Short Course on Submodular Functions
Part 2: Extensions and Related Problems
Session 2.A: Partitions

S. Thomas McCormick   Maurice Queyranne

Sauder School of Business, UBC

JPOC Summer School, June 2013

Partitions

Partitions

A *partition* $\mathbf{P} = \{P_1, \ldots, P_k\}$ of E satisfies

- $\emptyset \neq P_i \subseteq E$ for all i,
- $P_i \cap P_j = \emptyset$ for all $i \neq j$, and
- $\cup_{i=1}^{n} P_i = E$

for some $k \in \{1, \ldots, |E|\}$ $\quad$ ($\mathbf{P}$ is a *k-way* partition)

Partitions

A *partition* $\mathbf{P} = \{P_1, \ldots, P_k\}$ of E satisfies

- $\emptyset \neq P_i \subseteq E$ for all i,
- $P_i \cap P_j = \emptyset$ for all $i \neq j$, and
- $\cup_{i=1}^{n} P_i = E$

for some $k \in \{1, \ldots, |E|\}$ ($\mathbf{P}$ is a k-*way* partition)

Given a part cost function $f : 2^E \mapsto \mathbb{R}$, the *cost* of a partition $\mathbf{P}$ is

$$f(\mathbf{P}) = \sum_{i=1}^{|\mathbf{P}|} f(P_i)$$

Partitions

A *partition* $\mathbf{P} = \{P_1, \ldots, P_k\}$ of E satisfies

- $\emptyset \neq P_i \subseteq E$ for all i,
- $P_i \cap P_j = \emptyset$ for all $i \neq j$, and
- $\cup_{i=1}^{n} P_i = E$

for some $k \in \{1, \ldots, |E|\}$ ($\mathbf{P}$ is a *k-way* partition)

Given a part cost function $f : 2^E \mapsto \mathbb{R}$, the *cost* of a partition $\mathbf{P}$ is

$$f(\mathbf{P}) = \sum_{i=1}^{|\mathbf{P}|} f(P_i)$$

Optimum Partition Problems:

- given E and f
- find a partition $\mathbf{P}$ with minimum cost $f(\mathbf{P})$
  (subject to possible restrictions on the number $k = |\mathbf{P}|$ of parts)

Applications

Applications

Set Partitioning

- not all subsets are feasible
  $\Rightarrow$ let $f(S) = +\infty$ whenever S is not feasible
- many applications, e.g., airline crew scheduling, vehicle routing, etc.

Applications

Set Partitioning

- not all subsets are feasible
  $\Rightarrow$ let $f(S) = +\infty$ whenever S is not feasible
- many applications, e.g., airline crew scheduling, vehicle routing, etc.

Facility Location/Allocation

- E is a set of *clients* to be served
- $f(S)$ is the minimum cost to serve subset S
  (choosing a best *location* for serving S)

Applications

Set Partitioning

- not all subsets are feasible
  $\Rightarrow$ let $f(S) = +\infty$ whenever S is not feasible
- many applications, e.g., airline crew scheduling, vehicle routing, etc.

Facility Location/Allocation

- E is a set of *clients* to be served
- $f(S)$ is the minimum cost to serve subset S
  (choosing a best *location* for serving S)

Clustering

- E is a set of items to be classified
- $f(S)$ is the (negative of) the value of *cluster* S, reflecting
  - the similarities within S, and
  - the dissimilarities with $N \setminus S$

Applications (2)

Multi-layer VLSI Circuit Design (Netlist Partitioning)

- E is a set of *modules* to be located on a *k-layer chip*
  $\Rightarrow$ find a k-way partition of E
- $f(S)$ is the cost of splitting *netlist* S

Applications (2)

Multi-layer VLSI Circuit Design (Netlist Partitioning)

- E is a set of *modules* to be located on a *k-layer chip*
  $\Rightarrow$ find a k-way partition of E
- $f(S)$ is the cost of splitting *netlist* S

In most applications, there are additional constraints:

- on the parts P_i
  - e.g., VLSI: each part must fit on one layer
- other "complicating" constraints
  - e.g., Set Partitioning: aircraft types, home bases

Applications (2)

Multi-layer VLSI Circuit Design (Netlist Partitioning)

- E is a set of *modules* to be located on a *k-layer chip*
  $\Rightarrow$ find a k-way partition of E
- $f(S)$ is the cost of splitting *netlist* S

In most applications, there are additional constraints:

- on the parts P_i
  - e.g., VLSI: each part must fit on one layer
- other "complicating" constraints
  - e.g., Set Partitioning: aircraft types, home bases

Most of these problems are NP-hard

- many are hard to approximate
- just finding feasible solutions can be NP-hard

Applications (2)

Multi-layer VLSI Circuit Design (Netlist Partitioning)

- E is a set of *modules* to be located on a *k-layer chip*
  $\Rightarrow$ find a k-way partition of E
- $f(S)$ is the cost of splitting *netlist* S

In most applications, there are additional constraints:

- on the parts P_i
  - e.g., VLSI: each part must fit on one layer
- other "complicating" constraints
  - e.g., Set Partitioning: aircraft types, home bases

Most of these problems are NP-hard

- many are hard to approximate
- just finding feasible solutions can be NP-hard

Yet, some important and useful special cases can be solved efficiently when the cost function f is **submodular**

Examples with Submodular Part Costs

Some important and useful special cases can be solved efficiently when the cost function f is **submodular**:

Examples with Submodular Part Costs

Some important and useful special cases can be solved efficiently when the cost function f is **submodular**:

Clustering
The negative of *total* (pairwise) *similarity*

$$f(S) = -\sum_{j,k \in S} s(j,k)$$

is submodular when $s \geq 0$ (Why?)

Examples with Submodular Part Costs

Some important and useful special cases can be solved efficiently when the cost function f is **submodular**:

Clustering

The negative of *total* (pairwise) *similarity*

$$f(S) = -\sum_{j,k \in S} s(j,k)$$

is submodular when $s \geq 0$ (Why?)

VLSI Circuit Design

Given hypergraph (E, H) with edge weights w_h ($h \in H$), the *hypergraph cut function*

$$f(S) = \sum \{w_h \ : \ h \cap S \neq \emptyset \ \text{ and } \ h \setminus S \neq \emptyset\}$$

is submodular when $w \geq 0$ (Why?)

Optimum Unconstrained Partitions

Optimum Unconstrained Partitions

The *Dilworth truncation* f^D of a set function $f : 2^E \mapsto \mathbb{R}^N$ is the set function $f^D : 2^E \mapsto \mathbb{R}^N$ defined by

$$f^D(A) = \begin{cases} \min_{\mathbf{P} \in \Pi(A)} f(\mathbf{P}) & \text{if } A \neq \emptyset \\ 0 & \text{if } A = \emptyset \end{cases}$$

where $\Pi(A)$ is the set of all partitions of set A

Optimum Unconstrained Partitions

The *Dilworth truncation* f^D of a set function $f : 2^E \mapsto \mathbb{R}^N$ is the set function $f^D : 2^E \mapsto \mathbb{R}^N$ defined by

$$f^D(A) = \begin{cases} \min_{\mathbf{P} \in \Pi(A)} f(\mathbf{P}) & \text{if } A \neq \emptyset \\ 0 & \text{if } A = \emptyset \end{cases}$$

where $\Pi(A)$ is the set of all partitions of set A

Set partitioning formulation: w.l.o.g., assume $A = E$

Let $x_S = \begin{cases} 1 & \text{if } S \in \mathbf{P}; \\ 0 & \text{otherwise} \end{cases}$

$$\begin{aligned} f^D(E) = \quad \min \quad & \textstyle\sum_{S \subseteq E : S \neq \emptyset} f(S)\, x_S \\ \text{s.t.} \quad & \textstyle\sum_{S \subseteq E : j \in S} x_S = 1 \quad \forall j \in E \\ & x \geq 0 \\ & x \text{ integer} \end{aligned}$$

LPs and Dilworth Truncation

LPs and Dilworth Truncation

LP relaxation:

$$
\begin{array}{llll}
(P) & \min & \sum_{S \subseteq E : S \neq \emptyset} & f(S)\, x_S \\
 & \text{s.t.} & \sum_{S \subseteq E : j \in S} & x_S = 1 \quad \forall j \in E \\
 & & x \geq 0 &
\end{array}
$$

LP relaxation:

$$
\begin{array}{llll}
(P) & \min & \sum_{S\subseteq E: S\neq\emptyset} & f(S)\, x_S \\
& \text{s.t.} & \sum_{S\subseteq E: j\in S} & x_S \quad = 1 \quad \forall j \in E \\
& & x \geq 0 &
\end{array}
$$

Its dual:

$$
\begin{array}{lll}
(D) & \max & \displaystyle\sum_{j\in E} y_j \\
& \text{s.t.} & y(S) \;\leq f(S) \quad \forall S \subseteq E,\; S \neq \emptyset
\end{array}
$$

LPs and Dilworth Truncation

LP relaxation:

$$\begin{array}{llll} (P) & \min & \sum_{S\subseteq E: S\neq\emptyset} \; f(S)\, x_S & \\ & \text{s.t.} & \sum_{S\subseteq E: j\in S} \quad x_S \; = 1 & \forall j \in E \\ & & x \geq 0 & \end{array}$$

Its dual:

$$\begin{array}{llll} (D) & \max & \sum_{j\in E} y_j & \\ & \text{s.t.} & y(S) \; \leq f(S) & \forall S\subseteq E,\ S\neq\emptyset \end{array}$$

This dual is almost linear optimization on a submodular polyhedron (solvable by the Greedy Algorithm seen yesterday)

LPs and Dilworth Truncation

LP relaxation:

$$\begin{array}{llll} (P) & \min & \sum_{S \subseteq E : S \neq \emptyset} & f(S)\, x_S \\ & \text{s.t.} & \sum_{S \subseteq E : j \in S} & x_S \quad = 1 \quad \forall j \in E \\ & & x \geq 0 & \end{array}$$

Its dual:

$$\begin{array}{lll} (D) & \max & \displaystyle\sum_{j \in E} y_j \\ & \text{s.t.} & y(S) \leq f(S) \quad \forall S \subseteq E,\ S \neq \emptyset \end{array}$$

This dual is almost linear optimization on a submodular polyhedron (solvable by the Greedy Algorithm seen yesterday)

- except that here we may have $f(\emptyset) < 0$

What if $f(\emptyset) \geq 0$?

What if $f(\emptyset) \geq 0$?

If f is submodular and $f(\emptyset) \geq 0$ then: $A \cap B = \emptyset$ implies

$$f(A \cup B) \leq f(A) + f(B)$$

that is, f is *subadditive*

What if $f(\emptyset) \geq 0$?

If f is submodular and $f(\emptyset) \geq 0$ then: $A \cap B = \emptyset$ implies

$$f(A \cup B) \leq f(A) + f(B)$$

that is, f is *subadditive*

If f is subadditive, then $f^D = f$

- except perhaps that $f^D(\emptyset) = 0$

and we are done.

What if $f(\emptyset) \geq 0$?

If f is submodular and $f(\emptyset) \geq 0$ then: $A \cap B = \emptyset$ implies

$$f(A \cup B) \leq f(A) + f(B)$$

that is, f is *subadditive*

If f is subadditive, then $f^D = f$

- except perhaps that $f^D(\emptyset) = 0$

and we are done.

Hence we now consider the general case where we make no sign restriction on $f(\emptyset)$

A General Greedy Principle

(Edmonds, 1970; Frank & Tardos, 1988)

A General Greedy Principle

(Edmonds, 1970; Frank & Tardos, 1988)

Given polyhedron $P \subseteq \mathbb{R}^E$ and $w \in \mathbb{R}^E$, assume w.l.o.g. that $E = \{e_1, \ldots, e_n\}$ with $w_{e_1} \geq w_{e_2} \geq \cdots \geq w_{e_n} \geq 0$

- i.e., E is totally ordered by $\prec$ as: $e_1 \prec e_2 \prec \cdots \prec e_n$

A General Greedy Principle

(Edmonds, 1970; Frank & Tardos, 1988)

Given polyhedron $P \subseteq \mathbb{R}^E$ and $w \in \mathbb{R}^E$, assume w.l.o.g. that $E = \{e_1, \ldots, e_n\}$ with $w_{e_1} \geq w_{e_2} \geq \cdots \geq w_{e_n} \geq 0$

- i.e., E is totally ordered by $\prec$ as: $e_1 \prec e_2 \prec \cdots \prec e_n$

Recursively define $y^G \in \mathbb{R}^E$ as follows

- for $j = 1, \ldots, n$ let
  $y^G_{e_j} = \max\{y_{e_j} : \exists y \in P \quad \forall i < j \quad y_{e_i} = y^G_{e_i}\}$

A General Greedy Principle

(Edmonds, 1970; Frank & Tardos, 1988)

Given polyhedron $P \subseteq \mathbb{R}^E$ and $w \in \mathbb{R}^E$, assume w.l.o.g. that $E = \{e_1, \ldots, e_n\}$ with $w_{e_1} \geq w_{e_2} \geq \cdots \geq w_{e_n} \geq 0$

- i.e., E is totally ordered by $\prec$ as: $e_1 \prec e_2 \prec \cdots \prec e_n$

Recursively define $y^G \in \mathbb{R}^E$ as follows

- for $j = 1, \ldots, n$ let
  $y^G_{e_j} = \max\{y_{e_j} : \exists y \in P \quad \forall i < j \quad y_{e_i} = y^G_{e_i}\}$

This ensures that the resulting *greedy solution* $y^G \in P$

- at the expense of solving n optimization problems

A General Greedy Principle

(Edmonds, 1970; Frank & Tardos, 1988)

Given polyhedron $P \subseteq \mathbb{R}^E$ and $w \in \mathbb{R}^E$, assume w.l.o.g. that $E = \{e_1, \ldots, e_n\}$ with $w_{e_1} \geq w_{e_2} \geq \cdots \geq w_{e_n} \geq 0$

- i.e., E is totally ordered by $\prec$ as: $e_1 \prec e_2 \prec \cdots \prec e_n$

Recursively define $y^G \in \mathbb{R}^E$ as follows

- for $j = 1, \ldots, n$ let
  $y^G_{e_j} = \max\{y_{e_j} : \exists y \in P \quad \forall i < j \quad y_{e_i} = y^G_{e_i}\}$

This ensures that the resulting *greedy solution* $y^G \in P$

- at the expense of solving n optimization problems

If $P = \tilde{P}(f) = \{y \in \mathbb{R}^E : y(S) \leq f(S) \ \forall S \subseteq E, \ S \neq \emptyset\}$ for some set function f, then

A General Greedy Principle

(Edmonds, 1970; Frank & Tardos, 1988)

Given polyhedron $P \subseteq \mathbb{R}^E$ and $w \in \mathbb{R}^E$, assume w.l.o.g. that $E = \{e_1, \ldots, e_n\}$ with $w_{e_1} \geq w_{e_2} \geq \cdots \geq w_{e_n} \geq 0$

- i.e., E is totally ordered by $\prec$ as: $e_1 \prec e_2 \prec \cdots \prec e_n$

Recursively define $y^G \in \mathbb{R}^E$ as follows

- for $j = 1, \ldots, n$ let
  $y^G_{e_j} = \max\{y_{e_j} : \exists y \in P \quad \forall i < j \quad y_{e_i} = y^G_{e_i}\}$

This ensures that the resulting *greedy solution* $y^G \in P$

- at the expense of solving n optimization problems

If $P = \tilde{P}(f) = \{y \in \mathbb{R}^E : y(S) \leq f(S)\ \forall S \subseteq E,\ S \neq \emptyset\}$ for some set function f, then the Greedy Principle simplifies to:

- let $y^G(e_1) = f(\{e_1\})$ and for $j = 2, \ldots, n$ let

$$y^G_{e_j} = \min\left\{f(A + e_j) - y^G(A) : A \subseteq e_j^{\prec}\right\} \tag{1}$$

where $e_j^{\prec} = \{g \in A : g \prec e_j\} = \{e_1, \ldots, e_{j-1}\}$ for all $j = 1, \ldots, n$

Questions about the General Greedy Principle

Questions about the General Greedy Principle

Optimality Questions:

- is y^G an optimum solution to $\max\{wy : y \in P\}$?
- is the corresponding primal solution x^G integer?

Questions about the General Greedy Principle

Optimality Questions:

- is y^G an optimum solution to $\max\{wy : y \in P\}$?
- is the corresponding primal solution x^G integer?

Algorithmic Questions:

- can the optimization subroblem (1) be solved efficiently (i.e., in polynomial time)?
- if x^G is integer, can the corresponding optimal partition be recovered efficiently?

Questions about the General Greedy Principle

Optimality Questions:

- is y^G an optimum solution to $\max\{wy : y \in P\}$?
- is the corresponding primal solution x^G integer?

Algorithmic Questions:

- can the optimization subroblem (1) be solved efficiently (i.e., in polynomial time)?
- if x^G is integer, can the corresponding optimal partition be recovered efficiently?

We have seen that when f is submodular and normalized (as in $f(\emptyset) = 0$), the answer to all 4 questions is *YES!*

Questions about the General Greedy Principle

Optimality Questions:

- is y^G an optimum solution to $\max\{wy : y \in P\}$?
- is the corresponding primal solution x^G integer?

Algorithmic Questions:

- can the optimization subroblem (1) be solved efficiently (i.e., in polynomial time)?
- if x^G is integer, can the corresponding optimal partition be recovered efficiently?

We have seen that when f is submodular and normalized (as in $f(\emptyset) = 0$), the answer to all 4 questions is *YES!*
• in particular, subproblem (1) is solved as

$$y^G_{e_j} = \min\left\{f(A + e_j) - y^G(A) : A \subseteq e_j^{\prec}\right\}$$

Questions about the General Greedy Principle

Optimality Questions:

- is y^G an optimum solution to $\max\{wy : y \in P\}$?
- is the corresponding primal solution x^G integer?

Algorithmic Questions:

- can the optimization subroblem (1) be solved efficiently (i.e., in polynomial time)?
- if x^G is integer, can the corresponding optimal partition be recovered efficiently?

We have seen that when f is submodular and normalized (as in $f(\emptyset) = 0$), the answer to all 4 questions is *YES!*
• in particular, subproblem (1) is solved as

$$y^G_{e_j} = \min\left\{f(A + e_j) - y^G(A) : A \subseteq e_j^{\prec}\right\} = e_{j+1}^{\prec} - e_j^{\prec}$$

(i.e., optimum subset $A = e_j^{\prec}$)

Uncrossing Lemma

Consider the "general submodular case", where f is submodular and $f(\emptyset)$ is arbitrary

Uncrossing Lemma

Consider the "general submodular case", where f is submodular and $f(\emptyset)$ is arbitrary

Let A_j be an optimum subset in subproblem (1) and $B_j = A_j + e_j$

- so $y^G(B_j) = f(B_j)$

Uncrossing Lemma

Consider the "general submodular case", where f is submodular and $f(\emptyset)$ is arbitrary

Let A_j be an optimum subset in subproblem (1) and $B_j = A_j + e_j$

- so $y^G(B_j) = f(B_j)$

Uncrossing Lemma: *If $B_i \cap B_j \neq \emptyset$ for $i < j$ then $y^G(B_i \cup B_j) = f(B_i \cup B_j)$*

Uncrossing Lemma

Consider the "general submodular case", where f is submodular and $f(\emptyset)$ is arbitrary

Let A_j be an optimum subset in subproblem (1) and $B_j = A_j + e_j$

- so $y^G(B_j) = f(B_j)$

Uncrossing Lemma: *If $B_i \cap B_j \neq \emptyset$ for $i < j$ then $y^G(B_i \cup B_j) = f(B_i \cup B_j)$*

Proof.
Since $y^G \in P$ we have:

$$f(B_j \cup B_i) \leq f(B_j) + f(B_i) - f(B_i \cap B_j)$$

Uncrossing Lemma

Consider the "general submodular case", where f is submodular and $f(\emptyset)$ is arbitrary

Let A_j be an optimum subset in subproblem (1) and $B_j = A_j + e_j$

- so $y^G(B_j) = f(B_j)$

Uncrossing Lemma: *If $B_i \cap B_j \neq \emptyset$ for $i < j$ then* $y^G(B_i \cup B_j) = f(B_i \cup B_j)$

Proof.

Since $y^G \in P$ we have:

$$\begin{aligned} f(B_j \cup B_i) &\leq f(B_j) + f(B_i) - f(B_i \cap B_j) \\ &\leq y^G(B_j) + y^G(B_i) - y^G(B_i \cap B_j) \end{aligned}$$

Uncrossing Lemma

Consider the "general submodular case", where f is submodular and $f(\emptyset)$ is arbitrary

Let A_j be an optimum subset in subproblem (1) and $B_j = A_j + e_j$

- so $y^G(B_j) = f(B_j)$

Uncrossing Lemma: *If $B_i \cap B_j \neq \emptyset$ for $i < j$ then $y^G(B_i \cup B_j) = f(B_i \cup B_j)$*

Proof.

Since $y^G \in P$ we have:

$$\begin{aligned} f(B_j \cup B_i) &\leq f(B_j) + f(B_i) - f(B_i \cap B_j) \\ &\leq y^G(B_j) + y^G(B_i) - y^G(B_i \cap B_j) \\ &= y^G(B_j \cup B_i) \end{aligned}$$

Uncrossing Lemma

Consider the "general submodular case", where f is submodular and $f(\emptyset)$ is arbitrary

Let A_j be an optimum subset in subproblem (1) and $B_j = A_j + e_j$

- so $y^G(B_j) = f(B_j)$

Uncrossing Lemma: *If $B_i \cap B_j \neq \emptyset$ for $i < j$ then* $y^G(B_i \cup B_j) = f(B_i \cup B_j)$

Proof.

Since $y^G \in P$ we have:

$$\begin{aligned} f(B_j \cup B_i) &\leq f(B_j) + f(B_i) - f(B_i \cap B_j) \\ &\leq y^G(B_j) + y^G(B_i) - y^G(B_i \cap B_j) \\ &= y^G(B_j \cup B_i) \\ &\leq f(B_j \cup B_i) \end{aligned}$$

Uncrossing Lemma

Consider the "general submodular case", where f is submodular and $f(\emptyset)$ is arbitrary

Let A_j be an optimum subset in subproblem (1) and $B_j = A_j + e_j$

- so $y^G(B_j) = f(B_j)$

Uncrossing Lemma: *If $B_i \cap B_j \neq \emptyset$ for $i < j$ then* $y^G(B_i \cup B_j) = f(B_i \cup B_j)$

Proof.

Since $y^G \in P$ we have:

$$\begin{aligned} f(B_j \cup B_i) &\leq f(B_j) + f(B_i) - f(B_i \cap B_j) \\ &\leq y^G(B_j) + y^G(B_i) - y^G(B_i \cap B_j) \\ &= y^G(B_j \cup B_i) \\ &\leq f(B_j \cup B_i) \end{aligned}$$

hence all these inequalities must hold as equalities □

General Submodular Case: Optimality

General Submodular Case: Optimality

By the Uncrossing Lemma, at each step of the Greedy Algorithm, we may replace the current set B_j with its union with all earlier sets that it intersects, and delete all these earlier intersected sets

General Submodular Case: Optimality

By the Uncrossing Lemma, at each step of the Greedy Algorithm, we may replace the current set B_j with its union with all earlier sets that it intersects, and delete all these earlier intersected sets

At the end, the *surviving sets*, say, $P_1, \ldots, P_k$ form a partition of E and $y^G(E) = \sum_i f(P_i)$

General Submodular Case: Optimality

By the Uncrossing Lemma, at each step of the Greedy Algorithm, we may replace the current set B_j with its union with all earlier sets that it intersects, and delete all these earlier intersected sets

At the end, the *surviving sets*, say, $P_1, \ldots, P_k$ form a partition of E and $y^G(E) = \sum_i f(P_i)$

This implies that the primal solution x^G defined by

$$x^G(S) = \begin{cases} 1 & \text{if } S = P_i \text{ for some } i; \\ 0 & \text{otherwise} \end{cases}$$

General Submodular Case: Optimality

By the Uncrossing Lemma, at each step of the Greedy Algorithm, we may replace the current set B_j with its union with all earlier sets that it intersects, and delete all these earlier intersected sets

At the end, the *surviving sets*, say, $P_1, \ldots, P_k$ form a partition of E and $y^G(E) = \sum_i f(P_i)$

This implies that the primal solution x^G defined by

$$x^G(S) = \begin{cases} 1 & \text{if } S = P_i \text{ for some } i; \\ 0 & \text{otherwise} \end{cases}$$

is feasible for (P) and the primal and dual objective values

$$\sum_S f(S)\, x_S^G = \sum_{j=1}^{n} y_j^G$$

General Submodular Case: Optimality

By the Uncrossing Lemma, at each step of the Greedy Algorithm, we may replace the current set B_j with its union with all earlier sets that it intersects, and delete all these earlier intersected sets

At the end, the *surviving sets*, say, $P_1, \ldots, P_k$ form a partition of E and $y^G(E) = \sum_i f(P_i)$

This implies that the primal solution x^G defined by

$$x^G(S) = \begin{cases} 1 & \text{if } S = P_i \text{ for some } i; \\ 0 & \text{otherwise} \end{cases}$$

is feasible for (P) and the primal and dual objective values

$$\sum_S f(S)\, x_S^G = \sum_{j=1}^{n} y_j^G$$

Hence *both y^G and x^G are optimal*, answering both Optimality Questions, and giving an efficient construction of an *optimum partition $\mathbf{P} = (P_1, \ldots, P_k)$*

Algorithmic Questions; Submodularity of Dilworth Truncation

The optimization subproblem (1) is SFMin

Algorithmic Questions; Submodularity of Dilworth Truncation

The optimization subproblem (1) is SFMin

Therefore, for any subset $S \subseteq E$, the *value* $f^D(S)$ *of the Dilworth truncation* can be obtained in polynomial time, by solving $|S| - 1$ submodular minimization problems

Algorithmic Questions; Submodularity of Dilworth Truncation

The optimization subproblem (1) is SFMin

Therefore, for any subset $S \subseteq E$, the *value $f^D(S)$ of the Dilworth truncation* can be obtained in polynomial time, by solving $|S| - 1$ submodular minimization problems

Submodularity of the Dilworth Truncation

Algorithmic Questions; Submodularity of Dilworth Truncation

The optimization subproblem (1) is SFMin
Therefore, for any subset $S \subseteq E$, the *value $f^D(S)$ of the Dilworth truncation* can be obtained in polynomial time, by solving $|S| - 1$ submodular minimization problems

Submodularity of the Dilworth Truncation

Proposition (Lovász 1983) *The Dilworth truncation of a submodular function is submodular*
Proof: Let f be submodular.

Algorithmic Questions; Submodularity of Dilworth Truncation

The optimization subproblem (1) is SFMin
Therefore, for any subset $S \subseteq E$, the *value $f^D(S)$ of the Dilworth truncation* can be obtained in polynomial time, by solving $|S| - 1$ submodular minimization problems

Submodularity of the Dilworth Truncation

Proposition (Lovász 1983) *The Dilworth truncation of a submodular function is submodular*
Proof: Let f be submodular. Recall that $f^D(\emptyset) = 0$

Algorithmic Questions; Submodularity of Dilworth Truncation

The optimization subproblem (1) is SFMin

Therefore, for any subset $S \subseteq E$, the *value $f^D(S)$ of the Dilworth truncation* can be obtained in polynomial time, by solving $|S| - 1$ submodular minimization problems

Submodularity of the Dilworth Truncation

Proposition (Lovász 1983) *The Dilworth truncation of a submodular function is submodular*

Proof: Let f be submodular. Recall that $f^D(\emptyset) = 0$

It suffices to prove: for all $S \subset E$, $u, v \in E \setminus S$

$$f^D(S+u+v) - f^D(S+u) \leq f^D(S+v) - f^D(S) \; ?$$

Algorithmic Questions; Submodularity of Dilworth Truncation

The optimization subproblem (1) is SFMin

Therefore, for any subset $S \subseteq E$, the *value $f^D(S)$ of the Dilworth truncation* can be obtained in polynomial time, by solving $|S| - 1$ submodular minimization problems

Submodularity of the Dilworth Truncation

Proposition (Lovász 1983) *The Dilworth truncation of a submodular function is submodular*

Proof: Let f be submodular. Recall that $f^D(\emptyset) = 0$

It suffices to prove: for all $S \subset E$, $u, v \in E \setminus S$

$$f^D(S + u + v) - f^D(S + u) \leq f^D(S + v) - f^D(S) \;?$$

- If $S = \emptyset$ then $f^D(u + v) \leq f^D(u) + f^D(v)$ (Why?)

Submodularity of the Dilworth Truncation (continued)

Else, i.e., $S \neq \emptyset$, number the elements in E so $S = e^{\prec}_{i+1}$, $e_{i+1} = u$ and $e_{i+2} = v$

Submodularity of the Dilworth Truncation (continued)

Else, i.e., $S \neq \emptyset$, number the elements in E so $S = e^{\prec}_{i+1}$, $e_{i+1} = u$ and $e_{i+2} = v$ and apply the Greedy Algorithm: we have

$$y^G(T) = f^D(T) \qquad \text{for } T = S,\ S+u,\ \text{and } S+u+v$$

Submodularity of the Dilworth Truncation (continued)

Else, i.e., $S \neq \emptyset$, number the elements in E so $S = e^{\prec}_{i+1}$, $e_{i+1} = u$ and $e_{i+2} = v$ and apply the Greedy Algorithm: we have

$$y^G(T) = f^D(T) \qquad \text{for } T = S,\ S+u,\ \text{and } S+u+v$$

The Greedy Algorithm applied to $S + v$ just after S produces $\tilde{y}^G$ satisfying $\quad \tilde{y}^G_j = y^G_j$ for all $j \in S$

Submodularity of the Dilworth Truncation (continued)

Else, i.e., $S \neq \emptyset$, number the elements in E so $S = e_{i+1}^{\prec}$, $e_{i+1} = u$ and $e_{i+2} = v$ and apply the Greedy Algorithm: we have

$$y^G(T) = f^D(T) \qquad \text{for } T = S,\ S+u, \text{ and } S+u+v$$

The Greedy Algorithm applied to $S+v$ just after S produces $\tilde{y}^G$ satisfying $\quad \tilde{y}_j^G = y_j^G$ for all $j \in S \quad$ and for some $A \subseteq S$

$$f^D(S+v) - f^D(S) = \tilde{y}_v^G = f(A+v) - \tilde{y}^G(A)$$

Submodularity of the Dilworth Truncation (continued)

Else, i.e., $S \neq \emptyset$, number the elements in E so $S = e_{i+1}^{\prec}$, $e_{i+1} = u$ and $e_{i+2} = v$ and apply the Greedy Algorithm: we have

$$y^G(T) = f^D(T) \qquad \text{for } T = S,\ S+u, \text{ and } S+u+v$$

The Greedy Algorithm applied to $S+v$ just after S produces $\tilde{y}^G$ satisfying $\quad \tilde{y}_j^G = y_j^G$ for all $j \in S \quad$ and for some $A \subseteq S$

$$f^D(S+v) - f^D(S) = \tilde{y}_v^G = f(A+v) - \tilde{y}^G(A)$$

Then:

$$f^D(S+u+v) - f^D(S+u)$$

$$= \quad y_v^G$$

Submodularity of the Dilworth Truncation (continued)

Else, i.e., $S \neq \emptyset$, number the elements in E so $S = e^{\prec}_{i+1}$, $e_{i+1} = u$ and $e_{i+2} = v$ and apply the Greedy Algorithm: we have

$$y^G(T) = f^D(T) \qquad \text{for } T = S,\ S+u, \text{ and } S+u+v$$

The Greedy Algorithm applied to $S + v$ just after S produces $\tilde{y}^G$ satisfying $\tilde{y}^G_j = y^G_j$ for all $j \in S$ and for some $A \subseteq S$

$$f^D(S+v) - f^D(S) = \tilde{y}^G_v = f(A+v) - \tilde{y}^G(A)$$

Then:

$$\begin{aligned} & f^D(S+u+v) - f^D(S+u) \\ & \quad = \quad y^G_v \\ & \quad = \quad \min\{f(B+v) - y^G(B) : B \subseteq S+u\} \end{aligned}$$

Submodularity of the Dilworth Truncation (continued)

Else, i.e., $S \neq \emptyset$, number the elements in E so $S = e_{i+1}^{\prec}$, $e_{i+1} = u$ and $e_{i+2} = v$ and apply the Greedy Algorithm: we have

$$y^G(T) = f^D(T) \qquad \text{for } T = S,\ S+u, \text{ and } S+u+v$$

The Greedy Algorithm applied to $S+v$ just after S produces $\tilde{y}^G$ satisfying $\tilde{y}^G_j = y^G_j$ for all $j \in S$ and for some $A \subseteq S$

$$f^D(S+v) - f^D(S) = \tilde{y}^G_v = f(A+v) - \tilde{y}^G(A)$$

Then:

$$\begin{aligned} f^D(S+u+v) - f^D(S+u) & \\ &= y^G_v \\ &= \min\{f(B+v) - y^G(B) : B \subseteq S+u\} \\ &\leq f(A+v) - y^G(A) \end{aligned}$$

Submodularity of the Dilworth Truncation (continued)

Else, i.e., $S \neq \emptyset$, number the elements in E so $S = e_{i+1}^{\prec}$, $e_{i+1} = u$ and $e_{i+2} = v$ and apply the Greedy Algorithm: we have

$$y^G(T) = f^D(T) \qquad \text{for } T = S,\ S+u,\ \text{and } S+u+v$$

The Greedy Algorithm applied to $S+v$ just after S produces $\tilde{y}^G$ satisfying $\quad \tilde{y}^G_j = y^G_j$ for all $j \in S \quad$ and for some $A \subseteq S$

$$f^D(S+v) - f^D(S) = \tilde{y}^G_v = f(A+v) - \tilde{y}^G(A)$$

Then:

$$\begin{aligned} f^D(S+u+v) - f^D(S+u) & \\ &= y^G_v \\ &= \min\{f(B+v) - y^G(B) : B \subseteq S+u\} \\ &\leq f(A+v) - y^G(A) \\ &= f^D(S+v) - f^D(S) \end{aligned}$$

Submodularity of the Dilworth Truncation (continued)

Else, i.e., $S \neq \emptyset$, number the elements in E so $S = e^{\prec}_{i+1}$, $e_{i+1} = u$ and $e_{i+2} = v$ and apply the Greedy Algorithm: we have

$$y^G(T) = f^D(T) \qquad \text{for } T = S,\ S+u,\ \text{and } S+u+v$$

The Greedy Algorithm applied to $S+v$ just after S produces $\tilde{y}^G$ satisfying $\tilde{y}^G_j = y^G_j$ for all $j \in S$ and for some $A \subseteq S$

$$f^D(S+v) - f^D(S) = \tilde{y}^G_v = f(A+v) - \tilde{y}^G(A)$$

Then:

$$\begin{aligned} f^D(S+u+v) - f^D(S+u) & \\ &= y^G_v \\ &= \min\{f(B+v) - y^G(B) : B \subseteq S+u\} \\ &\leq f(A+v) - y^G(A) \\ &= f^D(S+v) - f^D(S) \end{aligned}$$

QED

Asymptotics of Potts Partition Functions
(Anglès d'Auriac & al., 2002)

Statistical Mechanics	*Graph Theory*
Lattice (V, E)	Graph $G = (V, E)$
Site $i \in V$	Node
Bond $ij \in E$	Edge
Coupling K_{ij}	Edge weight

An Application in Statistical Mechanics

Asymptotics of Potts Partition Functions
(Anglès d'Auriac & al., 2002)

Statistical Mechanics	*Graph Theory*
Lattice (V, E)	Graph $G = (V, E)$
Site $i \in V$	Node
Bond $ij \in E$	Edge
Coupling K_{ij}	Edge weight

Given are: the lattice, the couplings $K \geq 0$, and integer $q \geq 2$ (*number of spin values*)

An Application in Statistical Mechanics

Asymptotics of Potts Partition Functions
(Anglès d'Auriac & al., 2002)

Statistical Mechanics	*Graph Theory*
Lattice (V, E)	Graph $G = (V, E)$
Site $i \in V$	Node
Bond $ij \in E$	Edge
Coupling K_{ij}	Edge weight

Given are: the lattice, the couplings $K \geq 0$, and integer $q \geq 2$ (*number of spin values*)

A variable $\sigma_i \in \{0, 1, \ldots, q-1\}$, called a *spin*, is associated with each site $i \in V$

An Application in Statistical Mechanics

Asymptotics of Potts Partition Functions
(Anglès d'Auriac & al., 2002)

Statistical Mechanics	*Graph Theory*
Lattice (V, E)	Graph $G = (V, E)$
Site $i \in V$	Node
Bond $ij \in E$	Edge
Coupling K_{ij}	Edge weight

Given are: the lattice, the couplings $K \geq 0$, and integer $q \geq 2$ (*number of spin values*)

A variable $\sigma_i \in \{0, 1, \ldots, q-1\}$, called a *spin*, is associated with each site $i \in V$

Energy of *configuration* $\sigma = (\sigma_1, \ldots, \sigma_n)$: $\mathbf{E}(\sigma) = \sum_{ij \in E} K_{ij} \delta_{\sigma_i \sigma_j}$

where the *Kronecker symbol* $\delta_{ab} = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$

Potts Partition Function

$$Z(K) = \sum_{\sigma} \exp(\mathbf{E}(\sigma))$$

Potts Partition Function

$$Z(K) = \sum_{\sigma} \exp(\mathbf{E}(\sigma))$$

Letting $\nu_{ij} = \exp(K_{ij}) - 1 \geq 0$, we have

$$\exp(\mathbf{E}(\sigma)) \quad = \quad \prod_{ij \in E} \exp(K_{ij}\delta_{\sigma_i \sigma_j})$$

Potts Partition Function

$$Z(K) = \sum_{\sigma} \exp(\mathbf{E}(\sigma))$$

Letting $\nu_{ij} = \exp(K_{ij}) - 1 \geq 0$, we have

$$\begin{aligned} \exp(\mathbf{E}(\sigma)) &= \prod_{ij \in E} \exp(K_{ij}\delta_{\sigma_i \sigma_j}) \\ &= \prod_{ij \in E} \left(1 + (\exp(K_{ij}) - 1)\, \delta_{\sigma_i \sigma_j}\right) \end{aligned}$$

Potts Partition Function

$$Z(K) = \sum_{\sigma} \exp(\mathbf{E}(\sigma))$$

Letting $\nu_{ij} = \exp(K_{ij}) - 1 \geq 0$, we have

$$\begin{aligned}
\exp(\mathbf{E}(\sigma)) &= \prod_{ij \in E} \exp(K_{ij}\delta_{\sigma_i\sigma_j}) \\
&= \prod_{ij \in E} \left(1 + (\exp(K_{ij}) - 1)\,\delta_{\sigma_i\sigma_j}\right) \\
&= \sum_{F \in 2^E} \prod_{ij \in F} \nu_{ij}\,\delta_{\sigma_i\sigma_j}
\end{aligned}$$

Potts Partition Function (2)

$$Z(K) = \sum_{\sigma} \sum_{F \in 2^E} \prod_{ij \in F} \nu_{ij}\, \delta_{\sigma_i \sigma_j}$$

Potts Partition Function (2)

$$
\begin{aligned}
Z(K) &= \sum_{\sigma} \sum_{F \in 2^E} \prod_{ij \in F} \nu_{ij}\, \delta_{\sigma_i \sigma_j} \\
&= \sum_{F \in 2^E} \sum_{\sigma} \prod_{ij \in F} \nu_{ij}\, \delta_{\sigma_i \sigma_j}
\end{aligned}
$$

Potts Partition Function (2)

$$\begin{aligned} Z(K) &= \sum_{\sigma} \sum_{F \in 2^E} \prod_{ij \in F} \nu_{ij} \, \delta_{\sigma_i \sigma_j} \\ &= \sum_{F \in 2^E} \sum_{\sigma} \prod_{ij \in F} \nu_{ij} \, \delta_{\sigma_i \sigma_j} \\ &= \sum_{F \in 2^E} q^{nc(F)} \prod_{ij \in F} \nu_{ij} \end{aligned}$$

where $nc(F)$ is the number of connected components of $G_F = (V, F)$

Potts Partition Function (2)

$$\begin{aligned} Z(K) &= \sum_{\sigma} \sum_{F \in 2^E} \prod_{ij \in F} \nu_{ij}\, \delta_{\sigma_i \sigma_j} \\ &= \sum_{F \in 2^E} \sum_{\sigma} \prod_{ij \in F} \nu_{ij}\, \delta_{\sigma_i \sigma_j} \\ &= \sum_{F \in 2^E} q^{nc(F)} \prod_{ij \in F} \nu_{ij} \end{aligned}$$

where $nc(F)$ is the number of connected components of $G_F = (V, F)$

- Recall that nc is a *supermodular* function

Potts Partition Function (2)

$$
\begin{aligned}
Z(K) &= \sum_{\sigma} \sum_{F \in 2^E} \prod_{ij \in F} \nu_{ij}\, \delta_{\sigma_i \sigma_j} \\
&= \sum_{F \in 2^E} \sum_{\sigma} \prod_{ij \in F} \nu_{ij}\, \delta_{\sigma_i \sigma_j} \\
&= \sum_{F \in 2^E} q^{nc(F)} \prod_{ij \in F} \nu_{ij}
\end{aligned}
$$

where $nc(F)$ is the number of connected components of $G_F = (V, F)$

- Recall that nc is a *supermodular* function

Let $\alpha_{ij} = \log_q \nu_{ij}$ so $Z(K) = \sum_{F \in 2^E} q^{h(F)}$
where $h(F) = nc(F) + \sum_{ij \in F} \alpha_{ij}$

Asymptotics of Potts Partition Function

Asymptotics of Potts Partition Function

When q goes to infinity, $Z(K) \to Nq^{h^*}$ where N is the number of optimum sets F and

$$h^* = \max_{F \in 2^E} h(F) = \max_{F \in 2^E} \left(nc(F) + \sum_{ij \in F} \alpha_{ij} \right)$$

Asymptotics of Potts Partition Function

When q goes to infinity, $Z(K) \to Nq^{h^*}$ where N is the number of optimum sets F and

$$h^* = \max_{F \in 2^E} h(F) = \max_{F \in 2^E} \left(nc(F) + \sum_{ij \in F} \alpha_{ij} \right)$$

Since h is supermodular, finding the asymptotic exponent h^* is SFMin (where the ground set is the edge set E)

Asymptotics of Potts Partition Function

When q goes to infinity, $Z(K) \to Nq^{h^*}$ where N is the number of optimum sets F and

$$h^* = \max_{F \in 2^E} h(F) = \max_{F \in 2^E} \left(nc(F) + \sum_{ij \in F} \alpha_{ij} \right)$$

Since h is supermodular, finding the asymptotic exponent h^* is SFMin (where the ground set is the edge set E)

- Can we do better than general SFMin?

Two Simple Observations

Two Simple Observations

1. All $ij \in E$ with $\alpha_{ij} < 0$ may be eliminated
   (they cannot belong to any optimum subset)
   $\Rightarrow$ assume $\alpha \geq 0$

Two Simple Observations

1. All $ij \in E$ with $\alpha_{ij} < 0$ may be eliminated
   (they cannot belong to any optimum subset)
   $\Rightarrow$ assume $\alpha \geq 0$
2. Let F^* be an optimum subset and $P_1, \ldots, P_k$ the connected components of $G^* = (V, F^*)$,

Two Simple Observations

1. All $ij \in E$ with $\alpha_{ij} < 0$ may be eliminated
   (they cannot belong to any optimum subset)
   $\Rightarrow$ assume $\alpha \geq 0$
2. Let F^* be an optimum subset and $P_1, \ldots, P_k$ the connected components of $G^* = (V, F^*)$,

Two Simple Observations

1. All $ij \in E$ with $\alpha_{ij} < 0$ may be eliminated
   (they cannot belong to any optimum subset)
   $\Rightarrow$ assume $\alpha \geq 0$
2. Let F^* be an optimum subset and $P_1, \ldots, P_k$ the connected components of $G^* = (V, F^*)$,
   then we may add to F^* all edges in E within each P_i

Two Simple Observations

1. All $ij \in E$ with $\alpha_{ij} < 0$ may be eliminated
   (they cannot belong to any optimum subset)
   $\Rightarrow$ assume $\alpha \geq 0$
2. Let F^* be an optimum subset and $P_1, \ldots, P_k$ the connected components of $G^* = (V, F^*)$,
   then we may add to F^* all edges in E within each P_i

Therefore $h(F^*) = \alpha(E) - \sum_{i=1}^{k} f(P_i)$

where $f : 2^V \mapsto \mathbb{R}$, defined by $f(S) = \frac{1}{2}\left(\sum_{j \in S, k \notin S} \alpha_{jk}\right) - 1$,

is the cut function of the graph $G = (V, E)$ with edge "capacities" $\alpha \geq 0$, *minus the constant 1*

- so, $f(\emptyset) = -1 < 0$

A Faster Algorithm

A Faster Algorithm

Thus, finding $h*$ is equivalent to finding the value $f^D(V)$ of the Dilworth truncation of f

- Note: the ground set is now V, the node set

A Faster Algorithm

Thus, finding $h*$ is equivalent to finding the value $f^D(V)$ of the Dilworth truncation of f

- Note: the ground set is now V, the node set

The minimizations at each step of the Greedy Algorithm can be performed efficiently by *network flow techniques* (*minimum s, t-cuts* in an associated network)

A Faster Algorithm

Thus, finding $h*$ is equivalent to finding the value $f^D(V)$ of the Dilworth truncation of f

- Note: the ground set is now V, the node set

The minimizations at each step of the Greedy Algorithm can be performed efficiently by *network flow techniques* (*minimum s, t-cuts* in an associated network)

The running time is O($|V|^2\,|E|$)

- much faster than general SFMin on the old ground set $|E|$

Optimum Bipartition

Find a *bipartition* $\mathbf{P} = \{P_1, P_2\}$ of E with least total cost $f(\mathbf{P})$?

Optimum Bipartition

Find a *bipartition* $\mathbf{P} = \{P_1, P_2\}$ of E with least total cost $f(\mathbf{P})$?

Equivalently, find a *proper* subset S (i.e., $\emptyset \neq S \subset E$) which minimizes $f(S) + f(E \setminus S)$.

Optimum Bipartition

Find a *bipartition* $\mathbf{P} = \{P_1, P_2\}$ of E with least total cost $f(\mathbf{P})$?

Equivalently, find a *proper* subset S (i.e., $\emptyset \neq S \subset E$) which minimizes $f(S) + f(E \setminus S)$.

A set function $g : 2^E \mapsto \mathbb{R}$ is *symmetric* iff

$$g(S) = g(E \setminus S) \qquad \text{for all } S \subseteq E$$

Optimum Bipartition

Find a *bipartition* $\mathbf{P} = \{P_1, P_2\}$ of E with least total cost $f(\mathbf{P})$?

Equivalently, find a *proper* subset S (i.e., $\emptyset \neq S \subset E$) which minimizes $f(S) + f(E \setminus S)$.

A set function $g : 2^E \mapsto \mathbb{R}$ is *symmetric* iff

$$g(S) = g(E \setminus S) \qquad \text{for all } S \subseteq E$$

The function g_f defined by $g_f(S) = f(S) + f(E \setminus S)$ is:

- symmetric; and
- *submodular if f is submodular*

Sym-SFMin

If g is symmetric and submodular then, for all $S \subseteq E$

$$g(S) \quad = \quad 1/2\ (g(S) + g(E \setminus S))$$

Sym-SFMin

If g is symmetric and submodular then, for all $S \subseteq E$

$$\begin{aligned} g(S) &= 1/2\ (g(S) + g(E \setminus S)) \\ &\geq 1/2\ (g(E) + g(\emptyset)) \end{aligned}$$

Sym-SFMin

If g is symmetric and submodular then, for all $S \subseteq E$

$$\begin{aligned} g(S) &= 1/2\ (g(S) + g(E \setminus S)) \\ &\geq 1/2\ (g(E) + g(\emptyset)) \\ &= g(\emptyset) \end{aligned}$$

Sym-SFMin

If g is symmetric and submodular then, for all $S \subseteq E$

$$\begin{aligned} g(S) &= 1/2\ (g(S) + g(E \setminus S)) \\ &\geq 1/2\ (g(E) + g(\emptyset)) \\ &= g(\emptyset) \quad = \quad g(E) \end{aligned}$$

Sym-SFMin

If g is symmetric and submodular then, for all $S \subseteq E$

$$\begin{aligned} g(S) &= 1/2\ (g(S) + g(E \setminus S)) \\ &\geq 1/2\ (g(E) + g(\emptyset)) \\ &= g(\emptyset) \quad = \quad g(E) \end{aligned}$$

hence $\emptyset$, and also E, minimize g.

Sym-SFMin

If g is symmetric and submodular then, for all $S \subseteq E$

$$\begin{aligned} g(S) &= 1/2\ (g(S) + g(E \setminus S)) \\ &\geq 1/2\ (g(E) + g(\emptyset)) \\ &= g(\emptyset) \quad = \quad g(E) \end{aligned}$$

hence $\emptyset$, and also E, minimize g.

The Optimum Bipartition problem with submodular part costs, is equivalent to the **Symmetric Submodular Minimization problem (Sym-SFMin)**:

- given a symmetric submodular function $g : 2^E \mapsto \mathbb{R}$
- find a *proper* subset S of E which minimizes $g(S)$

Sym-SFMin and Decomposition

Sym-SFMin and Decomposition

Proposition Assume that f is normalized and submodular,

Sym-SFMin and Decomposition

Proposition Assume that f is normalized and submodular, and that there exists a proper subset $A \subset E$ such that $\breve{f}(A) = f(A) + f(\bar{A}) - f(E)$ satisfies $\breve{f}(A) = 0 = \breve{f}(\emptyset) = \breve{f}(E)$. where $\bar{A} = E \setminus A$.

Sym-SFMin and Decomposition

Proposition Assume that f is normalized and submodular, and that there exists a proper subset $A \subset E$ such that $\breve{f}(A) = f(A) + f(\bar{A}) - f(E)$ satisfies $\breve{f}(A) = 0 = \breve{f}(\emptyset) = \breve{f}(E)$. where $\bar{A} = E \setminus A$. Then f is decomposable as

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

Sym-SFMin and Decomposition

Proposition Assume that f is normalized and submodular, and that there exists a proper subset $A \subset E$ such that $\breve{f}(A) = f(A) + f(\bar{A}) - f(E)$ satisfies $\breve{f}(A) = 0 = \breve{f}(\emptyset) = \breve{f}(E)$. where $\bar{A} = E \setminus A$. Then f is decomposable as

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

Proof: Since f is normalized and submodular

$$f(B) = f\left((B \cap A) \cup (B \cap \bar{A})\right)$$

Sym-SFMin and Decomposition

Proposition Assume that f is normalized and submodular, and that there exists a proper subset $A \subset E$ such that $\breve{f}(A) = f(A) + f(\bar{A}) - f(E)$ satisfies $\breve{f}(A) = 0 = \breve{f}(\emptyset) = \breve{f}(E)$. where $\bar{A} = E \setminus A$. Then f is decomposable as

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

Proof: Since f is normalized and submodular

$$f(B) = f\left((B \cap A) \cup (B \cap \bar{A})\right) \leq f(B \cap A) + f(B \cap \bar{A})$$

Sym-SFMin and Decomposition

Proposition Assume that f is normalized and submodular, and that there exists a proper subset $A \subset E$ such that $\breve{f}(A) = f(A) + f(\bar{A}) - f(E)$ satisfies $\breve{f}(A) = 0 = \breve{f}(\emptyset) = \breve{f}(E)$. where $\bar{A} = E \setminus A$. Then f is decomposable as

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

Proof: Since f is normalized and submodular

$$f(B) = f\left((B \cap A) \cup (B \cap \bar{A})\right) \leq f(B \cap A) + f(B \cap \bar{A})$$

and

$$f(B) - f(B \cap A) - f(B \cap \bar{A}) \quad \geq \quad f(B \cup A) - f(A) - f(B \cap \bar{A})$$

Sym-SFMin and Decomposition

Proposition Assume that f is normalized and submodular, and that there exists a proper subset $A \subset E$ such that $\breve{f}(A) = f(A) + f(\bar{A}) - f(E)$ satisfies $\breve{f}(A) = 0 = \breve{f}(\emptyset) = \breve{f}(E)$. where $\bar{A} = E \setminus A$. Then f is decomposable as

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

Proof: Since f is normalized and submodular

$$f(B) = f\left((B \cap A) \cup (B \cap \bar{A})\right) \leq f(B \cap A) + f(B \cap \bar{A})$$

and

$$\begin{aligned} f(B) - f(B \cap A) - f(B \cap \bar{A}) &\geq f(B \cup A) - f(A) - f(B \cap \bar{A}) \\ &\geq f((B \cup A) \cup \bar{A}) - f(A) - f(\bar{A}) \end{aligned}$$

Sym-SFMin and Decomposition

Proposition Assume that f is normalized and submodular, and that there exists a proper subset $A \subset E$ such that $\breve{f}(A) = f(A) + f(\bar{A}) - f(E)$ satisfies $\breve{f}(A) = 0 = \breve{f}(\emptyset) = \breve{f}(E)$. where $\bar{A} = E \setminus A$. Then f is decomposable as

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

Proof: Since f is normalized and submodular

$$f(B) = f\left((B \cap A) \cup (B \cap \bar{A})\right) \leq f(B \cap A) + f(B \cap \bar{A})$$

and

$$\begin{aligned} f(B) - f(B \cap A) - f(B \cap \bar{A}) &\geq f(B \cup A) - f(A) - f(B \cap \bar{A}) \\ &\geq f((B \cup A) \cup \bar{A}) - f(A) - f(\bar{A}) \\ &= f(E) - f(A) - f(\bar{A}) \quad = \quad 0 \quad QED \end{aligned}$$

Separators

A proper subset A of E such that

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

is called a separator of f.

Separators

A proper subset A of E such that

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

is called a separator of f.

Example: Let X be a random vector indexed by E

Separators

A proper subset A of E such that

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

is called a separator of f.

Example: Let X be a random vector indexed by E
and let X_B denote the subvector indexed by any subset $B \subseteq E$

Separators

A proper subset A of E such that

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

is called a separator of f.

Example: Let X be a random vector indexed by E
and let X_B denote the subvector indexed by any subset $B \subseteq E$
If X_A and $X_{\bar{A}}$ are independent,

Separators

A proper subset A of E such that

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

is called a separator of f.

Example: Let X be a random vector indexed by E
and let X_B denote the subvector indexed by any subset $B \subseteq E$
If X_A and $X_{\bar{A}}$ are independent, then

- for every $B \subseteq A$ and $C \subseteq \bar{A}$, X_B and X_C are independent

Separators

A proper subset A of E such that

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

is called a separator of f.

Example: Let X be a random vector indexed by E
and let X_B denote the subvector indexed by any subset $B \subseteq E$
If X_A and $X_{\bar{A}}$ are independent, then

- for every $B \subseteq A$ and $C \subseteq \bar{A}$, X_B and X_C are independent
- Such a subset A is a separator of the entropy function for X

Separators

A proper subset A of E such that

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

is called a separator of f.

Example: Let X be a random vector indexed by E
and let X_B denote the subvector indexed by any subset $B \subseteq E$
If X_A and $X_{\bar{A}}$ are independent, then

- for every $B \subseteq A$ and $C \subseteq \bar{A}$, X_B and X_C are independent
- Such a subset A is a separator of the entropy function for X

The set of all separators of f is closed under intersection, union, and complementation

Separators

A proper subset A of E such that

$$f(B) = f(B \cap A) + f(B \cap \bar{A}) \quad \text{for all } B \subseteq E$$

is called a separator of f.

Example: Let X be a random vector indexed by E
and let X_B denote the subvector indexed by any subset $B \subseteq E$
If X_A and $X_{\bar{A}}$ are independent, then

- for every $B \subseteq A$ and $C \subseteq \bar{A}$, X_B and X_C are independent
- Such a subset A is a separator of the entropy function for X

The set of all separators of f is closed under intersection, union, and complementation

- Hence, the separators partition E

Pendent Pairs

Pendent Pairs

A pair $(u, v) \in E \times E$ $(u \neq v)$ is a *pendent pair* for (symmetric) set function g if

$$g(\{u\}) = \min\{g(S) : \forall S \subset E \text{ with } u \in S \text{ and } v \notin S\}$$

Pendent Pairs

A pair $(u, v) \in E \times E$ $(u \neq v)$ is a *pendent pair* for (symmetric) set function g if

$$g(\{u\}) = \min\{g(S) : \forall S \subset E \text{ with } u \in S \text{ and } v \notin S\}$$

A set $U \subset E$ *separates u and v* if

- $u \in U$ and $v \notin S$, or
- $u \notin U$ and $v \in S$

Pendent Pairs

A pair $(u, v) \in E \times E$ $(u \neq v)$ is a *pendent pair* for (symmetric) set function g if

$$g(\{u\}) = \min\{g(S) : \forall S \subset E \text{ with } u \in S \text{ and } v \notin S\}$$

A set $U \subset E$ *separates u and v* if

- $u \in U$ and $v \notin S$, or
- $u \notin U$ and $v \in S$

• equivalently, if $|S \cap \{u, v\}| = 1$

Pendent Pairs

A pair $(u, v) \in E \times E$ $(u \neq v)$ is a *pendent pair* for (symmetric) set function g if

$$g(\{u\}) = \min\{g(S) : \forall S \subset E \text{ with } u \in S \text{ and } v \notin S\}$$

A set $U \subset E$ *separates u and v* if

- $u \in U$ and $v \notin S$, or
- $u \notin U$ and $v \in S$

• equivalently, if $|S \cap \{u, v\}| = 1$

If (u, v) is a pendent pair for symmetric function g and S^* is a proper subset minimizing g then:

Pendent Pairs

A pair $(u, v) \in E \times E$ $(u \neq v)$ is a *pendent pair* for (symmetric) set function g if

$$g(\{u\}) = \min\{g(S) : \forall S \subset E \text{ with } u \in S \text{ and } v \notin S\}$$

A set $U \subset E$ *separates u and v* if

- $u \in U$ and $v \notin S$, or
- $u \notin U$ and $v \in S$

• equivalently, if $|S \cap \{u, v\}| = 1$

If (u, v) is a pendent pair for symmetric function g and S^* is a proper subset minimizing g then:

- *either* S^* separates u and v, and we may choose $S^* = \{u\}$

Pendent Pairs

A pair $(u, v) \in E \times E$ $(u \neq v)$ is a *pendent pair* for (symmetric) set function g if

$$g(\{u\}) = \min\{g(S) : \forall S \subset E \text{ with } u \in S \text{ and } v \notin S\}$$

A set $U \subset E$ *separates u and v* if

- $u \in U$ and $v \notin S$, or
- $u \notin U$ and $v \in S$

- equivalently, if $|S \cap \{u, v\}| = 1$

If (u, v) is a pendent pair for symmetric function g and S^* is a proper subset minimizing g then:

- *either* S^* separates u and v, and we may choose $S^* = \{u\}$
- *or else* u and v are on the same side of S^* and we may *contract* u and v into a single element

A Contraction Algorithm

(Q 1995, 1998; generalizing Nagamochi & Ibaraki, 1992)

A Contraction Algorithm

(Q 1995, 1998; generalizing Nagamochi & Ibaraki, 1992)

Assume we can efficiently find a pendent pair for any symmetric function in a class closed under contraction,

A Contraction Algorithm

(Q 1995, 1998; generalizing Nagamochi & Ibaraki, 1992)

Assume we can efficiently find a pendent pair for any symmetric function in a class closed under contraction, then we can find a proper subset S^* minimizing g after finding and contracting $n-1$ pendent pairs (u_1, v_1), (u_2, v_2), . . . , (u_{n-1}, v_{n-1}):

A Contraction Algorithm

(Q 1995, 1998; generalizing Nagamochi & Ibaraki, 1992)

Assume we can efficiently find a pendent pair for any symmetric function in a class closed under contraction, then we can find a proper subset S^* minimizing g after finding and contracting $n-1$ pendent pairs (u_1, v_1), (u_2, v_2), ..., (u_{n-1}, v_{n-1}):

Indeed, letting U_i be the original subset of E corresponding to u_i (for every iteration $i = 1, \ldots, n-1$), then

- *choose S^* as an U_i with least value $g(U_i)$*

A Contraction Algorithm

(Q 1995, 1998; generalizing Nagamochi & Ibaraki, 1992)

Assume we can efficiently find a pendent pair for any symmetric function in a class closed under contraction, then we can find a proper subset S^* minimizing g after finding and contracting $n-1$ pendent pairs (u_1, v_1), (u_2, v_2), ..., (u_{n-1}, v_{n-1}):

Indeed, letting U_i be the original subset of E corresponding to u_i (for every iteration $i = 1, \dots, n-1$), then

- *choose S^* as an U_i with least value $g(U_i)$*

Contracting u and v amounts to replacing

- the ground set E with $E_{u,v} = (E - u - v) + uv$
- the function g with $g_{u,v} : 2^{E_{u,v}} \mapsto \mathbb{R}$ defined by

$$g_{u,v}(S) = \begin{cases} g((S - uv) + u + v) & \text{if } uv \in S \\ g(S) & \text{otherwise} \end{cases}$$

A Contraction Algorithm

(Q 1995, 1998; generalizing Nagamochi & Ibaraki, 1992)

Assume we can efficiently find a pendent pair for any symmetric function in a class closed under contraction, then we can find a proper subset S^* minimizing g after finding and contracting $n-1$ pendent pairs (u_1, v_1), (u_2, v_2), ..., (u_{n-1}, v_{n-1}):

Indeed, letting U_i be the original subset of E corresponding to u_i (for every iteration $i = 1, \ldots, n-1$), then

- *choose S^* as an U_i with least value $g(U_i)$*

Contracting u and v amounts to replacing

- the ground set E with $E_{u,v} = (E - u - v) + uv$
- the function g with $g_{u,v} : 2^{E_{u,v}} \mapsto \mathbb{R}$ defined by

$$g_{u,v}(S) = \begin{cases} g((S - uv) + u + v) & \text{if } uv \in S \\ g(S) & \text{otherwise} \end{cases}$$

- If g is symmetric submodular then it remains so after contraction

A Contraction Algorithm

(Q 1995, 1998; generalizing Nagamochi & Ibaraki, 1992)

Assume we can efficiently find a pendent pair for any symmetric function in a class closed under contraction, then we can find a proper subset S^* minimizing g after finding and contracting $n-1$ pendent pairs (u_1, v_1), (u_2, v_2), ..., (u_{n-1}, v_{n-1}):

Indeed, letting U_i be the original subset of E corresponding to u_i (for every iteration $i = 1, \ldots, n-1$), then

- *choose S^* as an U_i with least value $g(U_i)$*

Contracting u and v amounts to replacing

- the ground set E with $E_{u,v} = (E - u - v) + uv$
- the function g with $g_{u,v} : 2^{E_{u,v}} \mapsto \mathbb{R}$ defined by

$$g_{u,v}(S) = \begin{cases} g((S - uv) + u + v) & \text{if } uv \in S \\ g(S) & \text{otherwise} \end{cases}$$

- If g is symmetric submodular then it remains so after contraction
- ...hence it remains to prove the existence of a pendent pair,

A Contraction Algorithm

(Q 1995, 1998; generalizing Nagamochi & Ibaraki, 1992)

Assume we can efficiently find a pendent pair for any symmetric function in a class closed under contraction, then we can find a proper subset S^* minimizing g after finding and contracting $n-1$ pendent pairs (u_1, v_1), (u_2, v_2), ..., (u_{n-1}, v_{n-1}):

Indeed, letting U_i be the original subset of E corresponding to u_i (for every iteration $i = 1, \ldots, n-1$), then

- *choose S^* as an U_i with least value $g(U_i)$*

Contracting u and v amounts to replacing

- the ground set E with $E_{u,v} = (E - u - v) + uv$
- the function g with $g_{u,v} : 2^{E_{u,v}} \mapsto \mathbb{R}$ defined by

$$g_{u,v}(S) = \begin{cases} g((S - uv) + u + v) & \text{if } uv \in S \\ g(S) & \text{otherwise} \end{cases}$$

- If g is symmetric submodular then it remains so after contraction
- ...hence it remains to prove the existence of a pendent pair, and to efficiently find one...

Finding a Pendent Pair

Finding a Pendent Pair

$E = (a_1, a_2, \ldots, a_n)$ is in Maximum Adjacency (MA) order if, for all $i = 1, \ldots, n-1$, a_{i+1} satisfies
$f(A_i + a_{i+1}) - f(\{a_{i+1}\}) = \min\{f(A_i + b) - f(\{b\}) : b \in E \setminus A_i\}$
where $A_i = \{a_1, \ldots, a_i\}$

Finding a Pendent Pair

$E = (a_1, a_2, \ldots, a_n)$ is in Maximum Adjacency (MA) order if, for all $i = 1, \ldots, n-1$, a_{i+1} satisfies
$f(A_i + a_{i+1}) - f(\{a_{i+1}\}) = \min\{f(A_i + b) - f(\{b\}) : b \in E \setminus A_i\}$
where $A_i = \{a_1, \ldots, a_i\}$

- a_1 is arbitrary, and then $a_2, \ldots, a_n$ are sequentially determined by this condition

Finding a Pendent Pair

$E = (a_1, a_2, \ldots, a_n)$ is in Maximum Adjacency (MA) order if, for all $i = 1, \ldots, n-1$, a_{i+1} satisfies
$f(A_i + a_{i+1}) - f(\{a_{i+1}\}) = \min\{f(A_i + b) - f(\{b\}) : b \in E \setminus A_i\}$
where $A_i = \{a_1, \ldots, a_i\}$

- a_1 is arbitrary, and then a_2, ..., a_n are sequentially determined by this condition

Lemma: If f is submodular, then for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$, $\quad f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$

Finding a Pendent Pair

$E = (a_1, a_2, \ldots, a_n)$ is in Maximum Adjacency (MA) order if, for all $i = 1, \ldots, n-1$, a_{i+1} satisfies
$f(A_i + a_{i+1}) - f(\{a_{i+1}\}) = \min\{f(A_i + b) - f(\{b\}) : b \in E \setminus A_i\}$
where $A_i = \{a_1, \ldots, a_i\}$

- a_1 is arbitrary, and then a_2, ..., a_n are sequentially determined by this condition

Lemma: If f is submodular, then for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$, $\quad f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$

- i.e., for every b not in A_i, $\{b\}$ is an optimum subset separating b from a_i for the symmetric function derived from the *restriction* of f to $A_i + b$

Finding a Pendent Pair

$E = (a_1, a_2, \ldots, a_n)$ is in Maximum Adjacency (MA) order if, for all $i = 1, \ldots, n-1$, a_{i+1} satisfies
$f(A_i + a_{i+1}) - f(\{a_{i+1}\}) = \min\{f(A_i + b) - f(\{b\}) : b \in E \setminus A_i\}$
where $A_i = \{a_1, \ldots, a_i\}$

- a_1 is arbitrary, and then $a_2, \ldots, a_n$ are sequentially determined by this condition

Lemma: If f is submodular, then for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$, $\quad f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$

- i.e., for every b not in A_i, $\{b\}$ is an optimum subset separating b from a_i for the symmetric function derived from the *restriction* of f to $A_i + b$

Corollary: If f is submodular, then (a_n, a_{n-1}) is a pendent pair for its symmetric function g_f

Pendent Pair Lemma

Proof of: $f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$
for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$

Pendent Pair Lemma

Proof of: $f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$
for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$

The inequality trivially holds for $i = 1$ (Why?)

Pendent Pair Lemma

Proof of: $f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$
for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$

The inequality trivially holds for $i = 1$ (Why?)
By induction, assume that it holds for all $i = 1, \ldots, k-1$

Pendent Pair Lemma

Proof of: $f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$
for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$

The inequality trivially holds for $i = 1$ (Why?)
By induction, assume that it holds for all $i = 1, \ldots, k-1$
Consider any $u \in E \setminus A_k$, and $S \subseteq A_{k-1}$

Pendent Pair Lemma

Proof of: $f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$
for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$

The inequality trivially holds for $i = 1$ (Why?)
By induction, assume that it holds for all $i = 1, \ldots, k-1$
Consider any $u \in E \setminus A_k$, and $S \subseteq A_{k-1}$
The choice of a_k implies $f(A_k) + f(u) \leq f(A_{k-1} + u) + f(a_k)$

Pendent Pair Lemma

Proof of: $f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$
for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$

The inequality trivially holds for $i = 1$ (Why?)
By induction, assume that it holds for all $i = 1, \ldots, k-1$
Consider any $u \in E \setminus A_k$, and $S \subseteq A_{k-1}$
The choice of a_k implies $f(A_k) + f(u) \leq f(A_{k-1} + u) + f(a_k)$

Let j be the smallest integer such that $S \subseteq A_{j-1}$

Pendent Pair Lemma

Proof of: $f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$
for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$

The inequality trivially holds for $i = 1$ (Why?)
By induction, assume that it holds for all $i = 1, \ldots, k-1$
Consider any $u \in E \setminus A_k$, and $S \subseteq A_{k-1}$
The choice of a_k implies $f(A_k) + f(u) \leq f(A_{k-1} + u) + f(a_k)$

Let j be the smallest integer such that $S \subseteq A_{j-1}$

- If $j = k$ then $a_{k-1} \in S$ and $A_{k-1} \setminus S \subseteq A_{k-2}$.

Pendent Pair Lemma

Proof of: $f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$
for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$

The inequality trivially holds for $i = 1$ (Why?)
By induction, assume that it holds for all $i = 1, \ldots, k-1$
Consider any $u \in E \setminus A_k$, and $S \subseteq A_{k-1}$
The choice of a_k implies $f(A_k) + f(u) \leq f(A_{k-1} + u) + f(a_k)$

Let j be the smallest integer such that $S \subseteq A_{j-1}$

- If $j = k$ then $a_{k-1} \in S$ and $A_{k-1} \setminus S \subseteq A_{k-2}$.
  Therefore,

$$f(A_k \setminus S) + f(S + u) \quad = \quad f((A_{k-1} \setminus S) + a_k) + f(S + u)$$

Pendent Pair Lemma

Proof of: $f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$
for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$

The inequality trivially holds for $i = 1$ (Why?)
By induction, assume that it holds for all $i = 1, \ldots, k-1$
Consider any $u \in E \setminus A_k$, and $S \subseteq A_{k-1}$
The choice of a_k implies $f(A_k) + f(u) \leq f(A_{k-1} + u) + f(a_k)$

Let j be the smallest integer such that $S \subseteq A_{j-1}$

- If $j = k$ then $a_{k-1} \in S$ and $A_{k-1} \setminus S \subseteq A_{k-2}$.
  Therefore,

$$\begin{aligned} f(A_k \setminus S) + f(S+u) &= f((A_{k-1} \setminus S) + a_k) + f(S+u) \\ &\geq f((A_{k-1}) + f(a_k) - f(S) + f(S+u) \end{aligned}$$

Pendent Pair Lemma

Proof of: $f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$
for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$

The inequality trivially holds for $i = 1$ (Why?)
By induction, assume that it holds for all $i = 1, \ldots, k-1$
Consider any $u \in E \setminus A_k$, and $S \subseteq A_{k-1}$
The choice of a_k implies $f(A_k) + f(u) \leq f(A_{k-1} + u) + f(a_k)$

Let j be the smallest integer such that $S \subseteq A_{j-1}$

- If $j = k$ then $a_{k-1} \in S$ and $A_{k-1} \setminus S \subseteq A_{k-2}$.
  Therefore,

$$
\begin{aligned}
f(A_k \setminus S) + f(S + u) &= f((A_{k-1} \setminus S) + a_k) + f(S + u) \\
&\geq f((A_{k-1}) + f(a_k) - f(S) + f(S + u) \\
&\geq f((A_{k-1} + u) + f(a_k)
\end{aligned}
$$

Pendent Pair Lemma

Proof of: $f(A_i) + f(b) \leq f(A_i \setminus S) + f(S + b)$
for all $i \in \{1, \ldots, n-1\}$, $b \in E \setminus A_i$ and $S \subseteq A_{i-1}$

The inequality trivially holds for $i = 1$ (Why?)
By induction, assume that it holds for all $i = 1, \ldots, k-1$
Consider any $u \in E \setminus A_k$, and $S \subseteq A_{k-1}$
The choice of a_k implies $f(A_k) + f(u) \leq f(A_{k-1} + u) + f(a_k)$

Let j be the smallest integer such that $S \subseteq A_{j-1}$

- If $j = k$ then $a_{k-1} \in S$ and $A_{k-1} \setminus S \subseteq A_{k-2}$.
  Therefore,

$$\begin{aligned} f(A_k \setminus S) + f(S+u) &= f((A_{k-1} \setminus S) + a_k) + f(S+u) \\ &\geq f((A_{k-1}) + f(a_k) - f(S) + f(S+u) \\ &\geq f((A_{k-1} + u) + f(a_k) \\ &\geq f(A_k) + f(u) \end{aligned}$$

Pendent Pair Lemma Proof (2)

- Else $j \leq k - 1$, thus $a_{j-1} \in S$ and none of $v_j, \ldots, v_k$ is in S

Pendent Pair Lemma Proof (2)

- Else $j \leq k-1$, thus $a_{j-1} \in S$ and none of $v_j, \ldots, v_k$ is in S
  Since $\{v_j, \ldots, v_k\} = A_k \setminus A_{j-1}$, we have,

$$f(A_k \setminus S) + f(S+u) \quad = \quad f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) + f(S+u)$$

Pendent Pair Lemma Proof (2)

- Else $j \leq k-1$, thus $a_{j-1} \in S$ and none of $v_j, \ldots, v_k$ is in S
  Since $\{v_j, \ldots, v_k\} = A_k \setminus A_{j-1}$, we have,

$$\begin{aligned} f(A_k \setminus S) + f(S+u) &= f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) + f(S+u) \\ &\geq f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) \\ &\qquad + f(A_j) - f(A_j \setminus S) + f(u) \end{aligned}$$

Pendent Pair Lemma Proof (2)

- Else $j \leq k-1$, thus $a_{j-1} \in S$ and none of $v_j, \ldots, v_k$ is in S
  Since $\{v_j, \ldots, v_k\} = A_k \setminus A_{j-1}$, we have,

$$\begin{aligned}
f(A_k \setminus S) + f(S+u) &= f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) + f(S+u) \\
&\geq f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) \\
&\qquad + f(A_j) - f(A_j \setminus S) + f(u) \\
&\geq f(A_k) + f(u) \qquad \text{QED}
\end{aligned}$$

Pendent Pair Lemma Proof (2)

- Else $j \leq k-1$, thus $a_{j-1} \in S$ and none of $v_j, \ldots, v_k$ is in S
  Since $\{v_j, \ldots, v_k\} = A_k \setminus A_{j-1}$, we have,

$$\begin{aligned} f(A_k \setminus S) + f(S+u) &= f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) + f(S+u) \\ &\geq f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) \\ &\qquad + f(A_j) - f(A_j \setminus S) + f(u) \\ &\geq f(A_k) + f(u) \end{aligned}$$

QED

The overall **Sym-SFMin algorithm** requires

- $n - i$ EO calls to find a_{i+1} (if we precompute all $f(\{u\})$)

Pendent Pair Lemma Proof (2)

- Else $j \leq k-1$, thus $a_{j-1} \in S$ and none of $v_j, \ldots, v_k$ is in S
  Since $\{v_j, \ldots, v_k\} = A_k \setminus A_{j-1}$, we have,

$$\begin{aligned} f(A_k \setminus S) + f(S+u) &= f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) + f(S+u) \\ &\geq f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) \\ &\qquad + f(A_j) - f(A_j \setminus S) + f(u) \\ &\geq f(A_k) + f(u) \qquad \text{QED} \end{aligned}$$

The overall **Sym-SFMin algorithm** requires

- $n - i$ EO calls to find a_{i+1} (if we precompute all $f(\{u\})$)
- $O(n^2)$ EO calls to find a MA order $a_1, a_2, \ldots, a_n$

Pendent Pair Lemma Proof (2)

- Else $j \le k-1$, thus $a_{j-1} \in S$ and none of $v_j, \dots, v_k$ is in S
  Since $\{v_j, \dots, v_k\} = A_k \setminus A_{j-1}$, we have,

$$\begin{aligned}
f(A_k \setminus S) + f(S+u) &= f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) + f(S+u) \\
&\ge f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) \\
&\qquad + f(A_j) - f(A_j \setminus S) + f(u) \\
&\ge f(A_k) + f(u) \qquad \text{QED}
\end{aligned}$$

The overall **Sym-SFMin algorithm** requires

- $n-i$ EO calls to find a_{i+1} (if we precompute all $f(\{u\})$)
- O(n^2) EO calls to find a MA order $a_1, a_2, \dots, a_n$
- O(n^3) EO calls to find a proper subset minimizing g_f
  and O(n^3) other operations

Pendent Pair Lemma Proof (2)

- Else $j \leq k-1$, thus $a_{j-1} \in S$ and none of $v_j, \ldots, v_k$ is in S
  Since $\{v_j, \ldots, v_k\} = A_k \setminus A_{j-1}$, we have,

$$\begin{aligned} f(A_k \setminus S) + f(S+u) &= f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) + f(S+u) \\ &\geq f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) \\ &\qquad + f(A_j) - f(A_j \setminus S) + f(u) \\ &\geq f(A_k) + f(u) \end{aligned}$$

QED

The overall **Sym-SFMin algorithm** requires

- $n-i$ EO calls to find a_{i+1} (if we precompute all $f(\{u\})$)
- O(n^2) EO calls to find a MA order $a_1, a_2, \ldots, a_n$
- O(n^3) EO calls to find a proper subset minimizing g_f
  and O(n^3) other operations

Pendent Pair Lemma Proof (2)

- Else $j \leq k-1$, thus $a_{j-1} \in S$ and none of $v_j, \ldots, v_k$ is in S Since $\{v_j, \ldots, v_k\} = A_k \setminus A_{j-1}$, we have,

$$\begin{aligned} f(A_k \setminus S) + f(S+u) &= f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) + f(S+u) \\ &\geq f\left((A_{j-1} \setminus S) \cup (A_k \setminus A_{j-1})\right) \\ &\qquad + f(A_j) - f(A_j \setminus S) + f(u) \\ &\geq f(A_k) + f(u) \qquad \text{QED} \end{aligned}$$

The overall **Sym-SFMin algorithm** requires

- $n-i$ EO calls to find a_{i+1} (if we precompute all $f(\{u\})$)
- O(n^2) EO calls to find a MA order $a_1, a_2, \ldots, a_n$
- O(n^3) EO calls to find a proper subset minimizing g_f and O(n^3) other operations

• Purely combinatorial, and faster than (current) general SFMin

Sym-SFMin: Examples and Extensions

Sym-SFMin: Examples and Extensions

Examples:

- Global MinCut in a Graph (Nagamochi & Ibaraki, 1992), where f is a graph cut function
  - $O(|V|^2 \log |V|)$ operations

Sym-SFMin: Examples and Extensions

Examples:

- Global MinCut in a Graph (Nagamochi & Ibaraki, 1992), where f is a graph cut function
  - O($|V|^2 \log |V|$) operations
- 2-Layer VLSI Circuit Design (Klimmek & Wagner, 1996), where f is a hypergraph cut function
  - O($|V|^2 \log |V| + |V||H|$) operations

Sym-SFMin: Examples and Extensions

Examples:

- Global MinCut in a Graph (Nagamochi & Ibaraki, 1992), where f is a graph cut function
  - $O(|V|^2 \log |V|)$ operations
- 2-Layer VLSI Circuit Design (Klimmek & Wagner, 1996), where f is a hypergraph cut function
  - $O(|V|^2 \log |V| + |V||H|)$ operations

Extensions: Minimizing

- posimodular functions (Nagamochi & Ibaraki, 1998), i.e., functions satisfying
  $f(A) + f(B) \geq f(A \setminus B) + f(B \setminus A)$ for all $A, B \subseteq V$

Sym-SFMin: Examples and Extensions

Examples:

- Global MinCut in a Graph (Nagamochi & Ibaraki, 1992), where f is a graph cut function
  - O($|V|^2 \log |V|$) operations
- 2-Layer VLSI Circuit Design (Klimmek & Wagner, 1996), where f is a hypergraph cut function
  - O($|V|^2 \log |V| + |V||H|$) operations

Extensions: Minimizing

- posimodular functions (Nagamochi & Ibaraki, 1998), i.e., functions satisfying
  $f(A) + f(B) \geq f(A \setminus B) + f(B \setminus A)$ for all $A, B \subseteq V$
- symmetric submodular function subject to hereditary family constraints (Goemans & Soto, 2013): $\min\{f(S) : S \in \mathcal{I}\}$ where $\mathcal{I} \subseteq 2^V$ satisfies, for all $A, B \subseteq V$,
  $\emptyset \neq A \subset B \in \mathcal{I} \quad \Rightarrow \quad A \in \mathcal{I}$

Optimum Proper Partition

(Baïou, Barahona & Mahjoub, 2000), motivated by a separation problem for certain connectivity constraints

Optimum Proper Partition

(Baïou, Barahona & Mahjoub, 2000), motivated by a separation problem for certain connectivity constraints

Find a *proper partition* $\mathbf{P}$ of E i.e., of size $|\mathbf{P}| \geq 2$, with minimum total cost $f(\mathbf{P})$?

Optimum Proper Partition

(Baïou, Barahona & Mahjoub, 2000), motivated by a separation problem for certain connectivity constraints

Find a *proper partition* $\mathbf{P}$ of E i.e., of size $|\mathbf{P}| \geq 2$, with minimum total cost $f(\mathbf{P})$?

For any set function f, if $\mathbf{P} = \{P_1, P_2, \ldots, P_k\}$ is optimum then:

Optimum Proper Partition

(Baïou, Barahona & Mahjoub, 2000), motivated by a separation problem for certain connectivity constraints

Find a *proper partition* $\mathbf{P}$ of E i.e., of size $|\mathbf{P}| \geq 2$, with minimum total cost $f(\mathbf{P})$?

For any set function f, if $\mathbf{P} = \{P_1, P_2, \ldots, P_k\}$ is optimum then:
- $\{P_1\}$ itself is an optimum partition of P_1

Optimum Proper Partition

(Baïou, Barahona & Mahjoub, 2000), motivated by a separation problem for certain connectivity constraints

Find a *proper partition* $\mathbf{P}$ of E i.e., of size $|\mathbf{P}| \geq 2$, with minimum total cost $f(\mathbf{P})$?

For any set function f, if $\mathbf{P} = \{P_1, P_2, \ldots, P_k\}$ is optimum then:

- $\{P_1\}$ itself is an optimum partition of P_1 and thus
  $f(P_1) = f^D(P_1)$, and

Optimum Proper Partition

(Baïou, Barahona & Mahjoub, 2000), motivated by a separation problem for certain connectivity constraints

Find a *proper partition* $\mathbf{P}$ of E i.e., of size $|\mathbf{P}| \geq 2$, with minimum total cost $f(\mathbf{P})$?

For any set function f, if $\mathbf{P} = \{P_1, P_2, \ldots, P_k\}$ is optimum then:

- $\{P_1\}$ itself is an optimum partition of P_1 and thus $f(P_1) = f^D(P_1)$, and
- $\{P_2, \ldots, P_k\}$ is an optimum partition of $E \setminus P_1$

Optimum Proper Partition

(Baïou, Barahona & Mahjoub, 2000), motivated by a separation problem for certain connectivity constraints

Find a *proper partition* $\mathbf{P}$ of E i.e., of size $|\mathbf{P}| \geq 2$, with minimum total cost $f(\mathbf{P})$?

For any set function f, if $\mathbf{P} = \{P_1, P_2, \ldots, P_k\}$ is optimum then:

- $\{P_1\}$ itself is an optimum partition of P_1 and thus
  $f(P_1) = f^D(P_1)$, and
- $\{P_2, \ldots, P_k\}$ is an optimum partition of $E \setminus P_1$ and thus
  $f(P_2) + \cdots + f(P_k) = f^D(E \setminus P_1)$

Optimum Proper Partition

(Baïou, Barahona & Mahjoub, 2000), motivated by a separation problem for certain connectivity constraints

Find a *proper partition* $\mathbf{P}$ of E i.e., of size $|\mathbf{P}| \geq 2$, with minimum total cost $f(\mathbf{P})$?

For any set function f, if $\mathbf{P} = \{P_1, P_2, \ldots, P_k\}$ is optimum then:

- $\{P_1\}$ itself is an optimum partition of P_1 and thus
  $f(P_1) = f^D(P_1)$, and
- $\{P_2, \ldots, P_k\}$ is an optimum partition of $E \setminus P_1$ and thus
  $f(P_2) + \cdots + f(P_k) = f^D(E \setminus P_1)$

Hence it suffices to find *an optimum bipartition of the Dilworth truncation f^D*

Optimum Proper Partition

(Baïou, Barahona & Mahjoub, 2000), motivated by a separation problem for certain connectivity constraints

Find a *proper partition* $\mathbf{P}$ of E i.e., of size $|\mathbf{P}| \geq 2$, with minimum total cost $f(\mathbf{P})$?

For any set function f, if $\mathbf{P} = \{P_1, P_2, \ldots, P_k\}$ is optimum then:

- $\{P_1\}$ itself is an optimum partition of P_1 and thus
  $f(P_1) = f^D(P_1)$, and
- $\{P_2, \ldots, P_k\}$ is an optimum partition of $E \setminus P_1$ and thus
  $f(P_2) + \cdots + f(P_k) = f^D(E \setminus P_1)$

Hence it suffices to find *an optimum bipartition of the Dilworth truncation f^D*

$\Rightarrow$ When f is submodular, O(n^4) EO's suffice

More Parts?

What is the computational complexity of finding an optimum k-way partition with submodular part cost function f (given by a value oracle)?

More Parts?

What is the computational complexity of finding an optimum k-way partition with submodular part cost function f (given by a value oracle)?

- NP-hard when k is part of the input, even for graph cut functions (Goldschmidt & Hochbaum, 1994)

More Parts?

What is the computational complexity of finding an optimum k-way partition with submodular part cost function f (given by a value oracle)?

- NP-hard when k is part of the input, even for graph cut functions (Goldschmidt & Hochbaum, 1994)
- When f is submodular (and normalized) an optimum 3-way partition can be found in polytime (Okumoto & al., 2012)

More Parts?

What is the computational complexity of finding an optimum k-way partition with submodular part cost function f (given by a value oracle)?

- NP-hard when k is part of the input, even for graph cut functions (Goldschmidt & Hochbaum, 1994)
- When f is submodular (and normalized) an optimum 3-way partition can be found in polytime (Okumoto & al., 2012)
- When f is symmetric and submodular an optimum 4-way partition can be found in polytime

More Parts?

What is the computational complexity of finding an optimum k-way partition with submodular part cost function f (given by a value oracle)?

- NP-hard when k is part of the input, even for graph cut functions (Goldschmidt & Hochbaum, 1994)
- When f is submodular (and normalized) an optimum 3-way partition can be found in polytime (Okumoto & al., 2012)
- When f is symmetric and submodular an optimum 4-way partition can be found in polytime
  - e.g., based on (Nagamochi & Ibaraki 2000) and using optimum submodular-costs 3-way cuts

More Parts?

What is the computational complexity of finding an optimum k-way partition with submodular part cost function f (given by a value oracle)?

- NP-hard when k is part of the input, even for graph cut functions (Goldschmidt & Hochbaum, 1994)
- When f is submodular (and normalized) an optimum 3-way partition can be found in polytime (Okumoto & al., 2012)
- When f is symmetric and submodular an optimum 4-way partition can be found in polytime
  - e.g., based on (Nagamochi & Ibaraki 2000) and using optimum submodular-costs 3-way cuts
- . . . see Thursday afternoon talk for related complexity results and open questions

Optimum k-Way Partitions: An Approximation Algorithm

Optimum k-Way Partitions: An Approximation Algorithm

Assume that g is symmetric, submodular and *nonnegative* ($g(S) \geq 0$ for all $S \subseteq E$)

Optimum k-Way Partitions: An Approximation Algorithm

Assume that g is symmetric, submodular and *nonnegative* ($g(S) \geq 0$ for all $S \subseteq E$)

Greedy Splitting Algorithm:

1. Let $\mathbf{P} = \{A_1\}$ where $A_1 = E$

Optimum k-Way Partitions: An Approximation Algorithm

Assume that g is symmetric, submodular and *nonnegative* ($g(S) \geq 0$ for all $S \subseteq E$)

Greedy Splitting Algorithm:

1. Let $\mathbf{P} = \{A_1\}$ where $A_1 = E$
2. For $j = 2, \ldots, k$
   - Let A_i ($i \in \{1, \ldots, j-1\}$) be a subset whose optimum bipartition $\{B_1, B_2\}$ least increases the total cost

Optimum k-Way Partitions: An Approximation Algorithm

Assume that g is symmetric, submodular and *nonnegative* ($g(S) \geq 0$ for all $S \subseteq E$)

Greedy Splitting Algorithm:

1. Let $\mathbf{P} = \{A_1\}$ where $A_1 = E$
2. For $j = 2, \ldots, k$
   - Let A_i ($i \in \{1, \ldots, j-1\}$) be a subset whose optimum bipartition $\{B_1, B_2\}$ least increases the total cost
   - Replace A_i in $\mathbf{P}$ with B_1 and add B_2 to $\mathbf{P}$

Optimum k-Way Partitions: An Approximation Algorithm

Assume that g is symmetric, submodular and *nonnegative* ($g(S) \geq 0$ for all $S \subseteq E$)

Greedy Splitting Algorithm:

1. Let $\mathbf{P} = \{A_1\}$ where $A_1 = E$
2. For $j = 2, \ldots, k$
   - Let A_i $(i \in \{1, \ldots, j-1\})$ be a subset whose optimum bipartition $\{B_1, B_2\}$ least increases the total cost
   - Replace A_i in $\mathbf{P}$ with B_1 and add B_2 to $\mathbf{P}$

This requires $2\,k - 3$ Sym-SFMin,
$\Rightarrow$ O($k\,n^3$) EO's, and O($k\,n^3$) other operations

Optimum k-Way Partitions: An Approximation Algorithm

Assume that g is symmetric, submodular and *nonnegative* ($g(S) \geq 0$ for all $S \subseteq E$)

Greedy Splitting Algorithm:

1. Let $\mathbf{P} = \{A_1\}$ where $A_1 = E$
2. For $j = 2, \ldots, k$
   - Let A_i $(i \in \{1, \ldots, j-1\})$ be a subset whose optimum bipartition $\{B_1, B_2\}$ least increases the total cost
   - Replace A_i in $\mathbf{P}$ with B_1 and add B_2 to $\mathbf{P}$

This requires $2\,k - 3$ Sym-SFMin,
$\Rightarrow$ O($k\,n^3$) EO's, and O($k\,n^3$) other operations

Theorem: [Q 1999; Zhao, Nagamochi & Ibaraki 2005]
If g is symmetric, submodular and nonnegative, then (for every $k \geq 2$) the Greedy Splitting Algorithm produces a k-way partition with total cost at most $2 - \frac{2}{k}$ times the optimum

Notes

Short Course on Submodular Functions
Part 2: Extensions and Related Problems
Session 2.B: SFmax

S. Thomas McCormick   Maurice Queyranne

Sauder School of Business, UBC

JPOC Summer School, June 2013

SFMax

Maximizing an oracle-given submodular function f

SFMax

Maximizing an oracle-given submodular function f

- easy if f is a polymatroid function, i.e., also monotone (nondecreasing)

SFMax

Maximizing an oracle-given submodular function f

- easy if f is a polymatroid function, i.e., also monotone (nondecreasing)
  - take the whole set E

SFMax

Maximizing an oracle-given submodular function f

- easy if f is a polymatroid function, i.e., also monotone (nondecreasing)
  - take the whole set E
  - (works for *any* monotone set function)

SFMax

Maximizing an oracle-given submodular function f

- easy if f is a polymatroid function, i.e., also monotone (nondecreasing)
  - take the whole set E
  - (works for *any* monotone set function)
- NP-hard for (non-monotone) submodular f

SFMax

Maximizing an oracle-given submodular function f

- easy if f is a polymatroid function, i.e., also monotone (nondecreasing)
  - take the whole set E
  - (works for *any* monotone set function)
- NP-hard for (non-monotone) submodular f
  - example: MaxCut

SFMax

Maximizing an oracle-given submodular function f

- easy if f is a polymatroid function, i.e., also monotone (nondecreasing)
  - take the whole set E
  - (works for *any* monotone set function)
- NP-hard for (non-monotone) submodular f
  - example: MaxCut

SFMax

Maximizing an oracle-given submodular function f

- easy if f is a polymatroid function, i.e., also monotone (nondecreasing)
  - take the whole set E
  - (works for *any* monotone set function)
- NP-hard for (non-monotone) submodular f
  - example: MaxCut

We shall be interested in approximation algorithms

SFMax

Maximizing an oracle-given submodular function f

- easy if f is a polymatroid function, i.e., also monotone (nondecreasing)
  - take the whole set E
  - (works for *any* monotone set function)
- NP-hard for (non-monotone) submodular f
  - example: MaxCut

We shall be interested in approximation algorithms
and two special cases:

1. Maximizing a polymatroid function subject to a cardinality constraint

SFMax

Maximizing an oracle-given submodular function f

- easy if f is a polymatroid function, i.e., also monotone (nondecreasing)
  - take the whole set E
  - (works for *any* monotone set function)
- NP-hard for (non-monotone) submodular f
  - example: MaxCut

We shall be interested in approximation algorithms
and two special cases:

1. Maximizing a polymatroid function subject to a cardinality constraint
2. Maximizing a (non-monotone, nonnegative) submodular function

Max k-Cover

Max k-Cover

Given m (feasible) subsets $E_1, \ldots, E_m$ of ground set E,

Max k-Cover

Given m (feasible) subsets $E_1, \ldots, E_m$ of ground set E, define the cover function $f : 2^V \mapsto \mathbb{R}$ as $f(S) = |\bigcup_{i \in S} E_i|$ for any $S \subseteq V = \{1, \ldots, m\}$

Max k-Cover

Given m (feasible) subsets E_1, ..., E_m of ground set E, define the cover function $f : 2^V \mapsto \mathbb{R}$ as $f(S) = |\bigcup_{i \in S} E_i|$ for any $S \subseteq V = \{1, \ldots, m\}$

- the total number of elements of E covered by the subsets in (indexed by) S

Max k-Cover

Given m (feasible) subsets $E_1, \ldots, E_m$ of ground set E, define the cover function $f : 2^V \mapsto \mathbb{R}$ as $f(S) = |\bigcup_{i \in S} E_i|$ for any $S \subseteq V = \{1, \ldots, m\}$

- the total number of elements of E covered by the subsets in (indexed by) S
- f is a polymatroid function (Why?)

Max k-Cover

Given m (feasible) subsets $E_1, \ldots, E_m$ of ground set E, define the cover function $f : 2^V \mapsto \mathbb{R}$ as $f(S) = |\bigcup_{i \in S} E_i|$ for any $S \subseteq V = \{1, \ldots, m\}$

- the total number of elements of E covered by the subsets in (indexed by) S
- f is a polymatroid function (Why?)

Max k-Cover

Given m (feasible) subsets $E_1, \ldots, E_m$ of ground set E,
define the cover function $f : 2^V \mapsto \mathbb{R}$ as $f(S) = |\bigcup_{i \in S} E_i|$
for any $S \subseteq V = \{1, \ldots, m\}$

- the total number of elements of E covered by the subsets in (indexed by) S
- f is a polymatroid function (Why?)

Given integer $k \in V$, Max k-Cover is $\max\{f(S) : S \subseteq V, \ |S| \leq k\}$

Max k-Cover

Given m (feasible) subsets $E_1, \ldots, E_m$ of ground set E, define the cover function $f : 2^V \mapsto \mathbb{R}$ as $f(S) = |\bigcup_{i \in S} E_i|$ for any $S \subseteq V = \{1, \ldots, m\}$

- the total number of elements of E covered by the subsets in (indexed by) S
- f is a polymatroid function (Why?)

Given integer $k \in V$, Max k-Cover is $\max\{f(S) : S \subseteq V,\ |S| \leq k\}$

- maximize the total number of elements covered by at most k subsets

Max k-Cover

Given m (feasible) subsets $E_1, \ldots, E_m$ of ground set E, define the cover function $f : 2^V \mapsto \mathbb{R}$ as $f(S) = |\bigcup_{i \in S} E_i|$ for any $S \subseteq V = \{1, \ldots, m\}$

- the total number of elements of E covered by the subsets in (indexed by) S
- f is a polymatroid function (Why?)

Given integer $k \in V$, Max k-Cover is $\max\{f(S) : S \subseteq V,\ |S| \leq k\}$

- maximize the total number of elements covered by at most k subsets
- equivalently: $\max\{f(S) : S \subseteq E,\ |S| = k\}$ (Why?)

Max k-Cover

Given m (feasible) subsets $E_1, \ldots, E_m$ of ground set E, define the cover function $f : 2^V \mapsto \mathbb{R}$ as $f(S) = |\bigcup_{i \in S} E_i|$ for any $S \subseteq V = \{1, \ldots, m\}$

- the total number of elements of E covered by the subsets in (indexed by) S
- f is a polymatroid function (Why?)

Given integer $k \in V$, Max k-Cover is $\max\{f(S) : S \subseteq V, \ |S| \leq k\}$

- maximize the total number of elements covered by at most k subsets
- equivalently: $\max\{f(S) : S \subseteq E, \ |S| = k\}$ (Why?)
- NP-hard

Max k-Cover

Given m (feasible) subsets $E_1, \ldots, E_m$ of ground set E, define the cover function $f : 2^V \mapsto \mathbb{R}$ as $f(S) = |\bigcup_{i \in S} E_i|$ for any $S \subseteq V = \{1, \ldots, m\}$

- the total number of elements of E covered by the subsets in (indexed by) S
- f is a polymatroid function (Why?)

Given integer $k \in V$, Max k-Cover is $\max\{f(S) : S \subseteq V, \ |S| \leq k\}$

- maximize the total number of elements covered by at most k subsets
- equivalently: $\max\{f(S) : S \subseteq E, \ |S| = k\}$ (Why?)
- NP-hard
- cannot be approximated within a ratio better (larger) than $1 - 1/e \approx 0.632$, unless P = NP (Feige 1998)

Cardinality-Constrained Polymatroid Maximization

Cardinality-Constrained Polymatroid Maximization

Given a normalized polymatroid function f on E

- hence f is nonnegative

Cardinality-Constrained Polymatroid Maximization

Given a normalized polymatroid function f on E

- hence f is nonnegative

Cardinality-Constrained Polymatroid Maximization

Given a normalized polymatroid function f on E

- hence f is nonnegative

and integer k, let $\mathsf{OPT}_k = \max\{f(S) : S \subseteq E,\ |S| \leq k\}$

Cardinality-Constrained Polymatroid Maximization

Given a normalized polymatroid function f on E

- hence f is nonnegative

and integer k, let $\text{OPT}_k = \max\{f(S) : S \subseteq E,\ |S| \leq k\}$

Greedy Algorithm

Cardinality-Constrained Polymatroid Maximization

Given a normalized polymatroid function f on E

- hence f is nonnegative

and integer k, let $\mathsf{OPT}_k = \max\{f(S) : S \subseteq E,\ |S| \leq k\}$

Greedy Algorithm

Starting with $S_0 = \emptyset$, repeat the following greedy step:

for $i = 0, \ldots, (k-1)$ let

$S_{i+1} = S_i + v_i$ where $v_i \in \arg\max_{u \in E \setminus S_i} f(S_i + u)$

Cardinality-Constrained Polymatroid Maximization

Given a normalized polymatroid function f on E

- hence f is nonnegative

and integer k, let $\mathsf{OPT}_k = \max\{f(S) : S \subseteq E,\ |S| \leq k\}$

Greedy Algorithm

Starting with $S_0 = \emptyset$, repeat the following greedy step:
for $i = 0, \ldots, (k-1)$ let

$$S_{i+1} = S_i + v_i \text{ where } v_i \in \arg\max_{u \in E \setminus S_i} f(S_i + u)$$

- equivalently, v_i yields the largest increment
  $f(u|S_i) = f(S_i + u) - f(S_i)$

Cardinality-Constrained Polymatroid Maximization

Given a normalized polymatroid function f on E

- hence f is nonnegative

and integer k, let $\mathsf{OPT}_k = \max\{f(S) : S \subseteq E,\ |S| \le k\}$

Greedy Algorithm

Starting with $S_0 = \emptyset$, repeat the following greedy step:
for $i = 0, \ldots, (k-1)$ let

$$S_{i+1} = S_i + v_i \text{ where } v_i \in \arg\max_{u \in E \setminus S_i} f(S_i + u)$$

- equivalently, v_i yields the largest increment $f(u|S_i) = f(S_i + u) - f(S_i)$

Cardinality-Constrained Polymatroid Maximization

Given a normalized polymatroid function f on E

- hence f is nonnegative

and integer k, let $\mathsf{OPT}_k = \max\{f(S) : S \subseteq E,\ |S| \leq k\}$

Greedy Algorithm

Starting with $S_0 = \emptyset$, repeat the following greedy step:
for $i = 0, \ldots, (k-1)$ let

$$S_{i+1} = S_i + v_i \text{ where } v_i \in \arg\max_{u \in E \setminus S_i} f(S_i + u)$$

- equivalently, v_i yields the largest increment $f(u|S_i) = f(S_i + u) - f(S_i)$

Theorem (Nemhauser, Wolsey & Fisher, 1978)

Cardinality-Constrained Polymatroid Maximization

Given a normalized polymatroid function f on E

- hence f is nonnegative

and integer k, let $\text{OPT}_k = \max\{f(S) : S \subseteq E,\ |S| \le k\}$

Greedy Algorithm

Starting with $S_0 = \emptyset$, repeat the following greedy step:
for $i = 0, \ldots, (k-1)$ let

$$S_{i+1} = S_i + v_i \text{ where } v_i \in \arg\max_{u \in E \setminus S_i} f(S_i + u)$$

- equivalently, v_i yields the largest increment $f(u|S_i) = f(S_i + u) - f(S_i)$

Theorem (Nemhauser, Wolsey & Fisher, 1978)
If f is a normalized polymatroid function then the Greedy Algorithm returns sets S_i with values

$$f(S_i) \ge (1 - 1/e)\text{OPT}_i \text{ for all } i = 0, \ldots, k$$

Cardinality-Constrained Polymatroid Maximization

Given a normalized polymatroid function f on E

- hence f is nonnegative

and integer k, let $\text{OPT}_k = \max\{f(S) : S \subseteq E,\ |S| \leq k\}$

Greedy Algorithm

Starting with $S_0 = \emptyset$, repeat the following greedy step:
for $i = 0, \ldots, (k-1)$ let

$$S_{i+1} = S_i + v_i \text{ where } v_i \in \arg\max_{u \in E \setminus S_i} f(S_i + u)$$

- equivalently, v_i yields the largest increment
  $f(u|S_i) = f(S_i + u) - f(S_i)$

Theorem (Nemhauser, Wolsey & Fisher, 1978)
If f is a normalized polymatroid function then the Greedy Algorithm returns sets S_i with values

$$f(S_i) \geq (1 - 1/e)\text{OPT}_i \text{ for all } i = 0, \ldots, k$$

- since Max k-cover is a special case, by Feige's result this is the best possible approximation guarantee (unless P = NP)

Cardinality-Constrained Polymatroid Maximization

Given a normalized polymatroid function f on E

- hence f is nonnegative

and integer k, let $\mathsf{OPT}_k = \max\{f(S) : S \subseteq E,\ |S| \leq k\}$

Greedy Algorithm

Starting with $S_0 = \emptyset$, repeat the following greedy step:
for $i = 0, \ldots, (k-1)$ let

$$S_{i+1} = S_i + v_i \text{ where } v_i \in \arg\max_{u \in E \setminus S_i} f(S_i + u)$$

- equivalently, v_i yields the largest increment
  $f(u|S_i) = f(S_i + u) - f(S_i)$

Theorem (Nemhauser, Wolsey & Fisher, 1978)
If f is a normalized polymatroid function then the Greedy Algorithm returns sets S_i with values

$$f(S_i) \geq (1 - 1/e)\mathsf{OPT}_i \text{ for all } i = 0, \ldots, k$$

- since Max k-cover is a special case, by Feige's result this is the best possible approximation guarantee (unless P = NP)
- this guarantee holds at every step i (relative to OPT_i)

$(1 - 1/e)$-Approximation: Proof Idea

We shall prove a more general result, but here is where the $(1 - 1/e)$ factor comes from:

$(1 - 1/e)$-Approximation: Proof Idea

We shall prove a more general result, but here is where the $(1 - 1/e)$ factor comes from:

- Let $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$

$(1 - 1/e)$-Approximation: Proof Idea

We shall prove a more general result, but here is where the $(1 - 1/e)$ factor comes from:

- Let $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- By submodularity and the greedy step, we will prove that there exist $u \in S^* \setminus S_i$ such that the increment
$$f(u|S_i) \geq \frac{1}{k}\left(\mathsf{OPT}_k - f(S_i)\right)$$

$(1 - 1/e)$-Approximation: Proof Idea

We shall prove a more general result, but here is where the $(1 - 1/e)$ factor comes from:

- Let $S^* \in \arg\max\{f(S) : S \subseteq E, \ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- By submodularity and the greedy step, we will prove that there exist $u \in S^* \setminus S_i$ such that the increment
  $$f(u|S_i) \geq \frac{1}{k}\left(\mathsf{OPT}_k - f(S_i)\right)$$
  - $(\mathsf{OPT}_k - f(S_i))$ is the current (absolute) gap at iteration S_i (relative to the size-k optimum)

$(1-1/e)$-Approximation: Proof Idea

We shall prove a more general result, but here is where the $(1-1/e)$ factor comes from:

- Let $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- By submodularity and the greedy step, we will prove that there exist $u \in S^* \setminus S_i$ such that the increment
  $$f(u|S_i) \geq \frac{1}{k}\left(\mathsf{OPT}_k - f(S_i)\right)$$
  - $(\mathsf{OPT}_k - f(S_i))$ is the current (absolute) gap at iteration S_i (relative to the size-k optimum)
- The increment at greedy step i is at least that large, hence
  $$f(S_{i+1}) - f(S_i) \geq \frac{1}{k}\left(\mathsf{OPT}_k - f(S_i)\right)$$

$(1-1/e)$-Approximation: Proof Idea

We shall prove a more general result, but here is where the $(1-1/e)$ factor comes from:

- Let $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- By submodularity and the greedy step, we will prove that there exist $u \in S^* \setminus S_i$ such that the increment
  $$f(u|S_i) \geq \frac{1}{k}\left(\mathsf{OPT}_k - f(S_i)\right)$$
  - $(\mathsf{OPT}_k - f(S_i))$ is the current (absolute) gap at iteration S_i (relative to the size-k optimum)
- The increment at greedy step i is at least that large, hence
  $$f(S_{i+1}) - f(S_i) \geq \frac{1}{k}\left(\mathsf{OPT}_k - f(S_i)\right)$$
- Equivalently, $\mathsf{OPT}_k - f(S_{i+1}) \leq \left(1 - \frac{1}{k}\right)\left(\mathsf{OPT}_k - f(S_i)\right)$

$(1 - 1/e)$-Approximation: Proof Idea

We shall prove a more general result, but here is where the $(1 - 1/e)$ factor comes from:

- Let $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \text{OPT}_k$
- By submodularity and the greedy step, we will prove that there exist $u \in S^* \setminus S_i$ such that the increment
$$f(u|S_i) \geq \frac{1}{k}\left(\text{OPT}_k - f(S_i)\right)$$
  - $(\text{OPT}_k - f(S_i))$ is the current (absolute) gap at iteration S_i (relative to the size-k optimum)
- The increment at greedy step i is at least that large, hence
$$f(S_{i+1}) - f(S_i) \geq \frac{1}{k}\left(\text{OPT}_k - f(S_i)\right)$$
- Equivalently, $\text{OPT}_k - f(S_{i+1}) \leq \left(1 - \frac{1}{k}\right)\left(\text{OPT}_k - f(S_i)\right)$

$(1 - 1/e)$-Approximation: Proof Idea

We shall prove a more general result, but here is where the $(1 - 1/e)$ factor comes from:

- Let $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- By submodularity and the greedy step, we will prove that there exist $u \in S^* \setminus S_i$ such that the increment
  $$f(u|S_i) \geq \frac{1}{k}\left(\mathsf{OPT}_k - f(S_i)\right)$$
  - $(\mathsf{OPT}_k - f(S_i))$ is the current (absolute) gap at iteration S_i (relative to the size-k optimum)
- The increment at greedy step i is at least that large, hence
  $$f(S_{i+1}) - f(S_i) \geq \frac{1}{k}\left(\mathsf{OPT}_k - f(S_i)\right)$$
- Equivalently, $\mathsf{OPT}_k - f(S_{i+1}) \leq \left(1 - \frac{1}{k}\right)\left(\mathsf{OPT}_k - f(S_i)\right)$ i.e., the gap decreases by a factor $\geq (1 - 1/k)$ at each step
- Since the initial gap $\mathsf{OPT}_k - f(S_0) \leq \mathsf{OPT}_k$, the final gap
  $$\mathsf{OPT}_k - S_k \leq \left(1 - \frac{1}{k}\right)^k \mathsf{OPT}_k$$

$(1 - 1/e)$-Approximation: Proof Idea

We shall prove a more general result, but here is where the $(1 - 1/e)$ factor comes from:

- Let $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- By submodularity and the greedy step, we will prove that there exist $u \in S^* \setminus S_i$ such that the increment
  $$f(u|S_i) \geq \frac{1}{k}\left(\mathsf{OPT}_k - f(S_i)\right)$$
  - $(\mathsf{OPT}_k - f(S_i))$ is the current (absolute) gap at iteration S_i (relative to the size-k optimum)
- The increment at greedy step i is at least that large, hence
  $$f(S_{i+1}) - f(S_i) \geq \frac{1}{k}\left(\mathsf{OPT}_k - f(S_i)\right)$$
- Equivalently, $\mathsf{OPT}_k - f(S_{i+1}) \leq \left(1 - \frac{1}{k}\right)\left(\mathsf{OPT}_k - f(S_i)\right)$ i.e., the gap decreases by a factor $\geq (1 - 1/k)$ at each step
- Since the initial gap $\mathsf{OPT}_k - f(S_0) \leq \mathsf{OPT}_k$, the final gap
  $$\mathsf{OPT}_k - S_k \leq \left(1 - \frac{1}{k}\right)^k \mathsf{OPT}_k$$

$(1 - 1/e)$-Approximation: Proof Idea

We shall prove a more general result, but here is where the $(1 - 1/e)$ factor comes from:

- Let $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- By submodularity and the greedy step, we will prove that there exist $u \in S^* \setminus S_i$ such that the increment
$$f(u|S_i) \geq \frac{1}{k}\left(\mathsf{OPT}_k - f(S_i)\right)$$
  - $(\mathsf{OPT}_k - f(S_i))$ is the current (absolute) gap at iteration S_i (relative to the size-k optimum)
- The increment at greedy step i is at least that large, hence
$$f(S_{i+1}) - f(S_i) \geq \frac{1}{k}\left(\mathsf{OPT}_k - f(S_i)\right)$$
- Equivalently, $\mathsf{OPT}_k - f(S_{i+1}) \leq \left(1 - \frac{1}{k}\right)\left(\mathsf{OPT}_k - f(S_i)\right)$ i.e., the gap decreases by a factor $\geq (1 - 1/k)$ at each step
- Since the initial gap $\mathsf{OPT}_k - f(S_0) \leq \mathsf{OPT}_k$, the final gap
$$\mathsf{OPT}_k - S_k \leq \left(1 - \frac{1}{k}\right)^k \mathsf{OPT}_k \leq \frac{1}{e}\,\mathsf{OPT}_k$$

$(1 - 1/e)$-Approximation: Proof Idea

We shall prove a more general result, but here is where the $(1 - 1/e)$ factor comes from:

- Let $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathrm{OPT}_k$
- By submodularity and the greedy step, we will prove that there exist $u \in S^* \setminus S_i$ such that the increment
  $$f(u|S_i) \geq \frac{1}{k}\left(\mathrm{OPT}_k - f(S_i)\right)$$
  - $(\mathrm{OPT}_k - f(S_i))$ is the current (absolute) gap at iteration S_i (relative to the size-k optimum)
- The increment at greedy step i is at least that large, hence
  $$f(S_{i+1}) - f(S_i) \geq \frac{1}{k}\left(\mathrm{OPT}_k - f(S_i)\right)$$
- Equivalently, $\mathrm{OPT}_k - f(S_{i+1}) \leq \left(1 - \frac{1}{k}\right)\left(\mathrm{OPT}_k - f(S_i)\right)$ i.e., the gap decreases by a factor $\geq (1 - 1/k)$ at each step
- Since the initial gap $\mathrm{OPT}_k - f(S_0) \leq \mathrm{OPT}_k$, the final gap
  $$\mathrm{OPT}_k - S_k \leq \left(1 - \frac{1}{k}\right)^k \mathrm{OPT}_k \leq \frac{1}{e}\,\mathrm{OPT}_k$$
- and therefore $f(S_k) \geq \left(1 - \frac{1}{e}\right)\mathrm{OPT}_k > 0.632\,\mathrm{OPT}_k$

A More General Approximation Guarantee

A More General Approximation Guarantee

Theorem (Nemhauser, Wolsey & Fisher, 1978)

A More General Approximation Guarantee

Theorem (Nemhauser, Wolsey & Fisher, 1978)
If f is a normalized polymatroid function then the Greedy Algorithm returns sets S_i $(i = 1, \ldots, n)$ with values

$$f(S_i) \geq (1 - e^{-i/k})\mathsf{OPT}_i \text{ for all } i = 0, \ldots, n$$

A More General Approximation Guarantee

Theorem (Nemhauser, Wolsey & Fisher, 1978)
If f is a normalized polymatroid function then the Greedy Algorithm returns sets S_i $(i = 1, \ldots, n)$ with values

$$f(S_i) \geq (1 - e^{-i/k})\text{OPT}_i \text{ for all } i = 0, \ldots, n$$

- the approximation guarantee improves with the iteration (obviously – why?)

A More General Approximation Guarantee

Theorem (Nemhauser, Wolsey & Fisher, 1978)
If f is a normalized polymatroid function then the Greedy Algorithm returns sets S_i $(i = 1, \ldots, n)$ with values

$$f(S_i) \geq (1 - e^{-i/k})\text{OPT}_i \text{ for all } i = 0, \ldots, n$$

- the approximation guarantee improves with the iteration (obviously – why?)
- values $i > k$ may be interpreted as *resource augmentation*

A More General Approximation Guarantee

Theorem (Nemhauser, Wolsey & Fisher, 1978)
If f is a normalized polymatroid function then the Greedy Algorithm returns sets S_i $(i = 1, \ldots, n)$ with values

$$f(S_i) \geq (1 - e^{-i/k})\text{OPT}_i \text{ for all } i = 0, \ldots, n$$

- the approximation guarantee improves with the iteration (obviously – why?)
- values $i > k$ may be interpreted as *resource augmentation*
- what if we want to guarantee at least $0.95\,\text{OPT}_k$?

A More General Approximation Guarantee

Theorem (Nemhauser, Wolsey & Fisher, 1978)
If f is a normalized polymatroid function then the Greedy Algorithm returns sets S_i $(i = 1, \ldots, n)$ with values

$$f(S_i) \geq (1 - e^{-i/k})\text{OPT}_i \text{ for all } i = 0, \ldots, n$$

- the approximation guarantee improves with the iteration (obviously – why?)
- values $i > k$ may be interpreted as *resource augmentation*
- what if we want to guarantee at least $0.95\,\text{OPT}_k$?
  - $0.95 = 1 - e^{i/k}$ gives $i = \lceil -k \ln(1 - 0.95\rceil \leq 4k$

A More General Approximation Guarantee

Theorem (Nemhauser, Wolsey & Fisher, 1978)
If f is a normalized polymatroid function then the Greedy Algorithm returns sets S_i $(i = 1, \ldots, n)$ with values

$$f(S_i) \geq (1 - e^{-i/k})\text{OPT}_i \text{ for all } i = 0, \ldots, n$$

- the approximation guarantee improves with the iteration (obviously – why?)
- values $i > k$ may be interpreted as *resource augmentation*
- what if we want to guarantee at least $0.95\,\text{OPT}_k$?
  - $0.95 = 1 - e^{i/k}$ gives $i = \lceil -k \ln(1 - 0.95\rceil \leq 4k$
  - (and for 0.999, $\lceil -k \ln(1 - 0.999\rceil = 7$)

A More General Approximation Guarantee

Theorem (Nemhauser, Wolsey & Fisher, 1978)
If f is a normalized polymatroid function then the Greedy Algorithm returns sets S_i $(i = 1, \ldots, n)$ with values

$$f(S_i) \geq (1 - e^{-i/k})\text{OPT}_i \text{ for all } i = 0, \ldots, n$$

- the approximation guarantee improves with the iteration (obviously – why?)
- values $i > k$ may be interpreted as *resource augmentation*
- what if we want to guarantee at least $0.95\,\text{OPT}_k$?
  - $0.95 = 1 - e^{i/k}$ gives $i = \lceil -k \ln(1 - 0.95\rceil \leq 4k$
  - (and for 0.999, $\lceil -k \ln(1 - 0.999\rceil = 7$)
- typical practical performance is much better

Proof of Greedy Performance

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$
- Let $(v_1, \ldots, v_\ell)$ be the greedy order chosen by the algorithm

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$
- Let $(v_1, \ldots, v_\ell)$ be the greedy order chosen by the algorithm
- Then, for all $i < \ell$, the gap

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$
- Let $(v_1, \ldots, v_\ell)$ be the greedy order chosen by the algorithm
- Then, for all $i < \ell$, the gap

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \text{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$
- Let $(v_1, \ldots, v_\ell)$ be the greedy order chosen by the algorithm
- Then, for all $i < \ell$, the gap

$$\delta_i \quad = \quad f(S^*) - f(S_i)$$

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$
- Let $(v_1, \ldots, v_\ell)$ be the greedy order chosen by the algorithm
- Then, for all $i < \ell$, the gap

$$\delta_i \quad = \quad f(S^*) - f(S_i) \quad \leq \quad f(S^* \cup S_i) - f(S_i)$$

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$
- Let $(v_1, \ldots, v_\ell)$ be the greedy order chosen by the algorithm
- Then, for all $i < \ell$, the gap

$$\begin{aligned}
\delta_i &= f(S^*) - f(S_i) \quad \leq \quad f(S^* \cup S_i) - f(S_i) \\
&= \sum_{j=1}^{k} f(v_j^* | S_i + v_1^* + \cdots + v_{j-1}^*)
\end{aligned}$$

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$
- Let $(v_1, \ldots, v_\ell)$ be the greedy order chosen by the algorithm
- Then, for all $i < \ell$, the gap

$$\begin{aligned}
\delta_i &= f(S^*) - f(S_i) \leq f(S^* \cup S_i) - f(S_i) \\
&= \sum_{j=1}^{k} f(v_j^* | S_i + v_1^* + \cdots + v_{j-1}^*) \leq \sum_{j=1}^{k} f(v_j^* | S_i)
\end{aligned}$$

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathrm{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$
- Let $(v_1, \ldots, v_\ell)$ be the greedy order chosen by the algorithm
- Then, for all $i < \ell$, the gap

$$
\begin{aligned}
\delta_i &= f(S^*) - f(S_i) \quad \leq \quad f(S^* \cup S_i) - f(S_i) \\
&= \sum_{j=1}^{k} f(v_j^* | S_i + v_1^* + \cdots + v_{j-1}^*) \quad \leq \quad \sum_{j=1}^{k} f(v_j^* | S_i) \\
&\leq k\, f(v_i | S_i)
\end{aligned}
$$

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$
- Let $(v_1, \ldots, v_\ell)$ be the greedy order chosen by the algorithm
- Then, for all $i < \ell$, the gap

$$\begin{aligned}
\delta_i &= f(S^*) - f(S_i) \leq f(S^* \cup S_i) - f(S_i) \\
&= \sum_{j=1}^{k} f(v_j^* | S_i + v_1^* + \cdots + v_{j-1}^*) \leq \sum_{j=1}^{k} f(v_j^* | S_i) \\
&\leq k\, f(v_i | S_i) = k\, (f(S_{i+1}) - f(S_i))
\end{aligned}$$

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$
- Let $(v_1, \ldots, v_\ell)$ be the greedy order chosen by the algorithm
- Then, for all $i < \ell$, the gap

$$\begin{aligned}
\delta_i &= f(S^*) - f(S_i) \leq f(S^* \cup S_i) - f(S_i) \\
&= \sum_{j=1}^{k} f(v_j^* | S_i + v_1^* + \cdots + v_{j-1}^*) \leq \sum_{j=1}^{k} f(v_j^* | S_i) \\
&\leq k\, f(v_i | S_i) = k\, (f(S_{i+1}) - f(S_i)) = k\, (\delta_i - \delta_{i+1})
\end{aligned}$$

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$
- Let $(v_1, \ldots, v_\ell)$ be the greedy order chosen by the algorithm
- Then, for all $i < \ell$, the gap

$$
\begin{aligned}
\delta_i &= f(S^*) - f(S_i) \leq f(S^* \cup S_i) - f(S_i) \\
&= \sum_{j=1}^{k} f(v_j^* | S_i + v_1^* + \cdots + v_{j-1}^*) \leq \sum_{j=1}^{k} f(v_j^* | S_i) \\
&\leq k\, f(v_i | S_i) = k\,(f(S_{i+1}) - f(S_i)) = k\,(\delta_i - \delta_{i+1})
\end{aligned}
$$

implying $\qquad \delta_{i+1} \leq (1 - \frac{1}{k})\delta_i$

Proof of Greedy Performance

- Fix ℓ (size of greedy solution) and k (size of optimal set)
- Fix $S^* \in \arg\max\{f(S) : S \subseteq E,\ |S| \leq k\}$, so $f(S^*) = \mathsf{OPT}_k$
- Assume w.l.o.g., that $|S^*| = k$ and let $S^* = \{v_1^*, \ldots, v_k^*\}$
- Let $(v_1, \ldots, v_\ell)$ be the greedy order chosen by the algorithm
- Then, for all $i < \ell$, the gap

$$\begin{aligned}
\delta_i &= f(S^*) - f(S_i) \quad \leq \quad f(S^* \cup S_i) - f(S_i) \\
&= \sum_{j=1}^{k} f(v_j^* | S_i + v_1^* + \cdots + v_{j-1}^*) \quad \leq \quad \sum_{j=1}^{k} f(v_j^* | S_i) \\
&\leq k\, f(v_i | S_i) \quad = \quad k\, (f(S_{i+1}) - f(S_i)) \quad = \quad k\,(\delta_i - \delta_{i+1})
\end{aligned}$$

implying $\quad \delta_{i+1} \leq (1 - \frac{1}{k})\delta_i \quad$ and thus

$\delta_\ell \ \leq\ (1 - \frac{1}{k})^\ell\, \delta_0 \ \leq\ (1 - \frac{1}{k})^\ell\, \mathsf{OPT}_k \ \leq\ e^{-\ell/k}\, \mathsf{OPT}_k$ QED

Greedy Algorithm: Running Time

Greedy Algorithm: Running Time

- Greedy computes a new maximum $n = |V|$ times

Greedy Algorithm: Running Time

- Greedy computes a new maximum $n = |V|$ times
- each maximum computation requires O(n) comparisons

Greedy Algorithm: Running Time

- Greedy computes a new maximum $n = |V|$ times
- each maximum computation requires $O(n)$ comparisons

Greedy Algorithm: Running Time

- Greedy computes a new maximum $n = |V|$ times
- each maximum computation requires $\mathsf{O}(n)$ comparisons

hence $\mathsf{O}(n^2)$ time overall

Greedy Algorithm: Running Time

- Greedy computes a new maximum $n = |V|$ times
- each maximum computation requires $\mathrm{O}(n)$ comparisons

hence $\mathrm{O}(n^2)$ time overall

This is not good enough for very large practical instances

- large water networks with many contamination scenarios; social networks; selecting blogs of greatest influence; document summarization; etc.

and can be made (much) faster by a simple trick, also based on submodularity:

Greedy Algorithm: Running Time

- Greedy computes a new maximum $n = |V|$ times
- each maximum computation requires $\mathrm{O}(n)$ comparisons

hence $\mathrm{O}(n^2)$ time overall

This is not good enough for very large practical instances

- large water networks with many contamination scenarios; social networks; selecting blogs of greatest influence; document summarization; etc.

and can be made (much) faster by a simple trick, also based on submodularity:

Greedy Algorithm: Running Time

- Greedy computes a new maximum $n = |V|$ times
- each maximum computation requires $\mathrm{O}(n)$ comparisons

hence $\mathrm{O}(n^2)$ time overall

This is not good enough for very large practical instances

- large water networks with many contamination scenarios; social networks; selecting blogs of greatest influence; document summarization; etc.

and can be made (much) faster by a simple trick, also based on submodularity:

Minoux's Accelerated Greedy (aka, **Lazy Selection**)

Greedy Algorithm: Running Time

- Greedy computes a new maximum $n = |V|$ times
- each maximum computation requires $\mathrm{O}(n)$ comparisons

hence $\mathrm{O}(n^2)$ time overall

This is not good enough for very large practical instances

- large water networks with many contamination scenarios; social networks; selecting blogs of greatest influence; document summarization; etc.

and can be made (much) faster by a simple trick, also based on submodularity:

Minoux's Accelerated Greedy (aka, **Lazy Selection**)
Idea: to reduce the number of function evaluations and of comparisons, store upper bounds α_v on the increments $f(v|S_i)$ in a priority queue, and only update α_v when element v is examined

Minoux's Accelerated Greedy

- Store the initial increments $\alpha_v = f(v|S_0)$ in a priority queue, and the iteration index $\beta_v = 0$ at which it was least updated

Minoux's Accelerated Greedy

- Store the initial increments $\alpha_v = f(v|S_0)$ in a priority queue, and the iteration index $\beta_v = 0$ at which it was least updated
- At iteration i, repeat

Minoux's Accelerated Greedy

- Store the initial increments $\alpha_v = f(v|S_0)$ in a priority queue, and the iteration index $\beta_v = 0$ at which it was least updated
- At iteration i, repeat
  - "pop" the top element (largest α_v), and let u be the new top

Minoux's Accelerated Greedy

- Store the initial increments $\alpha_v = f(v|S_0)$ in a priority queue, and the iteration index $\beta_v = 0$ at which it was least updated
- At iteration i, repeat
  - "pop" the top element (largest α_v), and let u be the new top
  - if $\beta_v < i$ then compute the exact increment $\alpha_v := f(v|S_i)$ and update $\beta_v = i$

Minoux's Accelerated Greedy

- Store the initial increments $\alpha_v = f(v|S_0)$ in a priority queue, and the iteration index $\beta_v = 0$ at which it was least updated
- At iteration i, repeat
  - "pop" the top element (largest α_v), and let u be the new top
  - if $\beta_v < i$ then compute the exact increment $\alpha_v := f(v|S_i)$ and update $\beta_v = i$
  - if $\alpha_v < \alpha_u$ then return v to the queue

Minoux's Accelerated Greedy

- Store the initial increments $\alpha_v = f(v|S_0)$ in a priority queue, and the iteration index $\beta_v = 0$ at which it was least updated
- At iteration i, repeat
  - "pop" the top element (largest α_v), and let u be the new top
  - if $\beta_v < i$ then compute the exact increment $\alpha_v := f(v|S_i)$ and update $\beta_v = i$
  - if $\alpha_v < \alpha_u$ then return v to the queue

Minoux's Accelerated Greedy

- Store the initial increments $\alpha_v = f(v|S_0)$ in a priority queue, and the iteration index $\beta_v = 0$ at which it was least updated
- At iteration i, repeat
  - "pop" the top element (largest α_v), and let u be the new top
  - if $\beta_v < i$ then compute the exact increment $\alpha_v := f(v|S_i)$ and update $\beta_v = i$
  - if $\alpha_v < \alpha_u$ then return v to the queue

  until $\alpha_v \geq \alpha_u$: v is now selected by Greedy

Minoux's Accelerated Greedy

- Store the initial increments $\alpha_v = f(v|S_0)$ in a priority queue, and the iteration index $\beta_v = 0$ at which it was least updated
- At iteration i, repeat
  - "pop" the top element (largest α_v), and let u be the new top
  - if $\beta_v < i$ then compute the exact increment $\alpha_v := f(v|S_i)$ and update $\beta_v = i$
  - if $\alpha_v < \alpha_u$ then return v to the queue

  until $\alpha_v \geq \alpha_u$: v is now selected by Greedy

Validity follows from submodularity, i.e., nonincreasing increments: as i increases, the current S_i also increases, the increments $f(v|S_i)$ decrease, and thus each α_v remains an upper bound on $f(v|S_i)$

Minoux's Accelerated Greedy

- Store the initial increments $\alpha_v = f(v|S_0)$ in a priority queue, and the iteration index $\beta_v = 0$ at which it was least updated
- At iteration i, repeat
  - "pop" the top element (largest α_v), and let u be the new top
  - if $\beta_v < i$ then compute the exact increment $\alpha_v := f(v|S_i)$ and update $\beta_v = i$
  - if $\alpha_v < \alpha_u$ then return v to the queue

  until $\alpha_v \geq \alpha_u$: v is now selected by Greedy

Validity follows from submodularity, i.e., nonincreasing increments: as i increases, the current S_i also increases, the increments $f(v|S_i)$ decrease, and thus each α_v remains an upper bound on $f(v|S_i)$

In practice, Minoux's trick often yields enormous speedups (over 700-fold) over standard implementation of Greedy, for very large data sets

Nonmonotone SFMax

- If f is an arbitrary submodular function (neither polymatroidal, nor necessarily positive or negative), then verifying whether its maximum is positive or negative is already NP-hard

Nonmonotone SFMax

- If f is an arbitrary submodular function (neither polymatroidal, nor necessarily positive or negative), then verifying whether its maximum is positive or negative is already NP-hard
- Therefore, submodular function max in such case is inapproximable (unless P$=$NP)

Nonmonotone SFMax

- If f is an arbitrary submodular function (neither polymatroidal, nor necessarily positive or negative), then verifying whether its maximum is positive or negative is already NP-hard
- Therefore, submodular function max in such case is inapproximable (unless P=NP)
  - since any such procedure would give us the sign of the max

Nonmonotone SFMax

- If f is an arbitrary submodular function (neither polymatroidal, nor necessarily positive or negative), then verifying whether its maximum is positive or negative is already NP-hard
- Therefore, submodular function max in such case is inapproximable (unless P=NP)
  - since any such procedure would give us the sign of the max
- Thus, we will assume that f is non-negative and otherwise arbitrary submodular

Nonmonotone SFMax

- If f is an arbitrary submodular function (neither polymatroidal, nor necessarily positive or negative), then verifying whether its maximum is positive or negative is already NP-hard
- Therefore, submodular function max in such case is inapproximable (unless P=NP)
  - since any such procedure would give us the sign of the max
- Thus, we will assume that f is non-negative and otherwise arbitrary submodular
- Feige, Mirrokni & Vondrak (2007, 2011) show that, in the value oracle model, for every $\epsilon > 0$ a $\left(\frac{1}{2} + \epsilon\right)$-approximation requires an exponential number of oracle calls

Nonmonotone SFMax

- If f is an arbitrary submodular function (neither polymatroidal, nor necessarily positive or negative), then verifying whether its maximum is positive or negative is already NP-hard
- Therefore, submodular function max in such case is inapproximable (unless P=NP)
  - since any such procedure would give us the sign of the max
- Thus, we will assume that f is non-negative and otherwise arbitrary submodular
- Feige, Mirrokni & Vondrak (2007, 2011) show that, in the value oracle model, for every $\epsilon > 0$ a $\left(\frac{1}{2} + \epsilon\right)$-approximation requires an exponential number of oracle calls
  - even if f is known to be symmetric

Nonmonotone SFMax

- If f is an arbitrary submodular function (neither polymatroidal, nor necessarily positive or negative), then verifying whether its maximum is positive or negative is already NP-hard
- Therefore, submodular function max in such case is inapproximable (unless P=NP)
  - since any such procedure would give us the sign of the max
- Thus, we will assume that f is non-negative and otherwise arbitrary submodular
- Feige, Mirrokni & Vondrak (2007, 2011) show that, in the value oracle model, for every $\epsilon > 0$ a $\left(\frac{1}{2} + \epsilon\right)$-approximation requires an exponential number of oracle calls
  - even if f is known to be symmetric
- We will see a $(\frac{1}{3} - \epsilon)$-approximation, also due to Feige & al,

Nonmonotone SFMax

- If f is an arbitrary submodular function (neither polymatroidal, nor necessarily positive or negative), then verifying whether its maximum is positive or negative is already NP-hard
- Therefore, submodular function max in such case is inapproximable (unless P=NP)
  - since any such procedure would give us the sign of the max
- Thus, we will assume that f is non-negative and otherwise arbitrary submodular
- Feige, Mirrokni & Vondrak (2007, 2011) show that, in the value oracle model, for every $\epsilon > 0$ a $\left(\frac{1}{2} + \epsilon\right)$-approximation requires an exponential number of oracle calls
  - even if f is known to be symmetric
- We will see a $(\frac{1}{3} - \epsilon)$-approximation, also due to Feige & al,
  - using O($\frac{1}{\epsilon} n^3 \log n$) EO's

Nonmonotone SFMax

- If f is an arbitrary submodular function (neither polymatroidal, nor necessarily positive or negative), then verifying whether its maximum is positive or negative is already NP-hard
- Therefore, submodular function max in such case is inapproximable (unless P=NP)
  - since any such procedure would give us the sign of the max
- Thus, we will assume that f is non-negative and otherwise arbitrary submodular
- Feige, Mirrokni & Vondrak (2007, 2011) show that, in the value oracle model, for every $\epsilon > 0$ a $\left(\frac{1}{2} + \epsilon\right)$-approximation requires an exponential number of oracle calls
  - even if f is known to be symmetric
- We will see a $(\frac{1}{3} - \epsilon)$-approximation, also due to Feige & al,
  - using O($\frac{1}{\epsilon} n^3 \log n$) EO's
  - and based on local search (not on a greedy approach!)

Local Search

Local Search

- A sequential method that starts at a feasible solution

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

  and tries to improve it by a sequence of (usually, simple) moves

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

  and tries to improve it by a sequence of (usually, simple) moves
  - e.g., add, or drop, an element to/from the current set S

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

  and tries to improve it by a sequence of (usually, simple) moves
  - e.g., add, or drop, an element to/from the current set S
  - It must be possible, in polytime, to find an improving move or decide none exists

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

  and tries to improve it by a sequence of (usually, simple) moves
  - e.g., add, or drop, an element to/from the current set S
  - It must be possible, in polytime, to find an improving move or decide none exists
- Local search methods differ in their search strategy

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

  and tries to improve it by a sequence of (usually, simple) moves
  - e.g., add, or drop, an element to/from the current set S
  - It must be possible, in polytime, to find an improving move or decide none exists
- Local search methods differ in their search strategy
  - simple hill climbing, restarts, "tabu search", simulated annealing,...

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

  and tries to improve it by a sequence of (usually, simple) moves
  - e.g., add, or drop, an element to/from the current set S
  - It must be possible, in polytime, to find an improving move or decide none exists
- Local search methods differ in their search strategy
  - simple hill climbing, restarts, "tabu search", simulated annealing,. . .

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

  and tries to improve it by a sequence of (usually, simple) moves
  - e.g., add, or drop, an element to/from the current set S
  - It must be possible, in polytime, to find an improving move or decide none exists
- Local search methods differ in their search strategy
  - simple hill climbing, restarts, "tabu search", simulated annealing,...

  but they terminate with a local optimum, i.e., a feasible solution that cannot be improved by the available moves

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

  and tries to improve it by a sequence of (usually, simple) moves
  - e.g., add, or drop, an element to/from the current set S
  - It must be possible, in polytime, to find an improving move or decide none exists
- Local search methods differ in their search strategy
  - simple hill climbing, restarts, "tabu search", simulated annealing,...

  but they terminate with a local optimum, i.e., a feasible solution that cannot be improved by the available moves
- Two main issues in evaluating a local search method:

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

  and tries to improve it by a sequence of (usually, simple) moves
  - e.g., add, or drop, an element to/from the current set S
  - It must be possible, in polytime, to find an improving move or decide none exists
- Local search methods differ in their search strategy
  - simple hill climbing, restarts, "tabu search", simulated annealing,...

  but they terminate with a local optimum, i.e., a feasible solution that cannot be improved by the available moves
- Two main issues in evaluating a local search method:
  - **Running time**: does it go thru a polynomially bounded number of steps?

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

  and tries to improve it by a sequence of (usually, simple) moves
  - e.g., add, or drop, an element to/from the current set S
  - It must be possible, in polytime, to find an improving move or decide none exists
- Local search methods differ in their search strategy
  - simple hill climbing, restarts, "tabu search", simulated annealing,...

  but they terminate with a local optimum, i.e., a feasible solution that cannot be improved by the available moves
- Two main issues in evaluating a local search method:
  - **Running time**: does it go thru a polynomially bounded number of steps?
  - **Solution quality**: do we have a performance guarantee?

Local Search

- A sequential method that starts at a feasible solution
  - e.g., any subset S of the ground set E

  and tries to improve it by a sequence of (usually, simple) moves
  - e.g., add, or drop, an element to/from the current set S
  - It must be possible, in polytime, to find an improving move or decide none exists
- Local search methods differ in their search strategy
  - simple hill climbing, restarts, "tabu search", simulated annealing,...

  but they terminate with a local optimum, i.e., a feasible solution that cannot be improved by the available moves
- Two main issues in evaluating a local search method:
  - **Running time**: does it go thru a polynomially bounded number of steps?
  - **Solution quality**: do we have a performance guarantee? i.e., how "bad" (in objective value) can a local optimum be?

Generic Local Search

Generic Local Search

The neighbourhood $N(S)$ of a solution is the set of all solutions that can be reached from S by an available move.

Generic Local Search

The neighbourhood $N(S)$ of a solution is the set of all solutions that can be reached from S by an available move.

Generic Local Search (“**Hill Climbing**”)

Generic Local Search

The neighbourhood $N(S)$ of a solution is the set of all solutions that can be reached from S by an available move.

Generic Local Search ("**Hill Climbing**")

1. Initialization: find a (feasible) solution S

Generic Local Search

The neighbourhood $N(S)$ of a solution is the set of all solutions that can be reached from S by an available move.

Generic Local Search ("**Hill Climbing**")

1. Initialization: find a (feasible) solution S
2. While there exists an improving solution $S^+ \in N(S)$ do
   $S := S^+$

Generic Local Search

The neighbourhood $N(S)$ of a solution is the set of all solutions that can be reached from S by an available move.

Generic Local Search ("**Hill Climbing**")

1. Initialization: find a (feasible) solution S
2. While there exists an improving solution $S^+ \in N(S)$ do $S := S^+$
3. Output S

Generic Local Search

The neighbourhood $N(S)$ of a solution is the set of all solutions that can be reached from S by an available move.

Generic Local Search ("**Hill Climbing**")

1. Initialization: find a (feasible) solution S
2. While there exists an improving solution $S^+ \in N(S)$ do
   $S := S^+$
3. Output S

Generic Local Search

The neighbourhood $N(S)$ of a solution is the set of all solutions that can be reached from S by an available move.

Generic Local Search ("**Hill Climbing**")

1. Initialization: find a (feasible) solution S
2. While there exists an improving solution $S^+ \in N(S)$ do $S := S^+$
3. Output S

If there is a finite number of solutions (and we only accept strict improvements) then Generic Local Search terminates in a finite number of steps and outputs a local optimum

Generic Local Search

The neighbourhood $N(S)$ of a solution is the set of all solutions that can be reached from S by an available move.

Generic Local Search ("**Hill Climbing**")

1. Initialization: find a (feasible) solution S
2. While there exists an improving solution $S^+ \in N(S)$ do
   $S := S^+$
3. Output S

If there is a finite number of solutions (and we only accept strict improvements) then Generic Local Search terminates in a finite number of steps and outputs a local optimum

- at this point, we can only guarantee finiteness, but not polynomiality

Generic Local Search

The neighbourhood $N(S)$ of a solution is the set of all solutions that can be reached from S by an available move.

Generic Local Search (“**Hill Climbing**”)

1. Initialization: find a (feasible) solution S
2. While there exists an improving solution $S^+ \in N(S)$ do $S := S^+$
3. Output S

If there is a finite number of solutions (and we only accept strict improvements) then Generic Local Search terminates in a finite number of steps and outputs a local optimum

- at this point, we can only guarantee finiteness, but not polynomiality
- in fact, most of these problems are PLS-complete

A Polytime Version of Local Search

Given $\epsilon > 0$, $S^+ \in N(S)$ is ϵ-improving if its objective value $f(S^+) > (1+\epsilon)f(S)$ (for a maximization problem)

A Polytime Version of Local Search

Given $\epsilon > 0$, $S^+ \in N(S)$ is ϵ-improving if its objective value $f(S^+) > (1+\epsilon)f(S)$ (for a maximization problem)
S is an ϵ-local optimum if $f(S^+) \leq (1+\epsilon)f(S)$ for all $S^+ \in N(S)$

A Polytime Version of Local Search

Given $\epsilon > 0$, $S^+ \in N(S)$ is ϵ-improving if its objective value $f(S^+) > (1+\epsilon)f(S)$ (for a maximization problem)
S is an ϵ-local optimum if $f(S^+) \leq (1+\epsilon)f(S)$ for all $S^+ \in N(S)$

- i.e., if it has no ϵ-improving neighbor

A Polytime Version of Local Search

Given $\epsilon > 0$, $S^+ \in N(S)$ is ϵ-improving if its objective value $f(S^+) > (1+\epsilon)f(S)$ (for a maximization problem)
S is an ϵ-local optimum if $f(S^+) \leq (1+\epsilon)f(S)$ for all $S^+ \in N(S)$

- i.e., if it has no ϵ-improving neighbor

Modified Local Search (**MLS**): given $\epsilon > 0$,

A Polytime Version of Local Search

Given $\epsilon > 0$, $S^+ \in N(S)$ is ϵ-improving if its objective value $f(S^+) > (1+\epsilon)f(S)$ (for a maximization problem)
S is an ϵ-local optimum if $f(S^+) \leq (1+\epsilon)f(S)$ for all $S^+ \in N(S)$

- i.e., if it has no ϵ-improving neighbor

Modified Local Search (**MLS**): given $\epsilon > 0$,

1. Initialization: find a (feasible) solution S

A Polytime Version of Local Search

Given $\epsilon > 0$, $S^+ \in N(S)$ is ϵ-improving if its objective value $f(S^+) > (1+\epsilon)f(S)$ (for a maximization problem)
S is an ϵ-local optimum if $f(S^+) \leq (1+\epsilon)f(S)$ for all $S^+ \in N(S)$

- i.e., if it has no ϵ-improving neighbor

Modified Local Search (**MLS**): given $\epsilon > 0$,

1. Initialization: find a (feasible) solution S
2. While there exists an ϵ-improving solution $S^+ \in N(S)$ do $S := S^+$

A Polytime Version of Local Search

Given $\epsilon > 0$, $S^+ \in N(S)$ is ϵ-improving if its objective value $f(S^+) > (1+\epsilon)f(S)$ (for a maximization problem)
S is an ϵ-local optimum if $f(S^+) \leq (1+\epsilon)f(S)$ for all $S^+ \in N(S)$

- i.e., if it has no ϵ-improving neighbor

Modified Local Search (**MLS**): given $\epsilon > 0$,

1. Initialization: find a (feasible) solution S
2. While there exists an ϵ-improving solution $S^+ \in N(S)$ do $S := S^+$
3. Output S

A Polytime Version of Local Search

Given $\epsilon > 0$, $S^+ \in N(S)$ is ϵ-improving if its objective value $f(S^+) > (1+\epsilon)f(S)$ (for a maximization problem)
S is an ϵ-local optimum if $f(S^+) \leq (1+\epsilon)f(S)$ for all $S^+ \in N(S)$

- i.e., if it has no ϵ-improving neighbor

Modified Local Search (**MLS**): given $\epsilon > 0$,

1. Initialization: find a (feasible) solution S
2. While there exists an ϵ-improving solution $S^+ \in N(S)$ do
   $S := S^+$
3. Output S

A Polytime Version of Local Search

Given $\epsilon > 0$, $S^+ \in N(S)$ is ϵ-improving if its objective value $f(S^+) > (1+\epsilon)f(S)$ (for a maximization problem)
S is an ϵ-local optimum if $f(S^+) \leq (1+\epsilon)f(S)$ for all $S^+ \in N(S)$

- i.e., if it has no ϵ-improving neighbor

Modified Local Search (MLS): given $\epsilon > 0$,

1. Initialization: find a (feasible) solution S
2. While there exists an ϵ-improving solution $S^+ \in N(S)$ do $S := S^+$
3. Output S

If $f(S_0) > 0$ then after k iterations the current solution S_k satisfies $f(S_k) > (1+\epsilon)^k f(S_0)$

A Polytime Version of Local Search

Given $\epsilon > 0$, $S^+ \in N(S)$ is ϵ-improving if its objective value $f(S^+) > (1+\epsilon)f(S)$ (for a maximization problem)
S is an ϵ-local optimum if $f(S^+) \leq (1+\epsilon)f(S)$ for all $S^+ \in N(S)$

- i.e., if it has no ϵ-improving neighbor

Modified Local Search (**MLS**): given $\epsilon > 0$,

1. Initialization: find a (feasible) solution S
2. While there exists an ϵ-improving solution $S^+ \in N(S)$ do $S := S^+$
3. Output S

If $f(S_0) > 0$ then after k iterations the current solution S_k satisfies $f(S_k) > (1+\epsilon)^k f(S_0)$
$\Rightarrow$ If $\log(\text{OPT}/f(S_0))$ is polynomially bounded (in the instance input size) then for every fixed $\epsilon > 0$, MLS terminates and outputs an ϵ-local optimum after at most $\log(\text{OPT}/f(S_0)) \,/\, \log(1+\epsilon)$ iterations, i.e. in polytime

Local Optima for SFMax

Local Optima for SFMax

Let $S \subseteq E$ be a local maximum for the add & drop moves

Local Optima for SFMax

Let $S \subseteq E$ be a local maximum for the add & drop moves

Lemma
If $f : 2^E \mapsto \mathbb{R}$ is normalized and submodular, and S is such a local optimum then

1. $f(R) \leq f(S)$ for all $R \subset S$, and

Local Optima for SFMax

Let $S \subseteq E$ be a local maximum for the add & drop moves

Lemma
If $f : 2^E \mapsto \mathbb{R}$ is normalized and submodular, and S is such a local optimum then

1. $f(R) \leq f(S)$ for all $R \subset S$, and
2. $f(T) \leq f(S)$ for all $T \supset S$

Local Optima for SFMax

Let $S \subseteq E$ be a local maximum for the add & drop moves

Lemma
If $f : 2^E \mapsto \mathbb{R}$ is normalized and submodular, and S is such a local optimum then

1. $f(R) \leq f(S)$ for all $R \subset S$, and
2. $f(T) \leq f(S)$ for all $T \supset S$

Local Optima for SFMax

Let $S \subseteq E$ be a local maximum for the add & drop moves

Lemma
If $f : 2^E \mapsto \mathbb{R}$ is normalized and submodular, and S is such a local optimum then

1. $f(R) \leq f(S)$ for all $R \subset S$, and
2. $f(T) \leq f(S)$ for all $T \supset S$

Proof by induction on $d = |S \setminus R|$ for 1.

Local Optima for SFMax

Let $S \subseteq E$ be a local maximum for the add & drop moves

Lemma
If $f : 2^E \mapsto \mathbb{R}$ is normalized and submodular, and S is such a local optimum then

1. $f(R) \leq f(S)$ for all $R \subset S$, and
2. $f(T) \leq f(S)$ for all $T \supset S$

Proof by induction on $d = |S \setminus R|$ for 1.
Base case: if $d = 1$ then $R \in N(S)$ and $f(R) \leq f(S)$

Local Optima for SFMax

Let $S \subseteq E$ be a local maximum for the add & drop moves

Lemma
If $f : 2^E \mapsto \mathbb{R}$ is normalized and submodular, and S is such a local optimum then

1. $f(R) \leq f(S)$ for all $R \subset S$, and
2. $f(T) \leq f(S)$ for all $T \supset S$

Proof by induction on $d = |S \setminus R|$ for 1.
Base case: if $d = 1$ then $R \in N(S)$ and $f(R) \leq f(S)$
Induction: assume 1 holds for $d - 1$ and consider any $R \subset S$ with $|S \setminus R| = d$. Choose $u \in S \setminus R$. Then

$$f(R) \quad \leq \quad f(R + u) + f(S - u) - f(S)$$

Local Optima for SFMax

Let $S \subseteq E$ be a local maximum for the add & drop moves

Lemma
If $f : 2^E \mapsto \mathbb{R}$ is normalized and submodular, and S is such a local optimum then

1. $f(R) \leq f(S)$ for all $R \subset S$, and
2. $f(T) \leq f(S)$ for all $T \supset S$

Proof by induction on $d = |S \setminus R|$ for 1.
Base case: if $d = 1$ then $R \in N(S)$ and $f(R) \leq f(S)$
Induction: assume 1 holds for $d - 1$ and consider any $R \subset S$ with $|S \setminus R| = d$. Choose $u \in S \setminus R$. Then

$$\begin{aligned} f(R) &\leq f(R+u) + f(S-u) - f(S) \\ &\leq f(S-u) \end{aligned}$$

Local Optima for SFMax

Let $S \subseteq E$ be a local maximum for the add & drop moves

Lemma
If $f : 2^E \mapsto \mathbb{R}$ is normalized and submodular, and S is such a local optimum then

1. $f(R) \leq f(S)$ for all $R \subset S$, and
2. $f(T) \leq f(S)$ for all $T \supset S$

Proof by induction on $d = |S \setminus R|$ for 1.
Base case: if $d = 1$ then $R \in N(S)$ and $f(R) \leq f(S)$
Induction: assume 1 holds for $d - 1$ and consider any $R \subset S$ with $|S \setminus R| = d$. Choose $u \in S \setminus R$. Then

$$\begin{aligned} f(R) &\leq f(R+u) + f(S-u) - f(S) \\ &\leq f(S-u) \\ &\leq f(S) \end{aligned}$$

Local Optima for SFMax

Let $S \subseteq E$ be a local maximum for the add & drop moves

Lemma
If $f : 2^E \mapsto \mathbb{R}$ is normalized and submodular, and S is such a local optimum then

1. $f(R) \leq f(S)$ for all $R \subset S$, and
2. $f(T) \leq f(S)$ for all $T \supset S$

Proof by induction on $d = |S \setminus R|$ for 1.
Base case: if $d = 1$ then $R \in N(S)$ and $f(R) \leq f(S)$
Induction: assume 1 holds for $d-1$ and consider any $R \subset S$ with $|S \setminus R| = d$. Choose $u \in S \setminus R$. Then

$$\begin{aligned} f(R) &\leq f(R+u) + f(S-u) - f(S) \\ &\leq f(S-u) \\ &\leq f(S) \end{aligned}$$

The proof of 2 is similar QED

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular,

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves,

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a $1/3$-approximation

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a $1/3$-approximation

Proof:

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a $1/3$-approximation

Proof: Let S^* be an optimum solution, then

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a $1/3$-approximation

Proof: Let S^* be an optimum solution, then

$$3\,f(S') \;\geq\; 2\,f(S') + f(N \setminus S')$$

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a 1/3-approximation

Proof: Let S^* be an optimum solution, then

$$\begin{aligned} 3\, f(S') &\geq 2\, f(S') + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(S' \cup S^*) + f(N \setminus S') \end{aligned}$$

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a $1/3$-approximation

Proof: Let S^* be an optimum solution, then

$$\begin{aligned} 3\,f(S') &\geq 2\,f(S') + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(S' \cup S^*) + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(N) + f(S^* \setminus S') \end{aligned}$$

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a $1/3$-approximation

Proof: Let S^* be an optimum solution, then

$$\begin{aligned}
3\,f(S') &\geq 2\,f(S') + f(N \setminus S') \\
&\geq f(S' \cap S^*) + f(S' \cup S^*) + f(N \setminus S') \\
&\geq f(S' \cap S^*) + f(N) + f(S^* \setminus S') \\
&\geq f(S^*) + f(\emptyset) + f(N)
\end{aligned}$$

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a 1/3-approximation

Proof: Let S^* be an optimum solution, then

$$\begin{aligned}
3\, f(S') &\geq 2\, f(S') + f(N \setminus S') \\
&\geq f(S' \cap S^*) + f(S' \cup S^*) + f(N \setminus S') \\
&\geq f(S' \cap S^*) + f(N) + f(S^* \setminus S') \\
&\geq f(S^*) + f(\emptyset) + f(N) \quad \geq \quad f(S^*) \qquad\qquad QED
\end{aligned}$$

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a 1/3-approximation

Proof: Let S^* be an optimum solution, then

$$\begin{aligned} 3\,f(S') &\geq 2\,f(S') + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(S' \cup S^*) + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(N) + f(S^* \setminus S') \\ &\geq f(S^*) + f(\emptyset) + f(N) \quad \geq \quad f(S^*) \qquad QED \end{aligned}$$

Theorem. If, in addition to the assumptions of the preceding theorem, f is also symmetric then S' is a 1/2-approximation

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a $1/3$-approximation

Proof: Let S^* be an optimum solution, then

$$\begin{aligned}
3\,f(S') &\geq 2\,f(S') + f(N \setminus S') \\
&\geq f(S' \cap S^*) + f(S' \cup S^*) + f(N \setminus S') \\
&\geq f(S' \cap S^*) + f(N) + f(S^* \setminus S') \\
&\geq f(S^*) + f(\emptyset) + f(N) \quad \geq \quad f(S^*) \qquad\qquad QED
\end{aligned}$$

Theorem. If, in addition to the assumptions of the preceding theorem, f is also symmetric then S' is a $1/2$-approximation

Proof:

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a $1/3$-approximation

Proof: Let S^* be an optimum solution, then

$$\begin{aligned} 3\,f(S') &\geq 2\,f(S') + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(S' \cup S^*) + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(N) + f(S^* \setminus S') \\ &\geq f(S^*) + f(\emptyset) + f(N) \quad \geq \quad f(S^*) \qquad QED \end{aligned}$$

Theorem. If, in addition to the assumptions of the preceding theorem, f is also symmetric then S' is a $1/2$-approximation

Proof: $N \setminus S'$ is also a local optimum, so

$$2\,f(S') = f(S') + f(N \setminus S')$$

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a $1/3$-approximation

Proof: Let S^* be an optimum solution, then

$$\begin{aligned} 3\,f(S') &\geq 2\,f(S') + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(S' \cup S^*) + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(N) + f(S^* \setminus S') \\ &\geq f(S^*) + f(\emptyset) + f(N) \quad \geq \quad f(S^*) \qquad QED \end{aligned}$$

Theorem. If, in addition to the assumptions of the preceding theorem, f is also symmetric then S' is a $1/2$-approximation

Proof: $N \setminus S'$ is also a local optimum, so

$$2\,f(S') = f(S') + f(N \setminus S') \geq f(S' \cap S^*) + f((N \setminus S') \cap S^*)$$

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a 1/3-approximation

Proof: Let S^* be an optimum solution, then

$$\begin{aligned} 3\,f(S') &\geq 2\,f(S') + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(S' \cup S^*) + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(N) + f(S^* \setminus S') \\ &\geq f(S^*) + f(\emptyset) + f(N) \quad \geq \quad f(S^*) \qquad QED \end{aligned}$$

Theorem. If, in addition to the assumptions of the preceding theorem, f is also symmetric then S' is a 1/2-approximation

Proof: $N \setminus S'$ is also a local optimum, so

$$\begin{aligned} 2\,f(S') &= f(S') + f(N \setminus S') \quad \geq \quad f(S' \cap S^*) + f((N \setminus S') \cap S^*) \\ &\geq f(S^*) + f(\emptyset) \end{aligned}$$

Local Optima for SFMax (2)

Theorem [Feige, Mirrokni & Vondrák, 2011]
If $f : 2^E \mapsto \mathbb{R}$ is normalized, nonnegative and submodular, and S is a local optimum for the add & drop moves, then

$$S' \in \arg\max\{f(T) : T \in \{S, N \setminus S\}\},$$

the better of S and its complement, is a 1/3-approximation

Proof: Let S^* be an optimum solution, then

$$\begin{aligned} 3\,f(S') &\geq 2\,f(S') + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(S' \cup S^*) + f(N \setminus S') \\ &\geq f(S' \cap S^*) + f(N) + f(S^* \setminus S') \\ &\geq f(S^*) + f(\emptyset) + f(N) \geq f(S^*) \qquad QED \end{aligned}$$

Theorem. If, in addition to the assumptions of the preceding theorem, f is also symmetric then S' is a 1/2-approximation

Proof: $N \setminus S'$ is also a local optimum, so

$$\begin{aligned} 2\,f(S') &= f(S') + f(N \setminus S') \geq f(S' \cap S^*) + f((N \setminus S') \cap S^*) \\ &\geq f(S^*) + f(\emptyset) \geq f(S^*) \qquad QED \end{aligned}$$

Approximation Algorithms for SFMax

Approximation Algorithms for SFMax

Combining with MLS (for polytime) we get

- a $(\frac{1}{3} - \epsilon)$-approximation for SFMax with a normalized nonnegative objective, and

Approximation Algorithms for SFMax

Combining with MLS (for polytime) we get

- a $(\frac{1}{3} - \epsilon)$-approximation for SFMax with a normalized nonnegative objective, and
- a $(\frac{1}{2} - \epsilon)$-approximation if it is also symmetric

Approximation Algorithms for SFMax

Combining with MLS (for polytime) we get

- a $(\frac{1}{3} - \epsilon)$-approximation for SFMax with a normalized nonnegative objective, and
- a $(\frac{1}{2} - \epsilon)$-approximation if it is also symmetric
  - matches the $\left(\frac{1}{2} + \epsilon\right)$ inapproximability for Sym-SFMax

Approximation Algorithms for SFMax

Combining with MLS (for polytime) we get

- a $(\frac{1}{3} - \epsilon)$-approximation for SFMax with a normalized nonnegative objective, and
- a $(\frac{1}{2} - \epsilon)$-approximation if it is also symmetric
  - matches the $\left(\frac{1}{2} + \epsilon\right)$ inapproximability for Sym-SFMax

Approximation Algorithms for SFMax

Combining with MLS (for polytime) we get

- a $(\frac{1}{3} - \epsilon)$-approximation for SFMax with a normalized nonnegative objective, and
- a $(\frac{1}{2} - \epsilon)$-approximation if it is also symmetric
  - matches the $\left(\frac{1}{2} + \epsilon\right)$ inapproximability for Sym-SFMax

Buchbinder, Feldman, Naor & Schwartz (2012): a randomized, linear-time, greedy-like algorithm which is a $\frac{1}{2}$-approximation

- therefore best possible for SFMax

Approximation Algorithms for SFMax

Combining with MLS (for polytime) we get

- a $(\frac{1}{3} - \epsilon)$-approximation for SFMax with a normalized nonnegative objective, and
- a $(\frac{1}{2} - \epsilon)$-approximation if it is also symmetric
  - matches the $\left(\frac{1}{2} + \epsilon\right)$ inapproximability for Sym-SFMax

Buchbinder, Feldman, Naor & Schwartz (2012): a randomized, linear-time, greedy-like algorithm which is a $\frac{1}{2}$-approximation

- therefore best possible for SFMax

Other recent approximation results for monotone and non-monotone SFMax subject to a variety of constraints

- one or several knapsacks, matroidal constraints, ...

Additional References (1)

- Anglès d'Auriac, Jean -Christian, Ferenc Iglói, Myriam Preissmann, and Andras Sebő, 2002. "Optimal cooperation and submodularity for computing Potts' partition functions with a large number of states" *J. Phys. A*35 6973–6983.
- Baïou, Mourad, Francisco Barahona, and Ridha Mahjoub, 2000. "Separation of Partition Inequalities" *Math. of OR* 25 243-254.
- Buchbinder, Niv, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz, 2012. "A tight linear time (1/2)-approximation for unconstrained submodular maximization" *FOCS 2012* 649–658.
- Feige, Uriel, 1998. "A threshold of ln n for approximating Set Cover" *J. ACM* 45 634–652.
- Feige, Uriel, Vahab S. Mirrokni, and Jan Vondrak, 2011. "Maximizing non-monotone submodular functions" *SIAM J. Comput.* 40(4) 1133–1153.
- Frank, András, and Eva Tardos, 1988. "Generalized polymatroids and submodular flows" *Math. Prog.* 42(1–3) 489–563.
- Goemans, Michel X., and José A. Soto, 2013. "Algorithms for Symmetric Submodular Function Minimization under Hereditary Constraints and Generalizations" *SIAM J. Discr. Math.* 27(2) 1123–1145.

Additional References (2)

- Goldschmidt, Olivier, and Dorit S. Hochbaum, 1994. "A polynomial algorithm for the k-cut problem for fixed k" *Math. of OR* 19(1) 24–37.
- Minoux, Michel, 1978. "Accelerated greedy algorithms for maximizing submodular set functions" *Optimization Techniques* (8th IFIP TC 7 Optimization Conference, Springer) 234–243.
- Nagamochi, Hiroshi, and Toshihide Ibaraki, 1998. "A note on minimizing submodular functions" *Info. Proc. Letters* 67 239–244.
- Nagamochi, Hiroshi, and Toshihide Ibaraki, 2000. "A fast algorithm for computing minimum 3-way and 4-way cuts" *Math. Prog.* 88(3) 507–520.
- Okumoto, Kazumasa, Takuro Fukunaga, and Hiroshi Nagamochi, 2010. "Divide-and-Conquer Algorithms for Partitioning Hypergraphs and Submodular Systems" *Algorithmica* 62(3-4) 787–806.
- Zhao, Liang, Hiroshi Nagamochi, and Toshihide Ibaraki, 2005. "Greedy splitting algorithms for approximating multiway partition problems" *Math. Prog.* 102(1) 167–183.

Short Course on Submodular Functions
Part 2: Extensions and Related Problems
Session 3: Submodularity in Vector Spaces

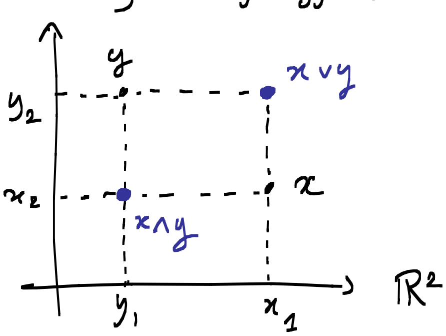
S. Thomas McCormick and Maurice Queyranne
Sauder School of Business, UBC

Rappels : Un **treillis** (en anglais : *lattice*) est un un ensemble partiellement ordonné (un "*poset*") $(L, \leq)$ tel que, pour tous $a, b \in L$ il existe

- une plus petite borne supérieure commune $a \vee b$, le **supremum** (ou **sup**) de a et b (en anglais, *the join of a and b*), c'est à dire un unique élément $s = a \vee b \in L$ tel que $a \leq s$, $b \leq s$ et pour tout $c \in L$ tel que $a \leq c$ et $b \leq c$ on doit avoir $s \leq c$
- et une plus grande borne inférieure commune $a \wedge b$, (l'**infimum** (ou **inf**) de a et b (*the meet of a and b*) : $a \wedge b \leq a$, $a \wedge b \leq b$ et $\forall c \in L$ $(c \leq a$ et $c \leq b) \Rightarrow c \leq a \wedge b$

Exemples : $(2^E, \subseteq)$ avec $a \vee b = a \cup b$ (union) et $a \wedge b = a \cap b$ (intersection)

- $(\mathbb{T}^E, \Rightarrow)$ où $\mathbb{T} = (\text{Vrai}, \text{Faux})$, $\mathbb{T}^E$ est l'ensemble des fonctions logiques définies sur E,
  $\Rightarrow$ est l'implication ($a \Rightarrow b$ signifie que $b(e) = \text{Vrai}$ pour tous les $e \in E$ tels que $a(e) = \text{Vrai}$)
  $\vee$ est la disjonction ("ou" logique : $\forall e \in E$ $a \vee b(e) = \text{Vrai}$ si et seulement si l'un au moins de $a(e)$ ou $b(e) = \text{Vrai}$)
  $\wedge$ est la conjonction ("et" logique : $\forall e \in E$ $a \wedge b(e) = \text{Vrai}$ ssi $a(e) = b(e) = \text{Vrai}$)
- $(\mathbb{R}^d, \leq)$ où $\leq$ est l'ordre partiel "par composantes" $x \leq y \Leftrightarrow x_j \leq y_j$ $\forall j = 1..d$
  $\vee$ est le supremum par composantes
  $(x \vee y)_j = x_j \vee y_j = \max\{x_j, y_j\}$ $\forall j = 1..d$
  $\wedge$ est l'infimum par composantes
  $(x \wedge y)_j = x_j \wedge y_j = \min\{x_j, y_j\}$ $\forall j = 1..d$



On définit de même $(\mathbb{Z}^d, \leq)$, $(\mathbb{B}^d, \leq)$ (où $\mathbb{B} = \{0,1\}$)
et plus généralement $(\bigotimes_{j=1}^{d} A_j, \leq)$ où $\bigotimes_{j=1}^{d} A_j$ est le produit Cartésien de sous-ensembles arbitraires $A_j \subseteq \mathbb{R}$, avec l'ordre partiel par composantes $\leq$

En particulier, pour $\ell, u \in \mathbb{Z}^d$ tels que $\ell \leq u$,
la **boîte** $B_{\ell,u} = \{x \in \mathbb{Z}^d : \ell \leq x \leq u\}$ est un treillis
(c'est un **sous-treillis** de $\mathbb{Z}^d$, c'est à dire un sous-ensemble de $\mathbb{Z}^d$ stable (ou fermé) pour les opérations $\vee$ et $\wedge$ de $(\mathbb{Z}^d, \leq)$)

Remarques: 1) on définit de même les boîtes (ou rectangles) dans $\mathbb{R}^d$
2) les treillis $(2^E, \subseteq)$, $(\mathbb{T}^E, \Rightarrow)$ et $(\mathbb{B}^E, \leq)$ sont, naturellement, "isomorphes"
3) les sous-treillis de $(2^E, \subseteq)$ sont les anneaux d'ensemble
4) dans tout treillis, on a les équivalences $a \leq b \Leftrightarrow a \vee b = b \Leftrightarrow a \wedge b = a$

Fonctions sous-modulaires dans les treillis et les espaces vectoriels :

Une fonction $f : L \to \mathbb{R}$ est **sous-modulaire** ssi

$$f(a \vee b) + f(a \wedge b) \leq f(a) + f(b) \qquad \forall a, b \in L$$

Caractérisations de la sous-modularité

- dans $\mathbb{Z}^d$: si $(L, \leq)$ est un sous treillis de $\mathbb{Z}^d$, $f : L \to \mathbb{R}$ est sous-modulaire ssi elle satisfait la **propriété d'incréments décroissants** :

$$f(x+e_i+e_j) - f(x+e_j) \leq f(x+e_i) - f(x) \qquad \forall x \text{ tel que } x+e_i \text{ et } x+e_j \in L$$

où $e_i = (0,\dots,0,\underset{\uparrow\, i^e \text{ position}}{1},0\dots,0)^T$ est le i^{eme} vecteur unitaire

exercice : prouver cette équivalence

- dans $\mathbb{R}^d$: $f : \mathbb{R}^d \to \mathbb{R}$ différentiable est sous-modulaire

$$\Leftrightarrow \quad \frac{\partial}{\partial x_i} f(x) \text{ est non-croissante en } x_j \quad \forall i \neq j$$

$$\Leftrightarrow \quad \frac{\partial^2}{\partial x_i \partial x_j} f(x) \leq 0 \quad \forall i \neq j$$

exercice : prouver cette équivalence

Remarque : cette dernière condition montre que la sous-modularité est différente à la fois de la convexité et de la concavité : en effet $f : \mathbb{R}^d \to \mathbb{R}$, deux fois différentiable, est

- sous modulaire ssi son Hessien $Hf(x) = \left(\frac{\partial^2}{\partial x_i \partial x_j} f(x)\right)_{\substack{i=1..d \\ j=1..d}}$ a, pour tout $x \in \mathbb{R}^d$, tous ses termes non-diagonaux qui sont non-positifs. (une propriété indépendante des termes diagonaux $\frac{\partial^2}{\partial x_i^2} f(x)$)
- convexe ssi, $\forall x \in \mathbb{R}^d$, $Hf(x)$ est positif semi-défini (psd)
- concave ssi, $\forall x \in \mathbb{R}^d$, $-Hf(x)$ est psd

(propriétés de toute la matrice $Hf(x)$)

SFMin dans une boîte discrète

étant donnés $l \leq u \in \mathbb{Z}^d$

et $f : B_{l,u} \to \mathbb{Q}$ sous-modulaire, donnée par un oracle de valeur

$$\text{SFMin}(B_{l,u}) : \quad \min \{ f(x) : x \in \mathbb{Z}^d,\ l \leq x \leq u \}$$

- Peut-on résoudre ce problème en temps polynomial (polynomial en d, les tailles d'input de l et u et d'une borne supérieure $M \geq \max\{|f(x)| : x \in B_{l,u}\}$) ?
- La réponse est NON :

Proposition : Tout algorithme pour résoudre $\text{SFMin}(B_{l,u})$ doit utiliser au moins $\sum_{i=1}^{d} (u_i - l_i + 1)$, un nombre pseudo-polynomial, d'appels à l'oracle de valeur.

Preuve : Toute fonction séparable $f = \sum_{i=1}^{d} f_i$ définie sur $B_{l,u}$, c.a.d., $f(x) = \sum_{i=1}^{d} f_i(x_i)$ où chaque $f_i : \{l_i, l_i+1, .., u_i\} \to \mathbb{Q}$ est sous-modulaire

exercice : vérifier cette affirmation

Comme les fonctions f_i peuvent être quelconque, il faut connaître toutes leurs valeurs pour pouvoir en minimiser la somme.

[Plus précisément, on définit la stratégie adverse suivante pour l'oracle de valeur : retourner la valeur $f(x) = d$ pour toute requête $x \in B_{\ell,u}$.
Alors, pour toute séquence de moins de $\sum_{i=1}^{d} f_i(x_i)$ requêtes il existe une coordonnée i et une valeur $v_i \in \{\ell_i, \ell_{i+1}, \dots, u_i\}$ qui n'apparait dans aucune requête. L'algorithme est incapable de différencier les fonctions $f^1 = f_i^1 + \sum_{j \neq i} f_j$ et $f^2 = f_i^2 + \sum_{j \neq i} f_j$ où $f_j(v) = 1$ pour toutes les coordonnées $j = 1..d$ et valeurs v, sauf que $f_i^1(v_i) = 0$ et $f_i^2(v_i) = 2$, et

$$\operatorname{argmin}\{f^1(x) : x \in B_{\ell,u}\} = \{x \in B_{\ell,u} : x_i = v_i\} \text{ alors que}$$

$$\operatorname{argmin}\{f^2(x) : x \in B_{\ell,u}\} = \{x \in B_{\ell,u} : x_i \neq v_i\}$$]

QED

Remarque : cet argument implique aussi une borne supérieure de $(1 - \frac{1}{d-1})$ sur l'approximabilité de SFMin($B_{\ell u}$) lorsque $f \geq 0$

Voici un **algorithme pseudo-polynomial pour SFMin($B_{\ell u}$) :**

- on définit l'**expansion unaire** de chaque coordonnée :

$$x_j = \ell_j + \sum_{k=1}^{w_j} y_{j,k} \quad \text{où } w_j = u_j - \ell_j \text{ et chaque } y_{j,k} \in \mathbb{B} \text{ satisfait}$$

$$y_{j,1} \geq y_{j,2} \geq \dots \geq y_{j,w_j}$$

- soient $E = \{(j,k) : j = 1..d,\ k = 1..w_j\}$ l'ensemble des indices de ces variables $y_{j,k}$

$$\mathcal{F} = \{S \subseteq E : (j,k) \in S \Rightarrow (j,k-1) \in S \quad \forall j = 1..d,\ \forall k = 1 \dots w_j - 1\}$$

$\varphi : \mathcal{F} \to B_{\ell,u}$ où $x = \varphi^{-1}(S)$ a pour composantes

$$x_j = \ell_j + |\{k : (j,k) \in S\}|$$

$$F = f \circ \varphi : \mathcal{F} \to \mathbb{Q} \quad (\text{c.à.d, } F(S) = f(\varphi(S)))$$

exercice : vérifier que

- $\mathcal{F}^i$ est stable pour l'union et l'intersection, donc un anneau d'ensembles
- φ est une bijection, et $S \subseteq T \iff \varphi(S) \leq \varphi(T)$

  donc φ est un isomorphisme de (sous-) treillis
- F est une fonction sous-modulaire sur l'anneau d'ensembles $\mathcal{F}^i$

  et $x \in \operatorname{argmin}\{f : x \in B_{\ell,u}\} \iff \varphi^{-1}(x) \in \operatorname{argmin}\{F(S) : S \in \mathcal{F}^i\}$

On peut donc résoudre SFMin$(B_{\ell,u})$ en temps pseudo-polynomial en résolvant SFMin pour la fonction F sur l'anneau d'ensembles $\mathcal{F}^i$

Référence (dans la liste distribuée avec les Problèmes)

[22] K. Murota (2003) Discrete Convex Analysis (livre)