Approximation

There are two kinds of problems

- decision problems: does a solution exist?
- optimization problems: what is an optimal solution?
  - maximization
  - minimization

Optimization problems can also be in P, NP, etc.

Approximation

Intractable optimization problems can be attacked using *heuristics*.

Origin: Greek word meaning "we don't actually know what we're doing here . . ."

We are interested in approximation techniques that can be formally analyzed.

Approximation

A *maximization (minimization) problem* consists of

- set of *feasible solutions*
- each feasible solution A has a *cost* $c(A)$
- solution with max (min) cost is *optimal*.

Definition: An $\epsilon - approximation$ algorithm A is one that

$$\frac{c(A)}{OPT} \geq 1 - \epsilon \qquad \textit{(maximization)}$$

$$\frac{OPT}{c(A)} \geq 1 - \epsilon \qquad \textit{(minimization)}$$

Note that $0 \leq \epsilon \leq 1$.

Approximation

Question: What is the smallest ϵ for which a given NP-complete problem has a *polynomial-time* ϵ-approximation?

Remark: Polynomial reduction does not necessarily preserve approximations.

NP-complete problems may have

- no ϵ-approximation, for any ϵ.
- an ϵ-approximation, for some ϵ.
- an ϵ-approximation, for any ϵ.

Not all NP-complete problems are created equal.

Node Cover

Given a graph (V, E)

- find the smallest set of nodes C
- such that for each edge in the graph,
- C contains at least one endpoint.

This problem is NP-complete.

The Greedy Heuristic

Remark: A node with high degree looks “useful” for a node cover.

- $C := \emptyset$
- while there are edges in G
    - choose node $v \in G$ with highest degree
    - add it to C
    - remove it from G

Question: How are we doing?

The Greedy Heuristic

Question: How are we doing?

Answer: Poorly. The Greedy heuristic is not an ϵ-approximation algorithm for any ϵ. Error may grow logarithmically with number of nodes.

Another Heuristic

- $C := \emptyset$
- while there are edges in G
  - choose any edge (u, v) in G
  - add u and v to C
  - remove them from G

Claim: This algorithm is a $\frac{1}{2}$-approximation algorithm for node cover.

Another Heuristic

Claim: This algorithm is a $\frac{1}{2}$-approximation algorithm for node cover.

- C contains $\frac{1}{2|C|}$ edges of G
- no two of which share a vertex
- any node cover, including the optimum,
- contains at least one node from each of these edges
- (otherwise an edge would not be covered)

It follows that $\frac{OPT(G) \geq 1}{2|C|}$.

Remark: This is the best known approximation to vertex cover.

MAXSAT

Given a CNF formula, find the truth assignment that satisfies the *most* terms.

Let

$$\Phi = \phi_1 \wedge \cdots \wedge \phi_m$$

Strategy: convenient to use probabilistic reasoning, even though our algorithm will be deterministic.

MAXSAT

Definition:

$$p(\phi_i) = \frac{\#\text{ satisfying assignments}}{\#\text{ assignments}}$$

Also the probability ϕ_i will be satisfied by a random assignment.

If ϕ_i has k variables, then

$$\frac{p(\phi_i) = 1 - \frac{1}{2^k} \geq 1}{2}$$

because only one assignment falsifies ϕ_i. **Which?**

MAXSAT

Definition:

$$p(\Phi) = \sum_{i=1}^{m} p(\phi_i).$$

Also the expected *number* of terms satisfied by a random assignment.

If ϕ_i has k_i variables,

$$p(\Phi) = \sum_{i=1}^{m} (1 - \frac{1}{2^{k_i}}) = m - \sum_{i=1}^{m} \frac{1}{2^{k_i}} \geq \frac{m}{2}$$

MAXSAT

Suppose

- we set $x_1 = 0$ in Φ
- now we have a formula in $x_2, \ldots, x_m$
- similarly if we set $x_1 = 1$
- call these $\Phi[x_1 = 0]$ and $\Phi[x_1 = 0]$

Claim:

$$\frac{p(\Phi) = 1}{2\left(p(\Phi[x_1 = 0]) + p(\Phi[x_1 = 1])\right)}$$

MAXSAT

$$\frac{p(\Phi) = 1}{2\,(p(\Phi[x_1 = 0]) + p(\Phi[x_1 = 1]))}$$

Algorithm

- Set x_1 to the t with larger $p(\Phi[x_1 = t])$.
- Note that $p(\Phi) \leq p(\Phi[x_1 = t])$.
- Continue with each subsequent variable.

At the end,

- each term is either 0 or 1
- each $p(\phi_i[x_1 = t_1, \ldots, x_m = t_m])$ is 0 or 1,
- $p(\Phi[x_1 = t_1, \ldots, x_m = t_m]) =$ # satisfied terms

MAXSAT

$p(\Phi[x_1 = t_1, \ldots, x_m = t_m])$ is the number of satisfied terms.

Recall that

$$p(\Phi[x_1 = t_1, \ldots, x_m = t_m]) \geq p(\Phi) \geq \frac{m}{2}.$$

Notice that the optimal number of satisfied terms $\leq m$.

The ratio of actually satisfied to optimally satisfied is $\geq \frac{1}{2}$.

MAXSAT

Claim: Algorithm described is $\frac{1}{2}$-approximation to MAXSAT.

Claim: If each term has $\geq k$ variables, then algorithm described is $\frac{1}{2^k}$-approximation to MAXSAT.

Max Cut

Partition the nodes of $G = (V, E)$ into two sets S and $V - S$ to maximize the number of edges between S and $V - S$.

This problem is NP-complete.

Local Improvement

Consider the following heuristic

- Pick any partition S and $V - S$
- If the cut can be improved by moving any vertex from $V - S$ to S, or vice-versa, do so.
- Quit when no improvement is possible.

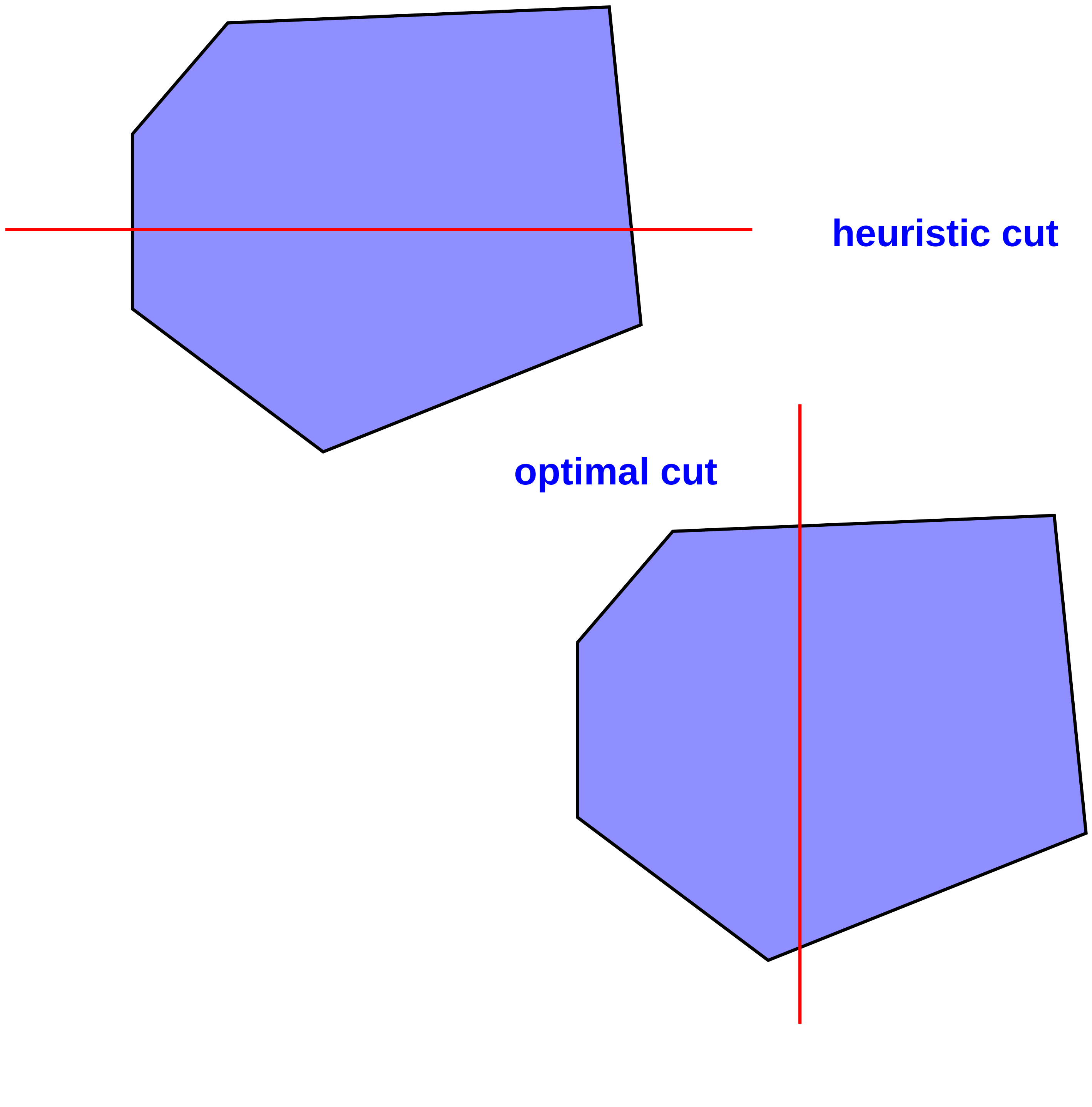
Running time

- max cut has at most $|E|$ edges
- at most $|E|$ improvements possible

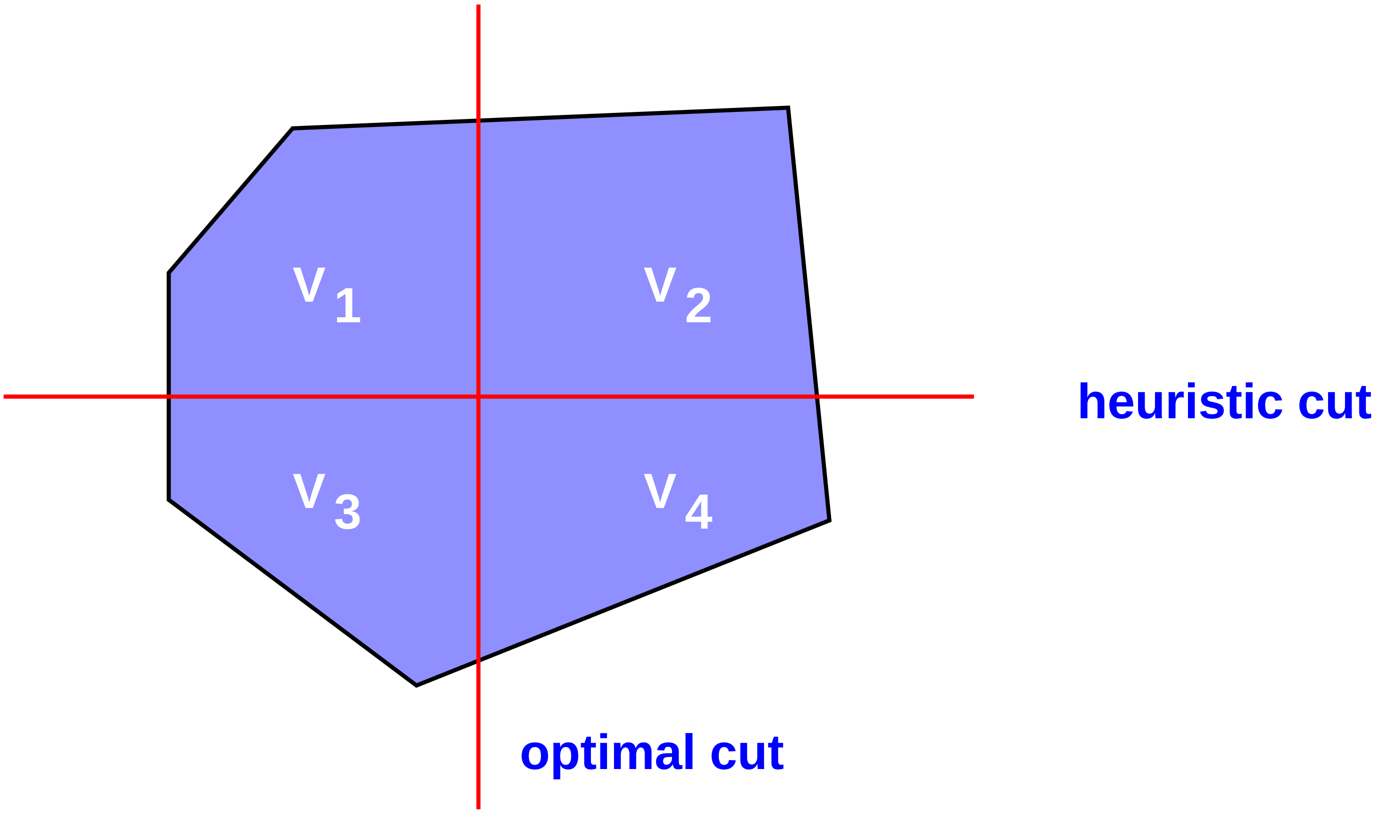
Heuristic is polynomial.

Claim: This algorithm is a $\frac{1}{2}$-approximation.

MAXCUT

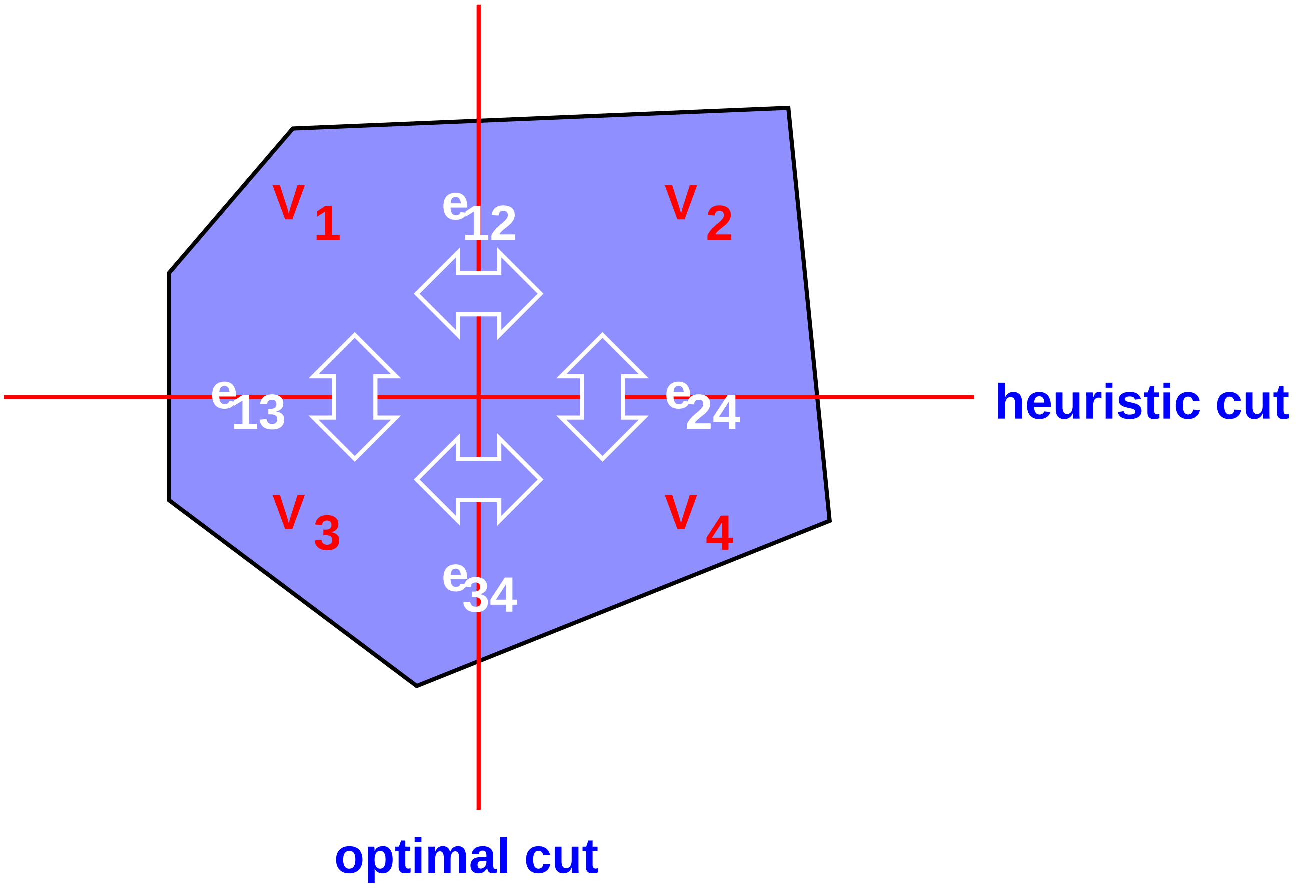


MAXCUT



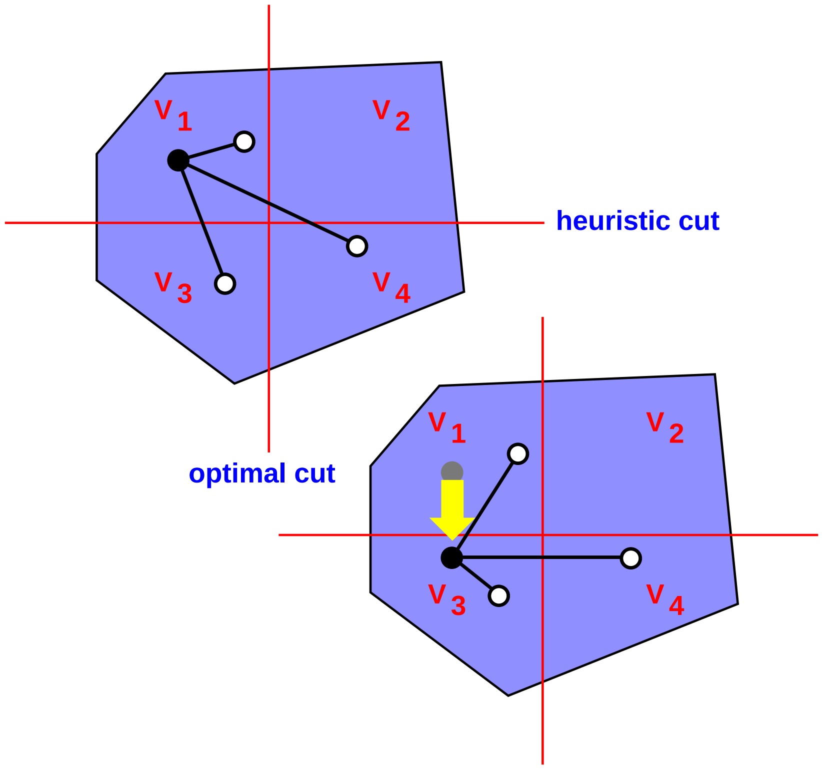
- Heuristic yields $V_1 \cup V_2$, $V_3 \cup V_4$
- Optimal yields $V_1 \cup V_3$, $V_2 \cup V_4$

MAXCUT



Let e_{ij} be the number of edges from V_i to V_j.

MAXCUT



For each node in V_1, edges to V_1 and V_2 outnumbered by edges to V_3 and V_4.

Summing over all nodes in V_1:

$$2e_{11} + e_{12} \leq e_{13} + e_{14}$$

MAXCUT

Summing over all nodes in V_1:

$$\begin{aligned} 2e_{11} + e_{12} &\leq e_{13} + e_{14} \\ e_{12} &\leq e_{13} + e_{14} \end{aligned}$$

Symmetrically,

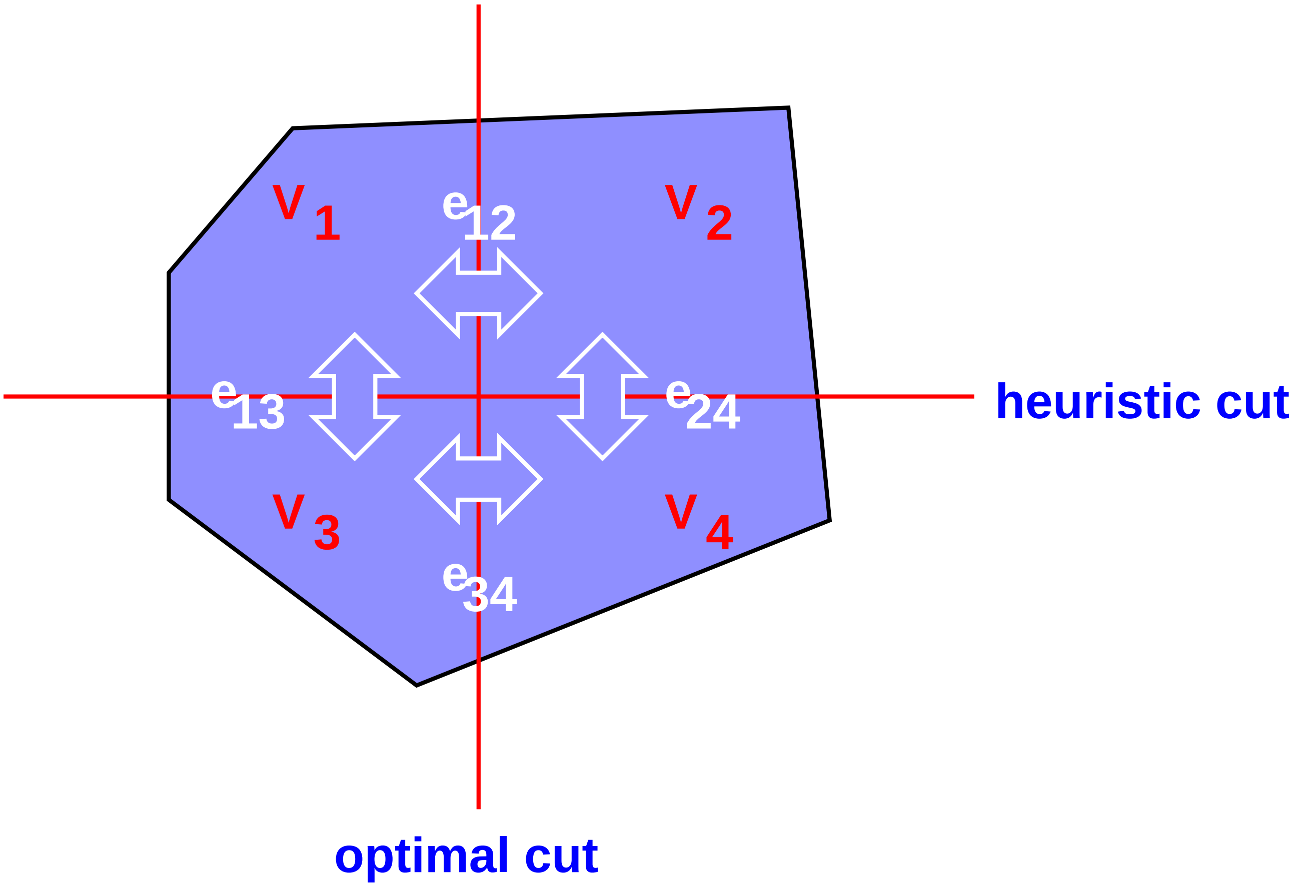
$$\begin{aligned} e_{12} &\leq e_{13} + e_{14} \\ e_{12} &\leq e_{23} + e_{24} \\ e_{34} &\leq e_{23} + e_{13} \\ e_{34} &\leq e_{14} + e_{24} \end{aligned}$$

After extensive (but boring) manipulation

$$e_{12} + e_{34} + e_{14} + e_{23} \leq 2(e_{13} + e_{14} + e_{23} + e_{24})$$

LHS is optimal, RHS is our approximation.

MAXCUT



$$e_{12} + e_{34} + e_{14} + e_{23} \leq 2(e_{13} + e_{14} + e_{23} + e_{24})$$

LHS is optimal, RHS is our approximation.

Theorem: There is a $\frac{1}{2}$-approximation to MAXCUT.