde{S}) + x_1 = x(S) \le f(S)$.
- These show that $y \in B(f)$ iff $y(E) = f(E)$ and
  - $\tilde{y}(S) \le \tilde{f}(S) \ \forall \ S \subseteq \tilde{E}$, and
  - $\tilde{y}(T) \ge \tilde{f}^{\#}(T) \equiv f(E) - f(T) \ \forall \ T$ s.t. $T \subseteq \tilde{E}$ (with $S \equiv E - T$, so that $1 \in S$).

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \le f(S)$.
- Case 1: Suppose that $1 \notin S$, so that $\tilde{S} = S$.
  - Then $y(S) = y(\tilde{S}) \le z(\tilde{S}) = z(S) \le f(S)$.
- Case 2: Suppose that $1 \in S$, so that $\tilde{S} = S - \{1\}$.
  - Then $y(E) = f(E)$ implies that $y_1 = f(E) - \tilde{y}(\tilde{E})$.
  - Thus $y(S) = \tilde{y}(\tilde{S}) + y_1 = \tilde{y}(\tilde{S}) + f(E) - \tilde{y}(\tilde{E}) = f(E) - \tilde{y}(\tilde{E} - \tilde{S}) \le f(E) - \tilde{x}(\tilde{E} - \tilde{S}) = \tilde{x}(\tilde{S}) + f(E) - \tilde{x}(\tilde{\tilde{E}}) = x(\tilde{S}) + x_1 = x(S) \le f(S)$.
- These show that $y \in B(f)$ iff $y(E) = f(E)$ and
  - $\tilde{y}(S) \le \tilde{f}(S) \ \forall\ S \subseteq \tilde{E}$, and
  - $\tilde{y}(T) \ge \tilde{f}^{\#}(T) \equiv f(E) - f(T) \ \forall\ T$ s.t. $T \subseteq \tilde{E}$ (with $S \equiv E - T$, so that $1 \in S$).
- I.e., iff $y(E) = f(E)$ and

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \leq f(S)$.
- Case 1: Suppose that $1 \notin S$, so that $\tilde{S} = S$.
  - Then $y(S) = y(\tilde{S}) \leq z(\tilde{S}) = z(S) \leq f(S)$.
- Case 2: Suppose that $1 \in S$, so that $\tilde{S} = S - \{1\}$.
  - Then $y(E) = f(E)$ implies that $y_1 = f(E) - \tilde{y}(\tilde{E})$.
  - Thus $y(S) = \tilde{y}(\tilde{S}) + y_1 = \tilde{y}(\tilde{S}) + f(E) - \tilde{y}(\tilde{E}) = f(E) - \tilde{y}(\tilde{E} - \tilde{S}) \leq f(E) - \tilde{x}(\tilde{E} - \tilde{S}) = \tilde{x}(\tilde{S}) + f(E) - \tilde{x}(\tilde{E}) = x(\tilde{S}) + x_1 = x(S) \leq f(S)$.
- These show that $y \in B(f)$ iff $y(E) = f(E)$ and
  - $\tilde{y}(S) \leq \tilde{f}(S)\ \forall\ S \subseteq \tilde{E}$, and
  - $\tilde{y}(T) \geq \tilde{f}^{\#}(T) \equiv f(E) - f(T)\ \forall\ T$ s.t. $T \subseteq \tilde{E}$ (with $S \equiv E - T$, so that $1 \in S$).
- I.e., iff $y(E) = f(E)$ and
  - $\tilde{y} \in P(\tilde{f})$ (submodular polyhedron) and

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \leq f(S)$.
- Case 1: Suppose that $1 \notin S$, so that $\tilde{S} = S$.
  - Then $y(S) = y(\tilde{S}) \leq z(\tilde{S}) = z(S) \leq f(S)$.
- Case 2: Suppose that $1 \in S$, so that $\tilde{S} = S - \{1\}$.
  - Then $y(E) = f(E)$ implies that $y_1 = f(E) - \tilde{y}(\tilde{E})$.
  - Thus $y(S) = \tilde{y}(\tilde{S}) + y_1 = \tilde{y}(\tilde{S}) + f(E) - \tilde{y}(\tilde{E}) =$ $f(E) - \tilde{y}(\tilde{E} - \tilde{S}) \leq f(E) - \tilde{x}(\tilde{E} - \tilde{S}) =$ $\tilde{x}(\tilde{S}) + f(E) - \tilde{x}(\tilde{\tilde{E}}) = x(\tilde{S}) + x_1 = x(S) \leq f(S)$.
- These show that $y \in B(f)$ iff $y(E) = f(E)$ and
  - $\tilde{y}(S) \leq \tilde{f}(S)\ \forall\ S \subseteq \tilde{E}$, and
  - $\tilde{y}(T) \geq \tilde{f}^{\#}(T) \equiv f(E) - f(T)\ \forall\ T$ s.t. $T \subseteq \tilde{E}$ (with $S \equiv E - T$, so that $1 \in S$).
- I.e., iff $y(E) = f(E)$ and
  - $\tilde{y} \in P(\tilde{f})$ (submodular polyhedron) and
  - $\tilde{y} \in P^{\#}(\tilde{f}^{\#})$ (supermodular polyhedron).

Proof of the theorem

- We need to show that for all $S \subseteq E$ that $y(S) \leq f(S)$.
- Case 1: Suppose that $1 \notin S$, so that $\tilde{S} = S$.
  - Then $y(S) = y(\tilde{S}) \leq z(\tilde{S}) = z(S) \leq f(S)$.
- Case 2: Suppose that $1 \in S$, so that $\tilde{S} = S - \{1\}$.
  - Then $y(E) = f(E)$ implies that $y_1 = f(E) - \tilde{y}(\tilde{E})$.
  - Thus $y(S) = \tilde{y}(\tilde{S}) + y_1 = \tilde{y}(\tilde{S}) + f(E) - \tilde{y}(\tilde{E}) =$ $f(E) - \tilde{y}(\tilde{E} - \tilde{S}) \leq f(E) - \tilde{x}(\tilde{E} - \tilde{S}) =$ $\tilde{x}(\tilde{S}) + f(E) - \tilde{x}(\tilde{E}) = x(\tilde{S}) + x_1 = x(S) \leq f(S)$.
- These show that $y \in B(f)$ iff $y(E) = f(E)$ and
  - $\tilde{y}(S) \leq \tilde{f}(S)\ \forall\ S \subseteq \tilde{E}$, and
  - $\tilde{y}(T) \geq \tilde{f}^{\#}(T) \equiv f(E) - f(T)\ \forall\ T$ s.t. $T \subseteq \tilde{E}$ (with $S \equiv E - T$, so that $1 \in S$).
- I.e., iff $y(E) = f(E)$ and
  - $\tilde{y} \in P(\tilde{f})$ (submodular polyhedron) and
  - $\tilde{y} \in P^{\#}(\tilde{f}^{\#})$ (supermodular polyhedron).
- This projection of $B(f)$ along one component is a g-polymatroid (and all g-polymatroids arise this way).

Implications of the theorem

- Suppose, e.g., that x and z are vertices of $B(f)$ coming from linear orders $\prec_x$ and $\prec_z$.

Implications of the theorem

- Suppose, e.g., that x and z are vertices of $B(f)$ coming from linear orders $\prec_x$ and $\prec_z$.
  - Thus $\prec_x$ and $\prec_z$ certify that x, $z \in B(f)$ via Greedy.

Implications of the theorem

- Suppose, e.g., that x and z are vertices of $B(f)$ coming from linear orders $\prec_x$ and $\prec_z$.
  - Thus $\prec_x$ and $\prec_z$ certify that x, $z \in B(f)$ via Greedy.
- Then $y(E) = f(E)$ and $\tilde{x} \leq \tilde{y} \leq \tilde{z}$ certify that $y \in B(f)$ using no linear algebra.

Implications of the theorem

- Suppose, e.g., that x and z are vertices of $B(f)$ coming from linear orders $\prec_x$ and $\prec_z$.
  - Thus $\prec_x$ and $\prec_z$ certify that x, $z \in B(f)$ via Greedy.
- Then $y(E) = f(E)$ and $\tilde{x} \leq \tilde{y} \leq \tilde{z}$ certify that $y \in B(f)$ using no linear algebra.
- Do these "projected boxes" starting from vertices cover $B(f)$?

Implications of the theorem

- Suppose, e.g., that x and z are vertices of $B(f)$ coming from linear orders $\prec_x$ and $\prec_z$.
  - Thus $\prec_x$ and $\prec_z$ certify that x, $z \in B(f)$ via Greedy.
- Then $y(E) = f(E)$ and $\tilde{x} \leq \tilde{y} \leq \tilde{z}$ certify that $y \in B(f)$ using no linear algebra.
- Do these "projected boxes" starting from vertices cover $B(f)$?
- YES; Proof:

Implications of the theorem

- Suppose, e.g., that x and z are vertices of $B(f)$ coming from linear orders $\prec_x$ and $\prec_z$.
  - Thus $\prec_x$ and $\prec_z$ certify that x, $z \in B(f)$ via Greedy.
- Then $y(E) = f(E)$ and $\tilde{x} \le \tilde{y} \le \tilde{z}$ certify that $y \in B(f)$ using no linear algebra.
- Do these "projected boxes" starting from vertices cover $B(f)$?
- YES; Proof:
  - Suppose that $y \in B(f)$.

Implications of the theorem

- Suppose, e.g., that x and z are vertices of $B(f)$ coming from linear orders $\prec_x$ and $\prec_z$.
  - Thus $\prec_x$ and $\prec_z$ certify that x, $z \in B(f)$ via Greedy.
- Then $y(E) = f(E)$ and $\tilde{x} \leq \tilde{y} \leq \tilde{z}$ certify that $y \in B(f)$ using no linear algebra.
- Do these "projected boxes" starting from vertices cover $B(f)$?
- YES; Proof:
  - Suppose that $y \in B(f)$.
  - This implies that $\tilde{y} \in P(\tilde{f})$.

Implications of the theorem

- Suppose, e.g., that x and z are vertices of $B(f)$ coming from linear orders $\prec_x$ and $\prec_z$.
  - Thus $\prec_x$ and $\prec_z$ certify that x, $z \in B(f)$ via Greedy.
- Then $y(E) = f(E)$ and $\tilde{x} \leq \tilde{y} \leq \tilde{z}$ certify that $y \in B(f)$ using no linear algebra.
- Do these "projected boxes" starting from vertices cover $B(f)$?
- YES; Proof:
  - Suppose that $y \in B(f)$.
  - This implies that $\tilde{y} \in P(\tilde{f})$.
    - We already saw that this means that we can increase components of $\tilde{y}$ to get $\tilde{z}$ with $\tilde{y} \leq \tilde{z}$ and $\tilde{z} \in B(\tilde{f})$.

Implications of the theorem

- Suppose, e.g., that x and z are vertices of $B(f)$ coming from linear orders $\prec_x$ and $\prec_z$.
  - Thus $\prec_x$ and $\prec_z$ certify that x, $z \in B(f)$ via Greedy.
- Then $y(E) = f(E)$ and $\tilde{x} \le \tilde{y} \le \tilde{z}$ certify that $y \in B(f)$ using no linear algebra.
- Do these "projected boxes" starting from vertices cover $B(f)$?
- YES; Proof:
  - Suppose that $y \in B(f)$.
  - This implies that $\tilde{y} \in P(\tilde{f})$.
    - We already saw that this means that we can increase components of $\tilde{y}$ to get $\tilde{z}$ with $\tilde{y} \le \tilde{z}$ and $\tilde{z} \in B(\tilde{f})$.
  - This also implies that $\tilde{y} \in P^{\#}(\tilde{f})$.

Implications of the theorem

- Suppose, e.g., that x and z are vertices of $B(f)$ coming from linear orders $\prec_x$ and $\prec_z$.
  - Thus $\prec_x$ and $\prec_z$ certify that x, $z \in B(f)$ via Greedy.
- Then $y(E) = f(E)$ and $\tilde{x} \leq \tilde{y} \leq \tilde{z}$ certify that $y \in B(f)$ using no linear algebra.
- Do these "projected boxes" starting from vertices cover $B(f)$?
- YES; Proof:
  - Suppose that $y \in B(f)$.
  - This implies that $\tilde{y} \in P(\tilde{f})$.
    - We already saw that this means that we can increase components of $\tilde{y}$ to get $\tilde{z}$ with $\tilde{y} \leq \tilde{z}$ and $\tilde{z} \in B(\tilde{f})$.
  - This also implies that $\tilde{y} \in P^{\#}(\tilde{f})$.
    - Similar proof show that this means that we can decrease components of $\tilde{y}$ to get $\tilde{x}$ with $\tilde{x} \leq \tilde{y}$ and $\tilde{x} \in B^{\#}(\tilde{f})$.

Implications of the theorem

- Suppose, e.g., that x and z are vertices of $B(f)$ coming from linear orders $\prec_x$ and $\prec_z$.
  - Thus $\prec_x$ and $\prec_z$ certify that x, $z \in B(f)$ via Greedy.
- Then $y(E) = f(E)$ and $\tilde{x} \leq \tilde{y} \leq \tilde{z}$ certify that $y \in B(f)$ using no linear algebra.
- Do these "projected boxes" starting from vertices cover $B(f)$?
- YES; Proof:
  - Suppose that $y \in B(f)$.
  - This implies that $\tilde{y} \in P(\tilde{f})$.
    - We already saw that this means that we can increase components of $\tilde{y}$ to get $\tilde{z}$ with $\tilde{y} \leq \tilde{z}$ and $\tilde{z} \in B(\tilde{f})$.
  - This also implies that $\tilde{y} \in P^{\#}(\tilde{f})$.
    - Similar proof show that this means that we can decrease components of $\tilde{y}$ to get $\tilde{x}$ with $\tilde{x} \leq \tilde{y}$ and $\tilde{x} \in B^{\#}(\tilde{f})$.
  - Now apply induction: $\tilde{z} \in B(\tilde{f})$ (one dimension less) implies that we express $\tilde{z}$ as a combinatorial hull from vertices.

Implications of the theorem

- Suppose, e.g., that x and z are vertices of $B(f)$ coming from linear orders $\prec_x$ and $\prec_z$.
  - Thus $\prec_x$ and $\prec_z$ certify that x, $z \in B(f)$ via Greedy.
- Then $y(E) = f(E)$ and $\tilde{x} \leq \tilde{y} \leq \tilde{z}$ certify that $y \in B(f)$ using no linear algebra.
- Do these "projected boxes" starting from vertices cover $B(f)$?
- YES; Proof:
  - Suppose that $y \in B(f)$.
  - This implies that $\tilde{y} \in P(\tilde{f})$.
    - We already saw that this means that we can increase components of $\tilde{y}$ to get $\tilde{z}$ with $\tilde{y} \leq \tilde{z}$ and $\tilde{z} \in B(\tilde{f})$.
  - This also implies that $\tilde{y} \in P^{\#}(\tilde{f})$.
    - Similar proof show that this means that we can decrease components of $\tilde{y}$ to get $\tilde{x}$ with $\tilde{x} \leq \tilde{y}$ and $\tilde{x} \in B^{\#}(\tilde{f})$.
  - Now apply induction: $\tilde{z} \in B(\tilde{f})$ (one dimension less) implies that we express $\tilde{z}$ as a combinatorial hull from vertices.
  - ...and apply induction to $\tilde{x} \in B^{\#}(\tilde{f})$ to get $\tilde{x}$ as a combinatorial hull from vertices.

Good news, bad news

- Good news: This proof shows that we can use this combinatorial hull operation to prove that $y \in B(f)$ without using linear algebra.

Good news, bad news

- Good news: This proof shows that we can use this combinatorial hull operation to prove that $y \in B(f)$ without using linear algebra.
- The "depth" of this representation is only $O(n)$.

Good news, bad news

- Good news: This proof shows that we can use this combinatorial hull operation to prove that $y \in B(f)$ without using linear algebra.
- The "depth" of this representation is only $O(n)$.
- Bad news: The size of this representation is potentially 2^n:

Good news, bad news

- Good news: This proof shows that we can use this combinatorial hull operation to prove that $y \in B(f)$ without using linear algebra.
- The "depth" of this representation is only $O(n)$.
- Bad news: The size of this representation is potentially 2^n:
  - Each new dimension doubles the number of points we are using.

Good news, bad news

- Good news: This proof shows that we can use this combinatorial hull operation to prove that $y \in B(f)$ without using linear algebra.
- The "depth" of this representation is only $O(n)$.
- Bad news: The size of this representation is potentially 2^n:
  - Each new dimension doubles the number of points we are using.
- This achieves only half of what we challenged ourselves to do:

Good news, bad news

- Good news: This proof shows that we can use this combinatorial hull operation to prove that $y \in B(f)$ without using linear algebra.
- The "depth" of this representation is only $O(n)$.
- Bad news: The size of this representation is potentially 2^n:
  - Each new dimension doubles the number of points we are using.
- This achieves only half of what we challenged ourselves to do:
  - It does avoid linear algebra, but

Good news, bad news

- Good news: This proof shows that we can use this combinatorial hull operation to prove that $y \in B(f)$ without using linear algebra.
- The "depth" of this representation is only $O(n)$.
- Bad news: The size of this representation is potentially 2^n:
  - Each new dimension doubles the number of points we are using.
- This achieves only half of what we challenged ourselves to do:
  - It does avoid linear algebra, but
  - It does not appear to be efficient (so far). That is, we don't have a combinatorial hull equivalent to Carathéodory's Theorem.

Outline

What is "combinatorial hull"?

- The word "hull" typically means a closure operation.

What is “combinatorial hull”?

- The word “hull” typically means a closure operation.
- Here is the closure operation we want: Given a set of points $P \subseteq \mathbb{R}^n$ satisfying $x(E) = c$ for all $x \in P$, we say that y with $y(E) = c$ is in the combinatorial hull of P if there exists some i with $1 \leq i \leq n$ and $x, z \in P$ such that $\tilde{x} \leq \tilde{y} \leq \tilde{z}$ (where the tildes are w.r.t. projecting out component i).

What is "combinatorial hull"?

- The word "hull" typically means a closure operation.
- Here is the closure operation we want: Given a set of points $P \subseteq \mathbb{R}^n$ satisfying $x(E) = c$ for all $x \in P$, we say that y with $y(E) = c$ is in the combinatorial hull of P if there exists some i with $1 \leq i \leq n$ and $x, z \in P$ such that $\tilde{x} \leq \tilde{y} \leq \tilde{z}$ (where the tildes are w.r.t. projecting out component i).
- Then the combinatorial hull of P, $\mathrm{combhull}(P)$, is the set of points we obtain by iterating this operation.

What is "combinatorial hull"?

- The word "hull" typically means a closure operation.
- Here is the closure operation we want: Given a set of points $P \subseteq \mathbb{R}^n$ satisfying $x(E) = c$ for all $x \in P$, we say that y with $y(E) = c$ is in the combinatorial hull of P if there exists some i with $1 \leq i \leq n$ and $x, z \in P$ such that $\tilde{x} \leq \tilde{y} \leq \tilde{z}$ (where the tildes are w.r.t. projecting out component i).
- Then the combinatorial hull of P, $\mathrm{combhull}(P)$, is the set of points we obtain by iterating this operation.
- In these terms we have shown that if $V(f)$ is the set of vertices of $B(f)$, then $\mathrm{combhull}(V(f)) = B(f)$, and $B(f) = \mathrm{combhull}(B(f))$.

What is "combinatorial hull"?

- The word "hull" typically means a closure operation.
- Here is the closure operation we want: Given a set of points $P \subseteq \mathbb{R}^n$ satisfying $x(E) = c$ for all $x \in P$, we say that y with $y(E) = c$ is in the combinatorial hull of P if there exists some i with $1 \leq i \leq n$ and $x, z \in P$ such that $\tilde{x} \leq \tilde{y} \leq \tilde{z}$ (where the tildes are w.r.t. projecting out component i).
- Then the combinatorial hull of P, $\mathrm{combhull}(P)$, is the set of points we obtain by iterating this operation.
- In these terms we have shown that if $V(f)$ is the set of vertices of $B(f)$, then $\mathrm{combhull}(V(f)) = B(f)$, and $B(f) = \mathrm{combhull}(B(f))$.
- What we have not shown is, starting from $V(f)$, how many iterations of the combinatorial hull operation are necessary to get to an arbitrary point of $B(f)$.

A glimmer of hope

- When $n = 2$, $B(f)$ is one-dimensional with at most two vertices, and it is easy to see that any $y \in B(f)$ is in the combinatorial hull of the two vertices.

A glimmer of hope

- When $n = 2$, $B(f)$ is one-dimensional with at most two vertices, and it is easy to see that any $y \in B(f)$ is in the combinatorial hull of the two vertices.
- When $n = 3$, $B(f)$ is two-dimensional.

A glimmer of hope

- When $n = 2$, $B(f)$ is one-dimensional with at most two vertices, and it is easy to see that any $y \in B(f)$ is in the combinatorial hull of the two vertices.
- When $n = 3$, $B(f)$ is two-dimensional.
  - Our naive proof would give a combinatorial hull representation using four vertices.

A glimmer of hope

- When $n = 2$, $B(f)$ is one-dimensional with at most two vertices, and it is easy to see that any $y \in B(f)$ is in the combinatorial hull of the two vertices.
- When $n = 3$, $B(f)$ is two-dimensional.
  - Our naive proof would give a combinatorial hull representation using four vertices.
  - But a brute force proof shows that we can always get a combinatorial hull representation using at most three vertices.

A glimmer of hope

- When $n = 2$, $B(f)$ is one-dimensional with at most two vertices, and it is easy to see that any $y \in B(f)$ is in the combinatorial hull of the two vertices.
- When $n = 3$, $B(f)$ is two-dimensional.
  - Our naive proof would give a combinatorial hull representation using four vertices.
  - But a brute force proof shows that we can always get a combinatorial hull representation using at most three vertices.
- So maybe this naive proof is not being clever enough, and maybe we could get a more clever proof that (efficiently, algorithmically) produces a representation with only a polynomial number of vertices?

A glimmer of hope

- When $n = 2$, $B(f)$ is one-dimensional with at most two vertices, and it is easy to see that any $y \in B(f)$ is in the combinatorial hull of the two vertices.
- When $n = 3$, $B(f)$ is two-dimensional.
  - Our naive proof would give a combinatorial hull representation using four vertices.
  - But a brute force proof shows that we can always get a combinatorial hull representation using at most three vertices.
- So maybe this naive proof is not being clever enough, and maybe we could get a more clever proof that (efficiently, algorithmically) produces a representation with only a polynomial number of vertices?
- Spoiler alert: I'm not going to give you such an algorithm.

A glimmer of hope

- When $n = 2$, $B(f)$ is one-dimensional with at most two vertices, and it is easy to see that any $y \in B(f)$ is in the combinatorial hull of the two vertices.
- When $n = 3$, $B(f)$ is two-dimensional.
  - Our naive proof would give a combinatorial hull representation using four vertices.
  - But a brute force proof shows that we can always get a combinatorial hull representation using at most three vertices.
- So maybe this naive proof is not being clever enough, and maybe we could get a more clever proof that (efficiently, algorithmically) produces a representation with only a polynomial number of vertices?
- Spoiler alert: I'm not going to give you such an algorithm.
- Hopefulness: But I will give you some tools you might use to construct such an algorithm.

Combinatorial hull and SFMin

- Imagine the last step of some SFMin algorithm where we are verifying that our current y is optimal.

Combinatorial hull and SFMin

- Imagine the last step of some SFMin algorithm where we are verifying that our current y is optimal.
  - We are using the old-style representation $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$, which proves that $y \in B(f)$.

Combinatorial hull and SFMin

- Imagine the last step of some SFMin algorithm where we are verifying that our current y is optimal.
  - We are using the old-style representation $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$, which proves that $y \in B(f)$.
  - But to prove optimality, we need more: we also have the subset S of elements reachable from S^-, and we want to verify that $y(S) = f(S)$ (part of complementary slackness).

Combinatorial hull and SFMin

- Imagine the last step of some SFMin algorithm where we are verifying that our current y is optimal.
  - We are using the old-style representation $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$, which proves that $y \in B(f)$.
  - But to prove optimality, we need more: we also have the subset S of elements reachable from S^-, and we want to verify that $y(S) = f(S)$ (part of complementary slackness).
  - "Reachable" implies that S is consecutive at the beginning of every $\prec_i$.

Combinatorial hull and SFMin

- Imagine the last step of some SFMin algorithm where we are verifying that our current y is optimal.
  - We are using the old-style representation $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$, which proves that $y \in B(f)$.
  - But to prove optimality, we need more: we also have the subset S of elements reachable from S^-, and we want to verify that $y(S) = f(S)$ (part of complementary slackness).
  - "Reachable" implies that S is consecutive at the beginning of every $\prec_i$.
  - Then Greedy implies that $v^i(S) = f(S)$ for all $i \in \mathcal{I}$.

Combinatorial hull and SFMin

- Imagine the last step of some SFMin algorithm where we are verifying that our current y is optimal.
  - We are using the old-style representation $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$, which proves that $y \in B(f)$.
  - But to prove optimality, we need more: we also have the subset S of elements reachable from S^-, and we want to verify that $y(S) = f(S)$ (part of complementary slackness).
  - "Reachable" implies that S is consecutive at the beginning of every $\prec_i$.
  - Then Greedy implies that $v^i(S) = f(S)$ for all $i \in \mathcal{I}$.
  - Then linear algebra says that $y(S) = \sum_{i \in \mathcal{I}} \lambda_i v^i(S) = \sum_{i \in \mathcal{I}} \lambda_i f(S) = f(S) \sum_{i \in \mathcal{I}} \lambda_i = f(S)$, and so S is indeed y-tight.

Combinatorial hull and SFMin

- Imagine the last step of some SFMin algorithm where we are verifying that our current y is optimal.
  - We are using the old-style representation $y = \sum_{i \in \mathcal{I}} \lambda_i v^i$, which proves that $y \in B(f)$.
  - But to prove optimality, we need more: we also have the subset S of elements reachable from S^-, and we want to verify that $y(S) = f(S)$ (part of complementary slackness).
  - "Reachable" implies that S is consecutive at the beginning of every $\prec_i$.
  - Then Greedy implies that $v^i(S) = f(S)$ for all $i \in \mathcal{I}$.
  - Then linear algebra says that $y(S) = \sum_{i \in \mathcal{I}} \lambda_i v^i(S) = \sum_{i \in \mathcal{I}} \lambda_i f(S) = f(S) \sum_{i \in \mathcal{I}} \lambda_i = f(S)$, and so S is indeed y-tight.
- Can we also do this for combinatorial hull?

Tightness and combinatorial hull

- Easy direction: Proof that S tight for x and z implies that S is also tight for y.

Tightness and combinatorial hull

- Easy direction: Proof that S tight for x and z implies that S is also tight for y.
  - If $1 \notin S$: $f(S) = x(S) = \tilde{x}(\tilde{S}) \leq \tilde{y}(\tilde{S}) \leq \tilde{z}(\tilde{S}) = z(S) = f(S)$, so we get equality everywhere, and so $\tilde{y}(\tilde{S}) = y(S) = f(S)$.

Tightness and combinatorial hull

- Easy direction: Proof that S tight for x and z implies that S is also tight for y.
  - If $1 \notin S$: $f(S) = x(S) = \tilde{x}(\tilde{S}) \leq \tilde{y}(\tilde{S}) \leq \tilde{z}(\tilde{S}) = z(S) = f(S)$, so we get equality everywhere, and so $\tilde{y}(\tilde{S}) = y(S) = f(S)$.
  - If $1 \in S$: (same proof with everything complemented).

Tightness and combinatorial hull

- Easy direction: Proof that S tight for x and z implies that S is also tight for y.
  - If $1 \notin S$: $f(S) = x(S) = \tilde{x}(\tilde{S}) \le \tilde{y}(\tilde{S}) \le \tilde{z}(\tilde{S}) = z(S) = f(S)$, so we get equality everywhere, and so $\tilde{y}(\tilde{S}) = y(S) = f(S)$.
  - If $1 \in S$: (same proof with everything complemented).
  - This is the direction that we need for SFMin optimality.

Tightness and combinatorial hull

- Easy direction: Proof that S tight for x and z implies that S is also tight for y.
  - If $1 \notin S$: $f(S) = x(S) = \tilde{x}(\tilde{S}) \leq \tilde{y}(\tilde{S}) \leq \tilde{z}(\tilde{S}) = z(S) = f(S)$, so we get equality everywhere, and so $\tilde{y}(\tilde{S}) = y(S) = f(S)$.
  - If $1 \in S$: (same proof with everything complemented).
  - This is the direction that we need for SFMin optimality.
- Slightly harder direction: If S is tight for y, is it necessarily also tight for x and z?

Tightness and combinatorial hull

- Easy direction: Proof that S tight for x and z implies that S is also tight for y.
  - If $1 \notin S$: $f(S) = x(S) = \tilde{x}(\tilde{S}) \le \tilde{y}(\tilde{S}) \le \tilde{z}(\tilde{S}) = z(S) = f(S)$, so we get equality everywhere, and so $\tilde{y}(\tilde{S}) = y(S) = f(S)$.
  - If $1 \in S$: (same proof with everything complemented).
  - This is the direction that we need for SFMin optimality.
- Slightly harder direction: If S is tight for y, is it necessarily also tight for x and z?
  - NO! A simple counterexample shows that we can have, e.g., S tight for y and z but not tight for x.

Tightness and combinatorial hull

- Easy direction: Proof that S tight for x and z implies that S is also tight for y.
  - If $1 \notin S$: $f(S) = x(S) = \tilde{x}(\tilde{S}) \leq \tilde{y}(\tilde{S}) \leq \tilde{z}(\tilde{S}) = z(S) = f(S)$, so we get equality everywhere, and so $\tilde{y}(\tilde{S}) = y(S) = f(S)$.
  - If $1 \in S$: (same proof with everything complemented).
  - This is the direction that we need for SFMin optimality.
- Slightly harder direction: If S is tight for y, is it necessarily also tight for x and z?
  - NO! A simple counterexample shows that we can have, e.g., S tight for y and z but not tight for x.
  - But if we have that $\tilde{x}_e < \tilde{y}_e < \tilde{z}_e$ for all $e \in \tilde{E}$ with $x_e < z_e$ (i.e., if y is strictly interior wherever possible), then it's fairly easy to show that S tight for y implies that it is also tight for x and z.

Membership and combinatorial hull

- Given a polyhedron P and point $y \in \mathbb{R}^n$, the membership problem is to either

Membership and combinatorial hull

- Given a polyhedron P and point $y \in \mathbb{R}^n$, the membership problem is to either
  1. Prove that $y \in P$, or

Membership and combinatorial hull

- Given a polyhedron P and point $y \in \mathbb{R}^n$, the membership problem is to either
  1. Prove that $y \in P$, or
  2. Find a valid inequality (facet?) $\alpha^T x \leq b$ for P violated by y, i.e., $\alpha^T y > b$.

Membership and combinatorial hull

- Given a polyhedron P and point $y \in \mathbb{R}^n$, the membership problem is to either
  1. Prove that $y \in P$, or
  2. Find a valid inequality (facet?) $\alpha^T x \leq b$ for P violated by y, i.e., $\alpha^T y > b$.
- Given an instance $B(f)$, y of membership, we are hoping for a polynomial algorithm that proves either that $y \in B(f)$ (via constructing a combinatorial hull representation), or some S such that $y(S) > f(S)$.

Membership and combinatorial hull

- Given a polyhedron P and point $y \in \mathbb{R}^n$, the membership problem is to either
  1. Prove that $y \in P$, or
  2. Find a valid inequality (facet?) $\alpha^T x \le b$ for P violated by y, i.e., $\alpha^T y > b$.
- Given an instance $B(f)$, y of membership, we are hoping for a polynomial algorithm that proves either that $y \in B(f)$ (via constructing a combinatorial hull representation), or some S such that $y(S) > f(S)$.
- It is easy to reduce general membership for general $B(f)$ and y to membership for a related submodular $B(\hat{f})$ and 0, where $\hat{f}(E) = \hat{f}(\emptyset) = 0$

Membership and combinatorial hull

- Given a polyhedron P and point $y \in \mathbb{R}^n$, the membership problem is to either
  1. Prove that $y \in P$, or
  2. Find a valid inequality (facet?) $\alpha^T x \leq b$ for P violated by y, i.e., $\alpha^T y > b$.
- Given an instance $B(f)$, y of membership, we are hoping for a polynomial algorithm that proves either that $y \in B(f)$ (via constructing a combinatorial hull representation), or some S such that $y(S) > f(S)$.
- It is easy to reduce general membership for general $B(f)$ and y to membership for a related submodular $B(\hat{f})$ and 0, where $\hat{f}(E) = \hat{f}(\emptyset) = 0$
  - Define $\hat{f}(S) = f(S) - y(S)$.

Membership and combinatorial hull

- Given a polyhedron P and point $y \in \mathbb{R}^n$, the membership problem is to either
  1. Prove that $y \in P$, or
  2. Find a valid inequality (facet?) $\alpha^T x \leq b$ for P violated by y, i.e., $\alpha^T y > b$.
- Given an instance $B(f)$, y of membership, we are hoping for a polynomial algorithm that proves either that $y \in B(f)$ (via constructing a combinatorial hull representation), or some S such that $y(S) > f(S)$.
- It is easy to reduce general membership for general $B(f)$ and y to membership for a related submodular $B(\hat{f})$ and 0, where $\hat{f}(E) = \hat{f}(\emptyset) = 0$
  - Define $\hat{f}(S) = f(S) - y(S)$.
  - Clearly $\hat{f}(E) = \hat{f}(\emptyset) = 0$.

Membership and combinatorial hull

- Given a polyhedron P and point $y \in \mathbb{R}^n$, the membership problem is to either
  1. Prove that $y \in P$, or
  2. Find a valid inequality (facet?) $\alpha^T x \le b$ for P violated by y, i.e., $\alpha^T y > b$.
- Given an instance $B(f)$, y of membership, we are hoping for a polynomial algorithm that proves either that $y \in B(f)$ (via constructing a combinatorial hull representation), or some S such that $y(S) > f(S)$.
- It is easy to reduce general membership for general $B(f)$ and y to membership for a related submodular $B(\hat{f})$ and 0, where $\hat{f}(E) = \hat{f}(\emptyset) = 0$
  - Define $\hat{f}(S) = f(S) - y(S)$.
  - Clearly $\hat{f}(E) = \hat{f}(\emptyset) = 0$.
  - Since $y(S)$ is modular, $\hat{f}(S)$ is submodular.

Membership and combinatorial hull

- Given a polyhedron P and point $y \in \mathbb{R}^n$, the membership problem is to either
  1. Prove that $y \in P$, or
  2. Find a valid inequality (facet?) $\alpha^T x \leq b$ for P violated by y, i.e., $\alpha^T y > b$.
- Given an instance $B(f)$, y of membership, we are hoping for a polynomial algorithm that proves either that $y \in B(f)$ (via constructing a combinatorial hull representation), or some S such that $y(S) > f(S)$.
- It is easy to reduce general membership for general $B(f)$ and y to membership for a related submodular $B(\hat{f})$ and 0, where $\hat{f}(E) = \hat{f}(\emptyset) = 0$
  - Define $\hat{f}(S) = f(S) - y(S)$.
  - Clearly $\hat{f}(E) = \hat{f}(\emptyset) = 0$.
  - Since $y(S)$ is modular, $\hat{f}(S)$ is submodular.
  - Now S proves that $y \notin B(f)$ iff $y(S) > f(S)$ iff $0 > f(S) - y(S) = \hat{f}(S)$, and so $y \in B(f)$ iff $0 \in B(\hat{f})$.

Membership and SFMin

- Even better, if we can solve this membership problem w.r.t. $B(\hat{f})$ and 0, then Fujishige and Iwata (2001) show that $O(n^2)$ calls to this subroutine suffice to solve SFMin.

Membership and SFMin

- Even better, if we can solve this membership problem w.r.t. $B(\hat{f})$ and 0, then Fujishige and Iwata (2001) show that $O(n^2)$ calls to this subroutine suffice to solve SFMin.
  - If we had a membership subroutine that didn't use linear algebra, then we'd have a linear algebra-free SFMin algorithm.

Membership and SFMin

- Even better, if we can solve this membership problem w.r.t. $B(\hat{f})$ and 0, then Fujishige and Iwata (2001) show that $O(n^2)$ calls to this subroutine suffice to solve SFMin.
  - If we had a membership subroutine that didn't use linear algebra, then we'd have a linear algebra-free SFMin algorithm.
- Thus a polynomial combinatorial algorithm that uses combinatorial hull to solve membership of 0 in $B(\hat{f})$ is a worthwhile target.

Membership and SFMin

- Even better, if we can solve this membership problem w.r.t. $B(\hat{f})$ and 0, then Fujishige and Iwata (2001) show that $O(n^2)$ calls to this subroutine suffice to solve SFMin.
  - If we had a membership subroutine that didn't use linear algebra, then we'd have a linear algebra-free SFMin algorithm.
- Thus a polynomial combinatorial algorithm that uses combinatorial hull to solve membership of 0 in $B(\hat{f})$ is a worthwhile target.
- With this reduction to membership for 0, what we are trying to do is to construct points x, $z \in B(\hat{f})$ via combinatorial hull such that $\tilde{x} \leq 0$ and $\tilde{z} \geq 0$.

Membership and SFMin

- Even better, if we can solve this membership problem w.r.t. $B(\hat{f})$ and 0, then Fujishige and Iwata (2001) show that $O(n^2)$ calls to this subroutine suffice to solve SFMin.
  - If we had a membership subroutine that didn't use linear algebra, then we'd have a linear algebra-free SFMin algorithm.
- Thus a polynomial combinatorial algorithm that uses combinatorial hull to solve membership of 0 in $B(\hat{f})$ is a worthwhile target.
- With this reduction to membership for 0, what we are trying to do is to construct points x, $z \in B(\hat{f})$ via combinatorial hull such that $\tilde{x} \leq 0$ and $\tilde{z} \geq 0$.
- The problem is symmetric between x and z: If we can succeed in constructing a point $z \in B(\hat{f})$ with $\tilde{z} \geq 0$ (or prove that no such z exists), then we could run the same algorithm with signs reversed to get some $x \in B(\hat{f})$ with $\tilde{x} \leq 0$ (or prove that no such x exists).

Some useful facts about vertices of $B(f)$

- We would start an algorithm with a Greedy vertex v coming from linear order $\prec$.

Some useful facts about vertices of $B(f)$

- We would start an algorithm with a Greedy vertex v coming from linear order $\prec$.
- Recall that Greedy has the property that if S is an *initial* subset of $\prec$, then $v(S) = f(S)$.

Some useful facts about vertices of $B(f)$

- We would start an algorithm with a Greedy vertex v coming from linear order $\prec$.
- Recall that Greedy has the property that if S is an *initial* subset of $\prec$, then $v(S) = f(S)$.
  - Since $y(S) \leq f(S)$ for all $y \in B(f)$, clearly v solves $\max_{y \in B(f)} y(S)$.

Some useful facts about vertices of $B(f)$

- We would start an algorithm with a Greedy vertex v coming from linear order $\prec$.
- Recall that Greedy has the property that if S is an *initial* subset of $\prec$, then $v(S) = f(S)$.
  - Since $y(S) \leq f(S)$ for all $y \in B(f)$, clearly v solves $\max_{y \in B(f)} y(S)$.
- Slightly trickier: Suppose now that S is a *terminal* subset of $\prec$.

Some useful facts about vertices of $B(f)$

- We would start an algorithm with a Greedy vertex v coming from linear order $\prec$.
- Recall that Greedy has the property that if S is an *initial* subset of $\prec$, then $v(S) = f(S)$.
  - Since $y(S) \leq f(S)$ for all $y \in B(f)$, clearly v solves $\max_{y \in B(f)} y(S)$.
- Slightly trickier: Suppose now that S is a *terminal* subset of $\prec$.
  - Since v solves $\max_{y \in B(f)} y(E - S)$, and since $y(E)$ is constant on $B(f)$, we have that v must solve $\min_{y \in B(f)} y(S)$.

Some useful facts about vertices of $B(f)$

- We would start an algorithm with a Greedy vertex v coming from linear order $\prec$.
- Recall that Greedy has the property that if S is an *initial* subset of $\prec$, then $v(S) = f(S)$.
  - Since $y(S) \leq f(S)$ for all $y \in B(f)$, clearly v solves $\max_{y \in B(f)} y(S)$.
- Slightly trickier: Suppose now that S is a *terminal* subset of $\prec$.
  - Since v solves $\max_{y \in B(f)} y(E - S)$, and since $y(E)$ is constant on $B(f)$, we have that v must solve $\min_{y \in B(f)} y(S)$.
- Application to membership of 0 in $B(\hat{f})$: Suppose that $\prec = (1, 2, 3, \ldots, n)$.

Some useful facts about vertices of $B(f)$

- We would start an algorithm with a Greedy vertex v coming from linear order $\prec$.
- Recall that Greedy has the property that if S is an *initial* subset of $\prec$, then $v(S) = f(S)$.
  - Since $y(S) \le f(S)$ for all $y \in B(f)$, clearly v solves $\max_{y \in B(f)} y(S)$.
- Slightly trickier: Suppose now that S is a *terminal* subset of $\prec$.
  - Since v solves $\max_{y \in B(f)} y(E - S)$, and since $y(E)$ is constant on $B(f)$, we have that v must solve $\min_{y \in B(f)} y(S)$.
- Application to membership of 0 in $B(\hat{f})$: Suppose that $\prec = (1, 2, 3, \ldots, n)$.
  - If $v_1 < 0$, then $y(\{1\}) \le \hat{f}(\{1\}) = v_1$ certifies that $0 \notin B(\hat{f})$.

Some useful facts about vertices of $B(f)$

- We would start an algorithm with a Greedy vertex v coming from linear order $\prec$.
- Recall that Greedy has the property that if S is an *initial* subset of $\prec$, then $v(S) = f(S)$.
  - Since $y(S) \leq f(S)$ for all $y \in B(f)$, clearly v solves $\max_{y \in B(f)} y(S)$.
- Slightly trickier: Suppose now that S is a *terminal* subset of $\prec$.
  - Since v solves $\max_{y \in B(f)} y(E - S)$, and since $y(E)$ is constant on $B(f)$, we have that v must solve $\min_{y \in B(f)} y(S)$.
- Application to membership of 0 in $B(\hat{f})$: Suppose that $\prec = (1, 2, 3, \ldots, n)$.
  - If $v_1 < 0$, then $y(\{1\}) \leq \hat{f}(\{1\}) = v_1$ certifies that $0 \notin B(\hat{f})$.
  - If $v_n > 0$, then $-y_n = y(E) - y_n = y(E - \{n\}) \leq \hat{f}(E - \{n\}) = v(E - \{n\}) = v(E) - v_n = -v_n$ certifies that $0 \notin B(\hat{f})$.

Outline

An algorithmic idea

- Let's start with v coming from $\prec$, and we'll concentrate on trying to find an $\tilde{x} \leq 0$.

An algorithmic idea

- Let's start with v coming from $\prec$, and we'll concentrate on trying to find an $\tilde{x} \leq 0$.
- Previous slide showed that we can assume that $v_1 \geq 0$ and $v_n \leq 0$.

An algorithmic idea

- Let's start with v coming from $\prec$, and we'll concentrate on trying to find an $\tilde{x} \leq 0$.
- Previous slide showed that we can assume that $v_1 \geq 0$ and $v_n \leq 0$.
- More generally, we can assume that for any initial subset S of $\prec$, $v(S) \geq 0$ (same proof), and ...

An algorithmic idea

- Let's start with v coming from $\prec$, and we'll concentrate on trying to find an $\tilde{x} \leq 0$.
- Previous slide showed that we can assume that $v_1 \geq 0$ and $v_n \leq 0$.
- More generally, we can assume that for any initial subset S of $\prec$, $v(S) \geq 0$ (same proof), and ...
- ... for any terminal subset S of $\prec$ we can assume that $v(S) \leq 0$ (same proof).

An algorithmic idea

- Let's start with v coming from $\prec$, and we'll concentrate on trying to find an $\tilde{x} \leq 0$.
- Previous slide showed that we can assume that $v_1 \geq 0$ and $v_n \leq 0$.
- More generally, we can assume that for any initial subset S of $\prec$, $v(S) \geq 0$ (same proof), and ...
- ... for any terminal subset S of $\prec$ we can assume that $v(S) \leq 0$ (same proof).
- So now let's try to find combinatorial hull moves that will modify v into the $\tilde{x}$ we need.

An algorithmic idea

- Let's start with v coming from $\prec$, and we'll concentrate on trying to find an $\tilde{x} \leq 0$.
- Previous slide showed that we can assume that $v_1 \geq 0$ and $v_n \leq 0$.
- More generally, we can assume that for any initial subset S of $\prec$, $v(S) \geq 0$ (same proof), and ...
- ... for any terminal subset S of $\prec$ we can assume that $v(S) \leq 0$ (same proof).
- So now let's try to find combinatorial hull moves that will modify v into the $\tilde{x}$ we need.
  - All we need to do is to "re-distribute" the negativity in the terminal elements of v to make every individual component non-positive (not just the terminal partial sums).

A good start

- Suppose that $v_n < 0$ but that $v_{n-1} > 0$.

A good start

- Suppose that $v_n < 0$ but that $v_{n-1} > 0$.
- Consider v' generated by $\prec' = (1, 2, \ldots, n-2, n, n-1)$.

A good start

- Suppose that $v_n < 0$ but that $v_{n-1} > 0$.
- Consider v' generated by $\prec' = (1, 2, \ldots, n-2, n, n-1)$.
- We saw that $v'_n \geq v_n$ and $v'_{n-1} \leq v_{n-1}$, but that $v'_i = v_i$ for all $i < n-1$.

A good start

- Suppose that $v_n < 0$ but that $v_{n-1} > 0$.
- Consider v' generated by $\prec' = (1, 2, \ldots, n-2, n, n-1)$.
- We saw that $v'_n \geq v_n$ and $v'_{n-1} \leq v_{n-1}$, but that $v'_i = v_i$ for all $i < n-1$.
- As we move v in the $v' - v$ direction, v_n increases towards 0, and v_{n-1} decreases towards 0.

A good start

- Suppose that $v_n < 0$ but that $v_{n-1} > 0$.
- Consider v' generated by $\prec' = (1, 2, \ldots, n-2, n, n-1)$.
- We saw that $v'_n \geq v_n$ and $v'_{n-1} \leq v_{n-1}$, but that $v'_i = v_i$ for all $i < n-1$.
- As we move v in the $v' - v$ direction, v_n increases towards 0, and v_{n-1} decreases towards 0.
- This movement is along the edge which is the convex hull of v and v' (this sounds bad), but since $v' - v$ has all signs non-positive (after projecting out coordinate n), this is also a valid combinatorial hull operation (which sounds good).

Three possible outcomes

1. We could stop if v_n hits 0 before v_{n-1}.

Three possible outcomes

1. We could stop if v_n hits 0 before v_{n-1}.
   - But then the terminal partial sum of n and $n-1$ is positive, so we get a proof that $0 \notin B(\hat{f})$.

Three possible outcomes

1. We could stop if v_n hits 0 before v_{n-1}.
   - But then the terminal partial sum of n and $n-1$ is positive, so we get a proof that $0 \notin B(\hat{f})$.
2. We could stop if v_{n-1} hits 0 before v_n.

Three possible outcomes

1. We could stop if v_n hits 0 before v_{n-1}.
   - But then the terminal partial sum of n and $n-1$ is positive, so we get a proof that $0 \notin B(\hat{f})$.
2. We could stop if v_{n-1} hits 0 before v_n.
   - Now we have the last two components of v non-positive, so we could "continue the algorithm" (towards our ideal of having the last $n-1$ components of v non-positive).

Three possible outcomes

1. We could stop if v_n hits 0 before v_{n-1}.
   - But then the terminal partial sum of n and $n-1$ is positive, so we get a proof that $0 \notin B(\hat{f})$.
2. We could stop if v_{n-1} hits 0 before v_n.
   - Now we have the last two components of v non-positive, so we could "continue the algorithm" (towards our ideal of having the last $n-1$ components of v non-positive).
3. We could stop if we move all the way from v to v' and neither one of v_n nor v_{n-1} hits 0.

Three possible outcomes

1. We could stop if v_n hits 0 before v_{n-1}.
   - But then the terminal partial sum of n and $n-1$ is positive, so we get a proof that $0 \notin B(\hat{f})$.
2. We could stop if v_{n-1} hits 0 before v_n.
   - Now we have the last two components of v non-positive, so we could "continue the algorithm" (towards our ideal of having the last $n-1$ components of v non-positive).
3. We could stop if we move all the way from v to v' and neither one of v_n nor v_{n-1} hits 0.
   - Now we have effectively replaced v by v'.

Three possible outcomes

1. We could stop if v_n hits 0 before v_{n-1}.
   - But then the terminal partial sum of n and $n-1$ is positive, so we get a proof that $0 \notin B(\hat{f})$.
2. We could stop if v_{n-1} hits 0 before v_n.
   - Now we have the last two components of v non-positive, so we could "continue the algorithm" (towards our ideal of having the last $n-1$ components of v non-positive).
3. We could stop if we move all the way from v to v' and neither one of v_n nor v_{n-1} hits 0.
   - Now we have effectively replaced v by v'.
   - We ended with $v'_{n-1} > 0$.

Three possible outcomes

1. We could stop if v_n hits 0 before v_{n-1}.
   - But then the terminal partial sum of n and $n-1$ is positive, so we get a proof that $0 \notin B(\hat{f})$.
2. We could stop if v_{n-1} hits 0 before v_n.
   - Now we have the last two components of v non-positive, so we could "continue the algorithm" (towards our ideal of having the last $n-1$ components of v non-positive).
3. We could stop if we move all the way from v to v' and neither one of v_n nor v_{n-1} hits 0.
   - Now we have effectively replaced v by v'.
   - We ended with $v'_{n-1} > 0$.
   - This violates that $y_{n-1} \geq \hat{f}^{\#}(\{n-1\}) = v'_{n-1}$, proving that $0 \notin B(\hat{f})$.

Three possible outcomes

1. We could stop if v_n hits 0 before v_{n-1}.
   - But then the terminal partial sum of n and $n-1$ is positive, so we get a proof that $0 \notin B(\hat{f})$.
2. We could stop if v_{n-1} hits 0 before v_n.
   - Now we have the last two components of v non-positive, so we could "continue the algorithm" (towards our ideal of having the last $n-1$ components of v non-positive).
3. We could stop if we move all the way from v to v' and neither one of v_n nor v_{n-1} hits 0.
   - Now we have effectively replaced v by v'.
   - We ended with $v'_{n-1} > 0$.
   - This violates that $y_{n-1} \geq \hat{f}^{\#}(\{n-1\}) = v'_{n-1}$, proving that $0 \notin B(\hat{f})$.

- In all three case we make real progress.

More complicated

- Suppose instead that $v_n < 0$, but $v_{n-1} = 0$ and $v_{n-2} > 0$.

More complicated

- Suppose instead that $v_n < 0$, but $v_{n-1} = 0$ and $v_{n-2} > 0$.
- Consider now the block change

$$v' = (1\ 2\ \ldots n-3\ n\ n-2\ n-1)$$

More complicated

- Suppose instead that $v_n < 0$, but $v_{n-1} = 0$ and $v_{n-2} > 0$.
- Consider now the block change

$$v' = (1\ 2\ \ldots n-3\ n\ n-2\ n-1)$$

- The same proof shows that $v'_n \geq v_n$, and $v'_{n-2} \leq v_{n-2}$, $v'_{n-1} \leq v_{n-1}$.

More complicated

- Suppose instead that $v_n < 0$, but $v_{n-1} = 0$ and $v_{n-2} > 0$.
- Consider now the block change

$$v' = (1\ 2\ \ldots n-3\ n\ n-2\ n-1)$$

- The same proof shows that $v'_n \geq v_n$, and $v'_{n-2} \leq v_{n-2}$, $v'_{n-1} \leq v_{n-1}$.
- Again if we move v in the $v' - v$ direction, v_n increases towards 0, v_{n-1} decreases from 0, and v_{n-2} decreases towards 0.

More complicated

- Suppose instead that $v_n < 0$, but $v_{n-1} = 0$ and $v_{n-2} > 0$.
- Consider now the block change

$$v' = (1\ 2\ \ldots n-3\ n\ n-2\ n-1)$$

- The same proof shows that $v'_n \geq v_n$, and $v'_{n-2} \leq v_{n-2}$, $v'_{n-1} \leq v_{n-1}$.
- Again if we move v in the $v' - v$ direction, v_n increases towards 0, v_{n-1} decreases from 0, and v_{n-2} decreases towards 0.
- This move is not along an edge, and is a convex hull move (which again sounds bad), but since $v' - v$ has all signs non-positive (after projecting out coordinate n), this is also a combinatorial hull operation (which sounds good).

More complicated

- Suppose instead that $v_n < 0$, but $v_{n-1} = 0$ and $v_{n-2} > 0$.
- Consider now the block change

$$v' = (1\ 2\ \ldots n-3\ n\ n-2\ n-1)$$

- The same proof shows that $v'_n \geq v_n$, and $v'_{n-2} \leq v_{n-2}$, $v'_{n-1} \leq v_{n-1}$.
- Again if we move v in the $v' - v$ direction, v_n increases towards 0, v_{n-1} decreases from 0, and v_{n-2} decreases towards 0.
- This move is not along an edge, and is a convex hull move (which again sounds bad), but since $v' - v$ has all signs non-positive (after projecting out coordinate n), this is also a combinatorial hull operation (which sounds good).
  - This looks non-integral: Suppose that $v = (\ldots, 3, 0, -4)$ and $v' = (\ldots, -2, -1, 2)$. Then we'd move to $(\ldots, 0, -\frac{3}{5}, -\frac{2}{5})$, which is not integral (bad) ...

More complicated

- Suppose instead that $v_n < 0$, but $v_{n-1} = 0$ and $v_{n-2} > 0$.
- Consider now the block change

$$v' = (1\ 2\ \ldots n-3\ n\ n-2\ n-1)$$

- The same proof shows that $v'_n \geq v_n$, and $v'_{n-2} \leq v_{n-2}$, $v'_{n-1} \leq v_{n-1}$.
- Again if we move v in the $v' - v$ direction, v_n increases towards 0, v_{n-1} decreases from 0, and v_{n-2} decreases towards 0.
- This move is not along an edge, and is a convex hull move (which again sounds bad), but since $v' - v$ has all signs non-positive (after projecting out coordinate n), this is also a combinatorial hull operation (which sounds good).
  - This looks non-integral: Suppose that $v = (\ldots, 3, 0, -4)$ and $v' = (\ldots, -2, -1, 2)$. Then we'd move to $(\ldots, 0, -\frac{3}{5}, -\frac{2}{5})$, which is not integral (bad) ...
  - ... but we could move to $(\ldots, 0, -1, 0)$, which is integral (good).

Again three possible outcomes

1. We could stop if v_n hits 0 before v_{n-2}.

Again three possible outcomes

1. We could stop if v_n hits 0 before v_{n-2}.
   - Now we could do a step as before with $v_{n-2} > 0$, $v_{n-1} < 0$.

Again three possible outcomes

1. We could stop if v_n hits 0 before v_{n-2}.
   - Now we could do a step as before with $v_{n-2} > 0$, $v_{n-1} < 0$.
2. We could stop if v_{n-2} hits 0 before v_n.

Again three possible outcomes

1. We could stop if v_n hits 0 before v_{n-2}.
   - Now we could do a step as before with $v_{n-2} > 0$, $v_{n-1} < 0$.
2. We could stop if v_{n-2} hits 0 before v_n.
   - Now we have the last three components of v non-positive, so we could "continue the algorithm".

Again three possible outcomes

1. We could stop if v_n hits 0 before v_{n-2}.
   - Now we could do a step as before with $v_{n-2} > 0$, $v_{n-1} < 0$.
2. We could stop if v_{n-2} hits 0 before v_n.
   - Now we have the last three components of v non-positive, so we could "continue the algorithm".
3. We could stop if we move all the way from v to v' and neither v_n nor v_{n-2} hits 0.

Again three possible outcomes

1. We could stop if v_n hits 0 before v_{n-2}.
   - Now we could do a step as before with $v_{n-2} > 0$, $v_{n-1} < 0$.
2. We could stop if v_{n-2} hits 0 before v_n.
   - Now we have the last three components of v non-positive, so we could "continue the algorithm".
3. We could stop if we move all the way from v to v' and neither v_n nor v_{n-2} hits 0.
   - Now we have effectively replaced v by v'.

Again three possible outcomes

1. We could stop if v_n hits 0 before v_{n-2}.
   - Now we could do a step as before with $v_{n-2} > 0$, $v_{n-1} < 0$.
2. We could stop if v_{n-2} hits 0 before v_n.
   - Now we have the last three components of v non-positive, so we could "continue the algorithm".
3. We could stop if we move all the way from v to v' and neither v_n nor v_{n-2} hits 0.
   - Now we have effectively replaced v by v'.
   - We ended with $v'_n < 0$, $v'_{n-1} \leq 0$, and $v'_{n-2} > 0$.

Again three possible outcomes

1. We could stop if v_n hits 0 before v_{n-2}.
   - Now we could do a step as before with $v_{n-2} > 0$, $v_{n-1} < 0$.
2. We could stop if v_{n-2} hits 0 before v_n.
   - Now we have the last three components of v non-positive, so we could "continue the algorithm".
3. We could stop if we move all the way from v to v' and neither v_n nor v_{n-2} hits 0.
   - Now we have effectively replaced v by v'.
   - We ended with $v'_n < 0$, $v'_{n-1} \leq 0$, and $v'_{n-2} > 0$.
   - If $v'_{n-1} + v'_{n-2} > 0$ then we violate that the partial sum of the last two terms of v' must be non-positive.

Again three possible outcomes

1. We could stop if v_n hits 0 before v_{n-2}.
   - Now we could do a step as before with $v_{n-2} > 0$, $v_{n-1} < 0$.
2. We could stop if v_{n-2} hits 0 before v_n.
   - Now we have the last three components of v non-positive, so we could "continue the algorithm".
3. We could stop if we move all the way from v to v' and neither v_n nor v_{n-2} hits 0.
   - Now we have effectively replaced v by v'.
   - We ended with $v'_n < 0$, $v'_{n-1} \leq 0$, and $v'_{n-2} > 0$.
   - If $v'_{n-1} + v'_{n-2} > 0$ then we violate that the partial sum of the last two terms of v' must be non-positive.
   - If instead $v'_{n-1} + v'_{n-2} \leq 0$ then we can do a step as before where we move v'_{n-1} up and v'_{n-2} down.

Again three possible outcomes

1. We could stop if v_n hits 0 before v_{n-2}.
   - Now we could do a step as before with $v_{n-2} > 0$, $v_{n-1} < 0$.
2. We could stop if v_{n-2} hits 0 before v_n.
   - Now we have the last three components of v non-positive, so we could "continue the algorithm".
3. We could stop if we move all the way from v to v' and neither v_n nor v_{n-2} hits 0.
   - Now we have effectively replaced v by v'.
   - We ended with $v'_n < 0$, $v'_{n-1} \leq 0$, and $v'_{n-2} > 0$.
   - If $v'_{n-1} + v'_{n-2} > 0$ then we violate that the partial sum of the last two terms of v' must be non-positive.
   - If instead $v'_{n-1} + v'_{n-2} \leq 0$ then we can do a step as before where we move v'_{n-1} up and v'_{n-2} down.

- Again we make real progress in all cases.

A more complicated, problematic example

- Suppose that $\prec = (1\ 2\ 3\ 4\ 5)$ and $v = (1, 4, 3, 0, -8)$.

A more complicated, problematic example

- Suppose that $\prec = (1\ 2\ 3\ 4\ 5)$ and $v = (1, 4, 3, 0, -8)$.
- We'd set $\prec_1 = (1\ 2\ {\color{red}5}\ {\color{blue}3}\ {\color{blue}4})$, with, say, $v^1 = (1, 4, -1, -2, -2)$.

A more complicated, problematic example

- Suppose that $\prec = (1\ 2\ 3\ 4\ 5)$ and $v = (1, 4, 3, 0, -8)$.
- We'd set $\prec_1 = (1\ 2\ 5\ 3\ 4)$, with, say, $v^1 = (1, 4, -1, -2, -2)$.
- This would lead to, say, $v' = (1, 4, 0, -1, -4)$.

A more complicated, problematic example

- Suppose that $\prec = (1\ 2\ 3\ 4\ 5)$ and $v = (1, 4, 3, 0, -8)$.
- We'd set $\prec_1 = (1\ 2\ 5\ 3\ 4)$, with, say, $v^1 = (1, 4, -1, -2, -2)$.
- This would lead to, say, $v' = (1, 4, 0, -1, -4)$.
- Now we'd set $\prec_2 = (1\ 4\ 2\ 3\ 5)$, with, say, $v^2 = (1, -1, 1, 7, -8)$.

A more complicated, problematic example

- Suppose that $\prec = (1\ 2\ 3\ 4\ 5)$ and $v = (1, 4, 3, 0, -8)$.
- We'd set $\prec_1 = (1\ 2\ 5\ 3\ 4)$, with, say, $v^1 = (1, 4, -1, -2, -2)$.
- This would lead to, say, $v' = (1, 4, 0, -1, -4)$.
- Now we'd set $\prec_2 = (1\ 4\ 2\ 3\ 5)$, with, say, $v^2 = (1, -1, 1, 7, -8)$.
- We were expecting that after projecting out 4, we'd have that $v^2_2 \leq v'_2$ and $v^2_3 \leq v'_3$, but this is false.

A more complicated, problematic example

- Suppose that $\prec = (1\ 2\ 3\ 4\ 5)$ and $v = (1, 4, 3, 0, -8)$.
- We'd set $\prec_1 = (1\ 2\ 5\ 3\ 4)$, with, say, $v^1 = (1, 4, -1, -2, -2)$.
- This would lead to, say, $v' = (1, 4, 0, -1, -4)$.
- Now we'd set $\prec_2 = (1\ 4\ 2\ 3\ 5)$, with, say, $v^2 = (1, -1, 1, 7, -8)$.
- We were expecting that after projecting out 4, we'd have that $v^2_2 \le v'_2$ and $v^2_3 \le v'_3$, but this is false.
- This is the problem with combinatorial hull: Unlike convex hull, you cannot arbitrarily pile on an operation that works in one place (e.g., $v^2 - v$ is a good direction w.r.t. v) and necessarily have it work in another place (e.g., v^2 doesn't have the right signs w.r.t. v').

(Tentative) conclusions

1. Finding a more combinatorial replacement for ReduceV is important.

(Tentative) conclusions

1. Finding a more combinatorial replacement for $\textsc{ReduceV}$ is important.
2. Combinatorial hull has some good points and some bad points:

(Tentative) conclusions

1. Finding a more combinatorial replacement for $\textsc{ReduceV}$ is important.
2. Combinatorial hull has some good points and some bad points:
   - Good: It proves that $y \in B(f)$ with no linear algebra.

(Tentative) conclusions

1. Finding a more combinatorial replacement for $\textsc{ReduceV}$ is important.
2. Combinatorial hull has some good points and some bad points:
   - Good: It proves that $y \in B(f)$ with no linear algebra.
   - Good: There are some promising algorithmic ideas.

(Tentative) conclusions

1. Finding a more combinatorial replacement for $\textsc{ReduceV}$ is important.
2. Combinatorial hull has some good points and some bad points:
   - Good: It proves that $y \in B(f)$ with no linear algebra.
   - Good: There are some promising algorithmic ideas.
   - Good: These algorithmic ideas preserve integrality.

(Tentative) conclusions

1. Finding a more combinatorial replacement for $\textsc{ReduceV}$ is important.
2. Combinatorial hull has some good points and some bad points:
   - Good: It proves that $y \in B(f)$ with no linear algebra.
   - Good: There are some promising algorithmic ideas.
   - Good: These algorithmic ideas preserve integrality.
   - Bad: These algorithmic ideas don't yet seem strong enough to make combinatorial hull work.

(Tentative) conclusions

1. Finding a more combinatorial replacement for $\textsc{ReduceV}$ is important.
2. Combinatorial hull has some good points and some bad points:
   - Good: It proves that $y \in B(f)$ with no linear algebra.
   - Good: There are some promising algorithmic ideas.
   - Good: These algorithmic ideas preserve integrality.
   - Bad: These algorithmic ideas don't yet seem strong enough to make combinatorial hull work.
3. But we don't have an alternative to combinatorial hull in hand either ...