# Lecture 14

Plan:

1) Global min cut
2) Min T-odd cut.
3) Next time: matroids.

→ Mechthild Stoer

## Stoer-Wagner alg for global mincut.

**Setup**: • Let $G=(V,E)$ undirected graph
• $u: E \to \mathbb{R}_{\geq 0}$ nonnegative edge weights.

---

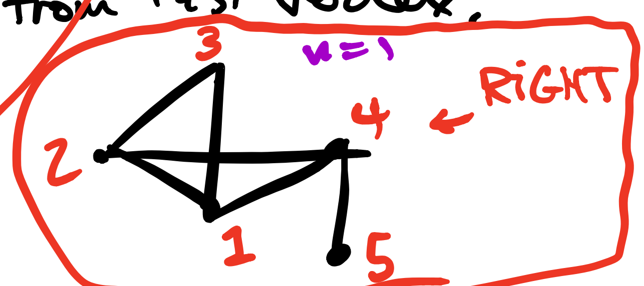
## Algorithm idea:

• starting with vertex, build "max adjacency ordering", greedily adding vertex w/ highest cost to prev vertices.

e.g. $u=1$


WRONG! uses min

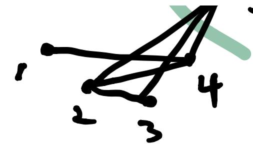
• Consider cut from last vertex.



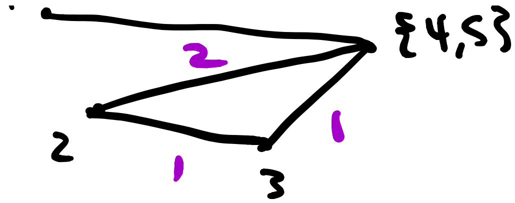

← RIGHT

e.g.

$u=1$




and cuts in



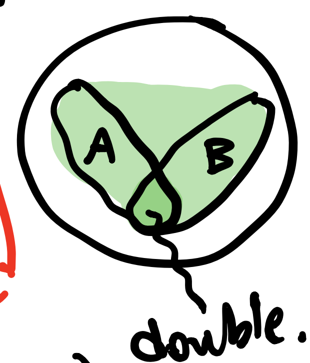

duplicate edges $\rightarrow$ add costs.

- **Claim:** best cut found this way is global min cut.

---

**Def:** For $A, B \subseteq V$, define

$$\boxed{u(A:B) = \sum_{\substack{i \in A \\ j \in B}} u((i,j)).}$$




**Algorithm** (Stoer-Wagner)

MINCUT(G) # outputs cut.

▷ Let $v_1$ any vertex of $G$

▷ $n := |V(G)|$

# create ordering

- initialize $S = \{v_1\}$
- for $i = 2 \ldots n$:
  - $v_i = \arg \underset{v \in V \setminus S}{\cancel{\min}\ \max} \boxed{u(S : \{v\})}$
  - $S \leftarrow S \cup \{v_i\}$
- if $n = 2$:
  - return $\delta(\{v_n\})$
- else:
  - Get $G'$ by shrinking $\{v_{n-1}, v_n\}$.

  # recursive call
  - Let $C = \text{MINCUT}(G')$
  - return less costly of $C$, $\delta(\{v_n\})$.

**Analysis:** uses a claim.

**Claim:** $\boxed{\{v_n\} \text{ is a min } v_{n-1}\text{-}v_n \text{ cut.}}$

**Claim $\Rightarrow$ correctness:**

- The min cut is either a min $v_{n-1}$-$v_n$ cut, or not.
- If it is, claim $\Rightarrow$ alg outputs it ✓
- If not, min cut in $G$ = min cut in $G'$.

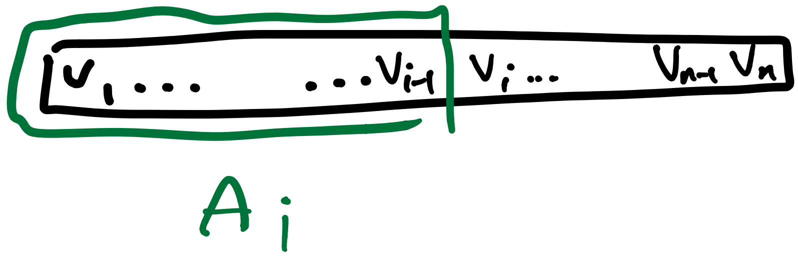
  induction $\Rightarrow$ alg outputs min cut in $G'$ ✓.

---

**Proof of Claim:**

Let $v_1 \; \cdots \; v_n$ be the ordering from alg.

- $A_i :=$ sequence $v_1 \dots v_{i-1}$




- Consider candidate $v_{n-1}$–$v_n$ cut, i.e. $C \subseteq V$ s.t.

$$\boxed{v_{n-1} \in C, \; v_n \notin C}$$

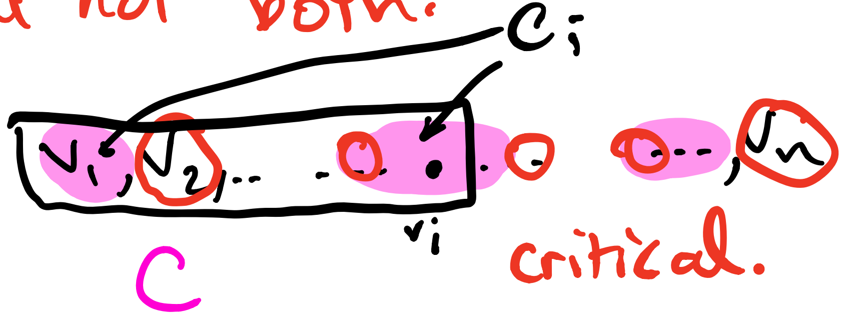
- Want to show

$$\boxed{u(\delta(A_n)) \leq u(\delta(C))}$$

i.e. cut from $\{v_n\}$ is at least as good as $C$.

• define $v_i$ $(i \geq 2)$ to be **critical**

if either $v_i$ or $v_{i-1} \in C$ but not both.



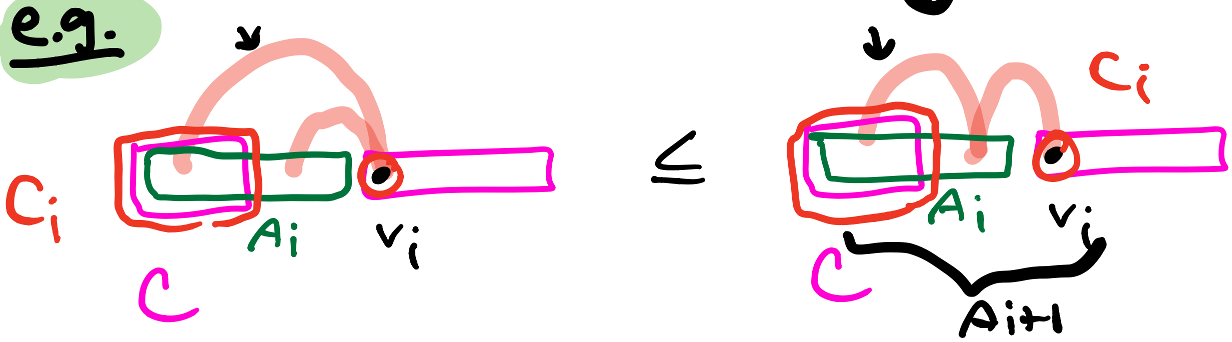

note: $v_n$ is critical b/c $C$ $v_{n-1}$–$v_n$ cut.

• **Subclaim:** Define $C_i := A_{i+1} \cap C$

If $v_i$ critical, then

$$\boxed{u(A_i : \{v_i\}) \leq u(C_i : A_{i+1} \setminus C_i)} \quad ☆$$

**e.g.**



highlighted edge means all edges between sets.

## Subclaim suffices:

$u(S : V \setminus S) = u(\delta(S))$ for any $S \subseteq V$.

because $v_n$ is critical,

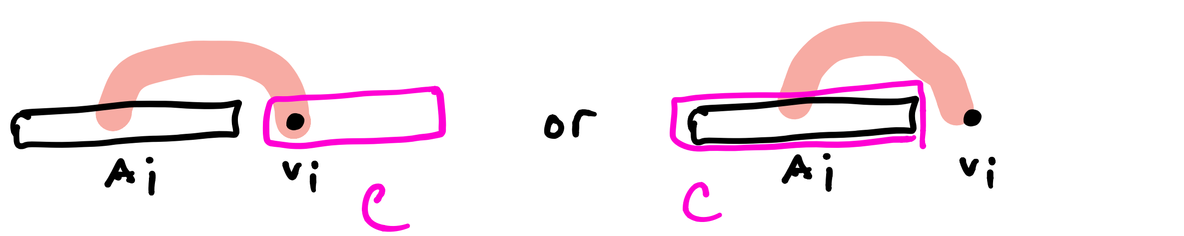
subclaim $\Rightarrow$ $\underbrace{u(\delta(A_n))}_{u(A_n : v_n)} \leq \underbrace{u(\delta(C))}_{u(C_n : A_{n+1} \setminus C_n)}$

✱ for $i = n$ LHS of ✱ RHS of ✱.

## Proof of Subclaim:

- induction on seq. of critical vertices.
- (base:) ✱ true for first critical $v_i$

$v_i$ either first vertex in $C$ or first not in $C$

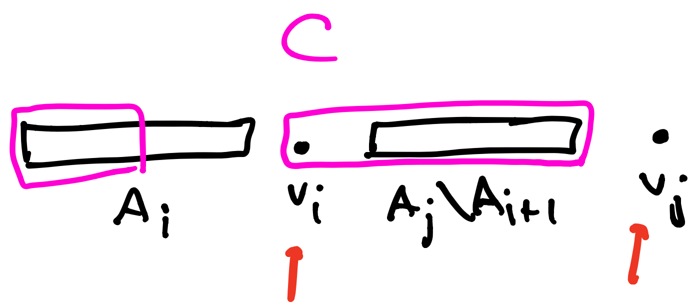



(both LHS & RHS of ✱ are just $u(A_i : \{v_i\})$ )

so ✡ holds with equality.

- (inductive) Assume ✡ true for critical $v_i$, let $v_j$ <u>next</u> critical.

<u>e.g.</u>

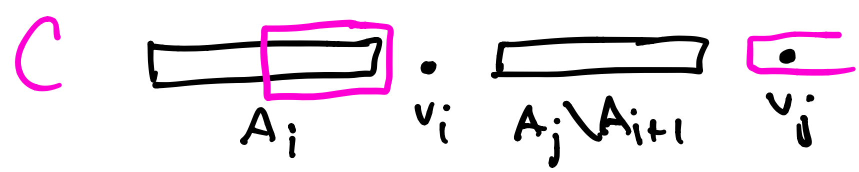


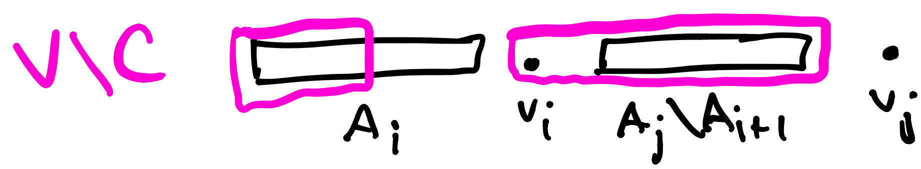

- $\boxed{\text{Assume } v_i \in C,\ v_j \notin C.}$ (as in pic)

<u>Is WLOG</u>: replace $C$ by $V \setminus C$

this preserves the RHS of ✡ (switches $C_j$, $A_{j+1} \setminus C_j$, $u$ symmetric).

<u>e.g.</u>



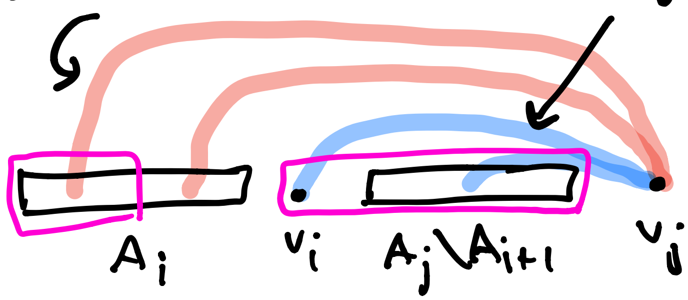

- Then WTS

$(A_i \cup \{v_i\}) \le u(C_j ; A_{j+1} \setminus C_j)$

$$\boxed{u(A_j : \{v_j\}) \leq u(C_j : A_{j+1} \setminus C_j)}$$

LHS

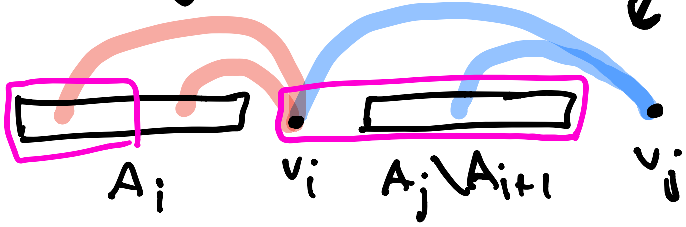
$$u(A_j : \{v_j\}) = u(A_i : \{v_j\}) + u(A_j \setminus A_i : \{v_j\})$$




by our ordering

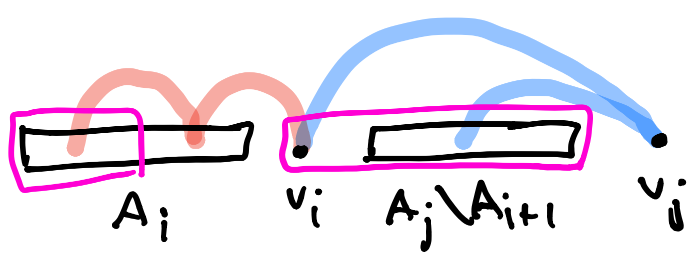
$$\leq u(A_i : \{v_i\}) + u(A_j \setminus A_i : \{v_j\})$$

★ for i



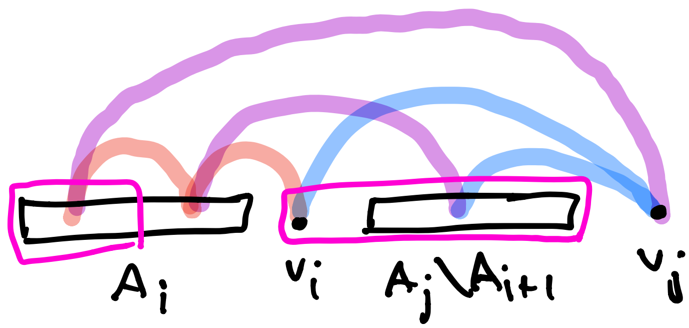

induction

$$\leq u(C_i : A_{i+1} \setminus C_i) + u(A_j \setminus A_i : \{v_j\})$$



$$\le u(C_j : A_{j+1} \setminus C_j) \quad \text{RHS}$$

$v_j$ is next critical.




purple contribution added, nothing removed because $A_j \setminus A_{i+1}$ is in $C$. □

---

## Running time:

Depends how you implement ordering.

- While building ordering, must maintain list of costs $c_i, \dots, c_n$ of remaining verts to $A_i$

- Must quickly find minimum & update new $c_{i+1}, \ldots, c_n$ after picking $v_i$.

- This is what "priority queues" are for, e.g. "Fibonacci heap".

- with Fibonacci heap, can build ordering in $O(m + \log n)$ time.

- total runtime $|\text{calls}| \cdot$
  $O(nm + n\log n)$.

- Compare to $\tilde{O}(nm \cdot n) = \tilde{O}(mn^2)$ from computing $O(n)$ maxflows.

- Negative weight / directed: Hao-Orlin $\tilde{O}(mn)$

# Submodularity

- Stoer-Wagner can be extended to minimize a more general class of functions than $S \mapsto u(\delta(S))$.

- function $f: 2^V \to \mathbb{R}$ **submodular** if $\boxed{f(A) + f(B) \geq f(A \cup B) + f(A \cap B)}$.

- **Examples**:
  - $f(S) = |S|$, "modular" b/c holds w/ equality.
  - $f(S) = u(\delta(S))$ for $u$ nonnegative, even if $G$ directed. E.C. is to prove
  - $V$ = food items on menu, $S \subseteq V$ meal

    $f(S)$ = enjoyment of eating $S$

Submodularity equivalent to "diminishing marginal returns": EC to prove.

For $S \supseteq T$, $v \notin S$,

$$\boxed{f(S+v) - f(S) \leq f(T+v) - f(T)}$$

- Stoer-Wagner algorithm can be extended to minimize any symmetric $(f(S) = f(V \setminus S)\ \forall S \subseteq V)$. submodular function.
  - ↳ Queyranne '95. (we won't cover).

Application of submodularity:

# Minimum T-odd cut

- $G=(V,E)$ undirected
  $u: E \to \mathbb{R}$ nonnegative
  $T \subseteq V$ even size subset.

- **minimum T-odd cut problem:**

$$S = \boxed{\underset{\substack{S \subseteq V \\ |S \cap T| \text{ odd}}}{\arg\min}\ u(\delta(S))}$$

- Say $S$ is T-odd if $|S \cap T|$ odd.

- Note: $S$ T-odd $\Leftrightarrow V \setminus S$ T-odd.

- why do we care? Matching polytope!

Recall

**THM (Edmonds)** Let

$X = \{1_M : M \text{ matching in } G\}$.

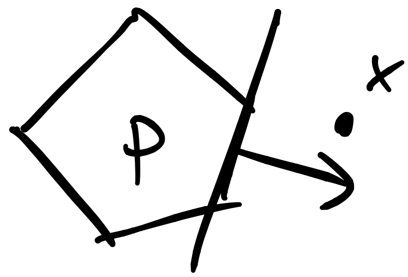
Then $\text{conv}(X) = P$ where

$$P = \left\{ x : \begin{array}{ll} \sum_{e \in \delta(v)} x_e \le 1 & \forall v \in V. \\ \sum_{e \in E(S)} x_e \le \frac{|S|-1}{2} & \forall S \subseteq V \\ & |S| \text{ odd} \\ x_e \ge 0 & \forall e \in E. \end{array} \right\}$$

- How can we quickly test if $x \in P$? exponential # of constraints!

- Padberg-Rao: Can express as min odd cut problem.

▷ what's more, can get separating hyperplane if $x \notin P$.




- This can be used to optimize over P via ellipsoid alg, despite there being no polynomial size LP for optimising over P (Rothvoss).

matching polytope.

- **Today:** ▷ poly. time alg. for min T-odd cut
  ▷ crucially uses submodularity.

## Algorithm ALG(G, T)

1) Find min cut among those with at least one vertex of T on each side:

$$S = \boxed{\underset{\emptyset \neq S \cap T \neq T}{\arg\min} \; u(\delta(S))}.$$

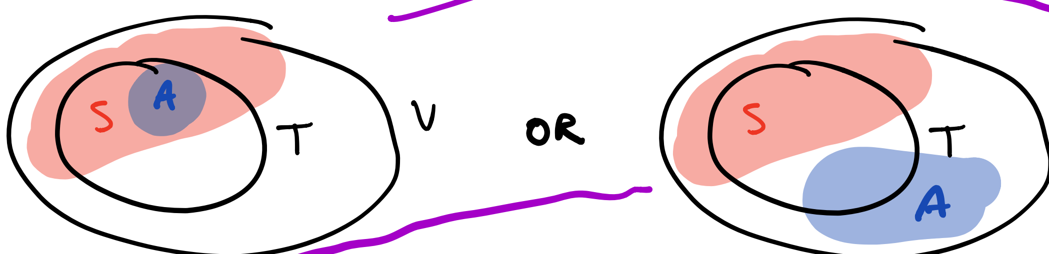
- Takes $|T|-1$ min $s$-$t$ cut computations: fix $s$ arbitrary in $T$, compute min $s$-$t$ cut for all $t \in T$.


2) If $S$ is $T$-odd cut, $|S \cap T|$ odd, is minimum, return $S$.

**Else:** If $S$ $T$-even, use

**Lemma:** If $S$ as in ☆, $|S \cap T|$ even, $\exists$ min $T$-odd cut $A$ w/ $A \subseteq S$ or $A \subseteq V \setminus S$.

e.g.




Pf: after alg. (uses submod.)

- Lemma $\Rightarrow$ 2 recursive calls suffice:

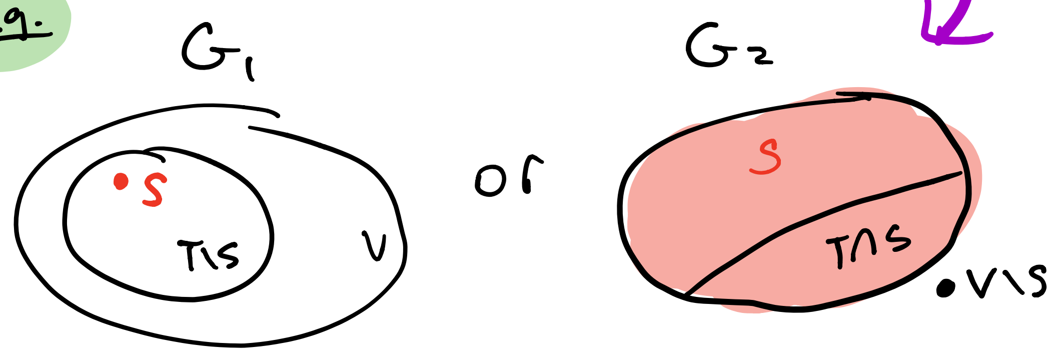
▷ $G_1 := G/S$ (shrink $S$ to single vertex)

$T_1 := T \setminus S$ remove $S$ from $T$

▷ $G_2 := G/(V\setminus S)$

$T_2 := T/(V\setminus S) = T\cap S$

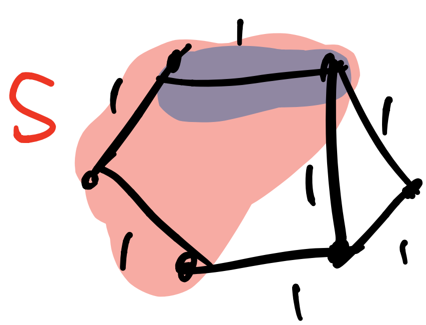
e.g.




**Return:**

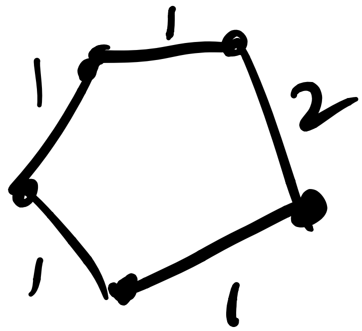
$\min\{ALG(G_1, T_1), ALG(G_2, T_2)\}$.

## e.g. of recursion




$\min_{A\subseteq S} \delta(A)$.

$G_2$ =




## Running time

why poly time if 2 recursive calls?

$R(k) :=$ $\boxed{\text{largest possible runtime with } |T| = k.}$

$(|V| \le n)$.

Then

a) $R(2) = \tau :=$ time for min s-t cut.

b) $R(k) \le \max\limits_{\substack{k_1 \geq 2 \\ k_2 \geq 2 \\ k_1 + k_2 = k}} (k-1)\tau + R(k_1) + R(k_2)$

sizes of $S \cap T$, $T \setminus S$

Step 1

recursive calls.

By induction, $\boxed{R(k) \le k^2 \tau}$

Pf: • base: True for $k=2$

• Inductive:

$$R(k) \le \max_{\substack{k_1 \geq 2 \\ k_2 \geq 2 \\ k_1 + k_2 = k}} \left((k-1)\tau + R(k_1) + R(k_2)\right)$$

induction & $\max\limits_{\substack{k_1 \geq 2 \\ k_2 \geq 2 \\ k_1 + k_2 = k}} k_1^2 + k_2^2 = 4 + (k-2)^2$.

$$\le (k-1)\tau + 4\tau + (k-2)^2 \tau$$

$$= (k^2 - 3k + 7)\tau$$

$\leq k^2 \tau$

($k \geq 4$ b/c $k$ even, $k \geq 2$).

**Thus:** algorithm is polynomial.

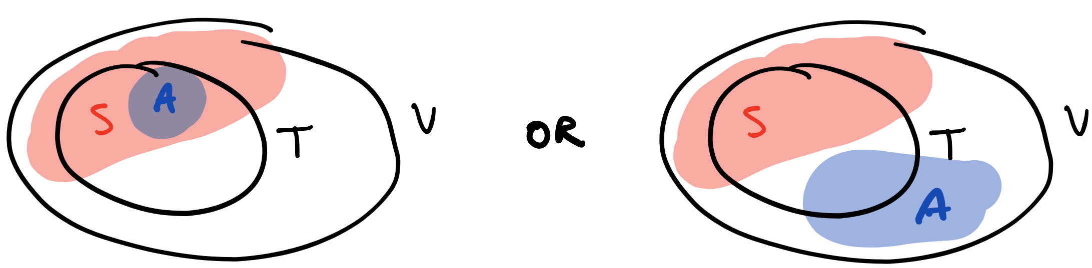
- Now for the lemma.
- Proof uses submodularity.

---

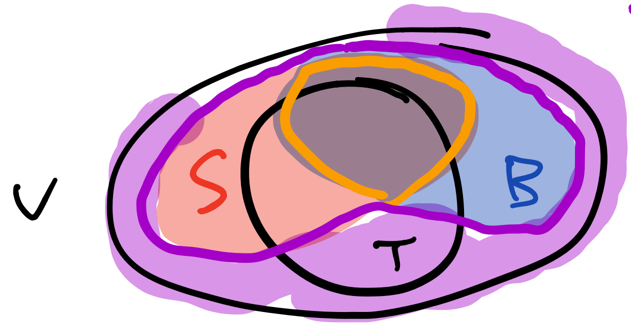
Recall: $|T|$ even.

**Lemma:** Let $S$ min cut of $G$ subject to $\emptyset \neq S \cap T \neq T$, $|S \cap T|$ even, then

$$\boxed{\begin{array}{l} \exists \text{ min } T\text{-odd cut } A \text{ with} \\ \qquad A \subseteq S \\ \text{or} \quad A \subseteq V \setminus S. \end{array}}$$



**Proof** • Let $B$ be any minimum $T$-odd cut.




• candidates for $A$.

• We'll show we can take

$$\boxed{A = S \cap B \quad \underline{\text{OR}} \quad A = S \cup B}$$

use $V \setminus A$ as min $T$-odd cut.

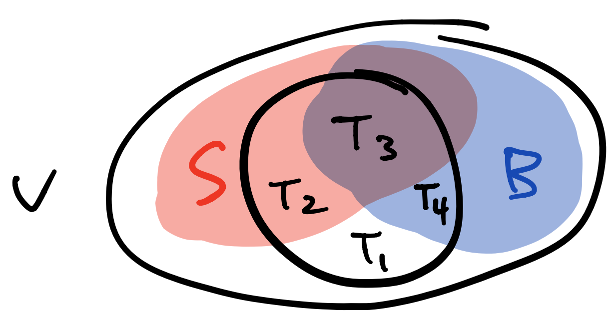
• Make a partition of $T$.

$T_1 = T \setminus (B \cup S)$, $T_2 = (T \cap S) \setminus B$

$T_3 = T \cap B \cap S$, $T_4 = (T \cap B) \setminus S$

and $V \setminus A$

$\underline{\subset} V \setminus S$.



- By definition of $B, S$, know all pairwise unions nonempty:

$$T_1 \cup T_4 \neq \emptyset \quad T_2 \cup T_3 \neq \emptyset \Big\} \ \emptyset \neq S \cap T \neq T$$

$$T_2 \cup T_1 \neq \emptyset \quad T_3 \cup T_4 \neq \emptyset \Big\} \ |B \cap T| \text{ odd}, \ |T| \text{ even} \Rightarrow \emptyset \neq B \cap T \neq T.$$

$\Rightarrow$ either $T_1$ and $T_3$ nonempty or $T_2$ and $T_4$ nonempty.

- By possibly replacing $B \leftarrow V \setminus B$, may assume $T_1$ and $T_3$ nonempty.
(replacing $B \leftarrow V \setminus B$) $T_2 \leftrightarrow T_3$, $T_4 \leftrightarrow T_1$)

- submodularity of cut $\Rightarrow$

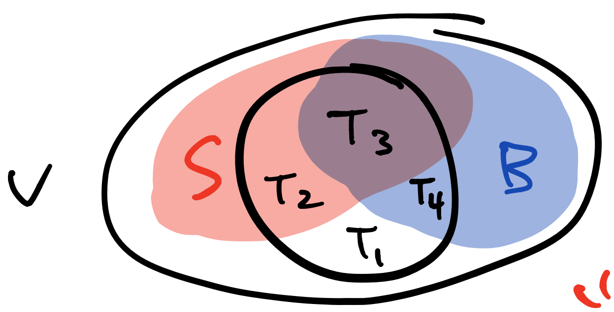
$$u(\delta(S)) + u(\delta(B))$$

$$(\Delta) \quad \geq \underbrace{u(\delta(S\cup B))}_{< u(\delta(S)).} + \underbrace{u(\delta(S\cap B))}_{\text{imagine odd but } > u(\delta(B)).}.$$

$\Leftarrow$

* b/c $S$ minimal.

⊙ As $T_1 \neq \emptyset$ & $T_3 \neq \emptyset$, $S\cap B$ and $S\cup B$ separate vertices of $T$.




(both $S\cap B$, $S\cup B$ still "candidates for $S$")

⊙ One of $S\cup B$, $S\cap B$ is T-even & the other T-odd because

$$|(S\cap B)\cap T| + |(S\cup B)\cap T| = |T_2| + 2|T_3| + |T_4|$$
$$= |S\cap T| + |B\cap T| = \text{is odd.}$$ (b/c $S$ T-even

(R−T odd).

Summary: one of $S \cup B$, $S \cap B$ is candidate "S", one is candidate "B".

- $\Delta \Rightarrow$ whichever of $S \cup B$, $S \cap B$ odd has cut value $\leq u(\delta(B))$ the other $\leq u(\delta(S))$.

see orange *

(else the other violates minimality of $B$ or $S$ due to $\Delta$).

$\Rightarrow$ Either $S \cup B$, $S \cap B$ (whichever is odd). is min T-odd cut. $\square$