Primal-Dual Methods

David P. Williamson

Today's tool is an adaptation of a fundamental tool in the design of algorithms for linear programming and combinatorial optimization: the primal-dual method.

The primal-dual method gives classic algorithms for:

- Linear Programming
- Assignment Problem
- Maximum Flow
- Minimum-cost Flow
- Shortest s-t path
- $\cdots$

The Primal-dual method uses complementary slackness conditions.

$$\begin{array}{ll} \text{Min } \sum_i c_i x_i \\ \sum_i a_{ij} x_i \geq b_j & \forall j \\ x \geq 0 \end{array} \qquad \begin{array}{ll} \text{Max } \sum_j b_j y_j \\ \sum_j a_{ij} y_j \leq c_i & \forall i \\ y \geq 0 \end{array}$$

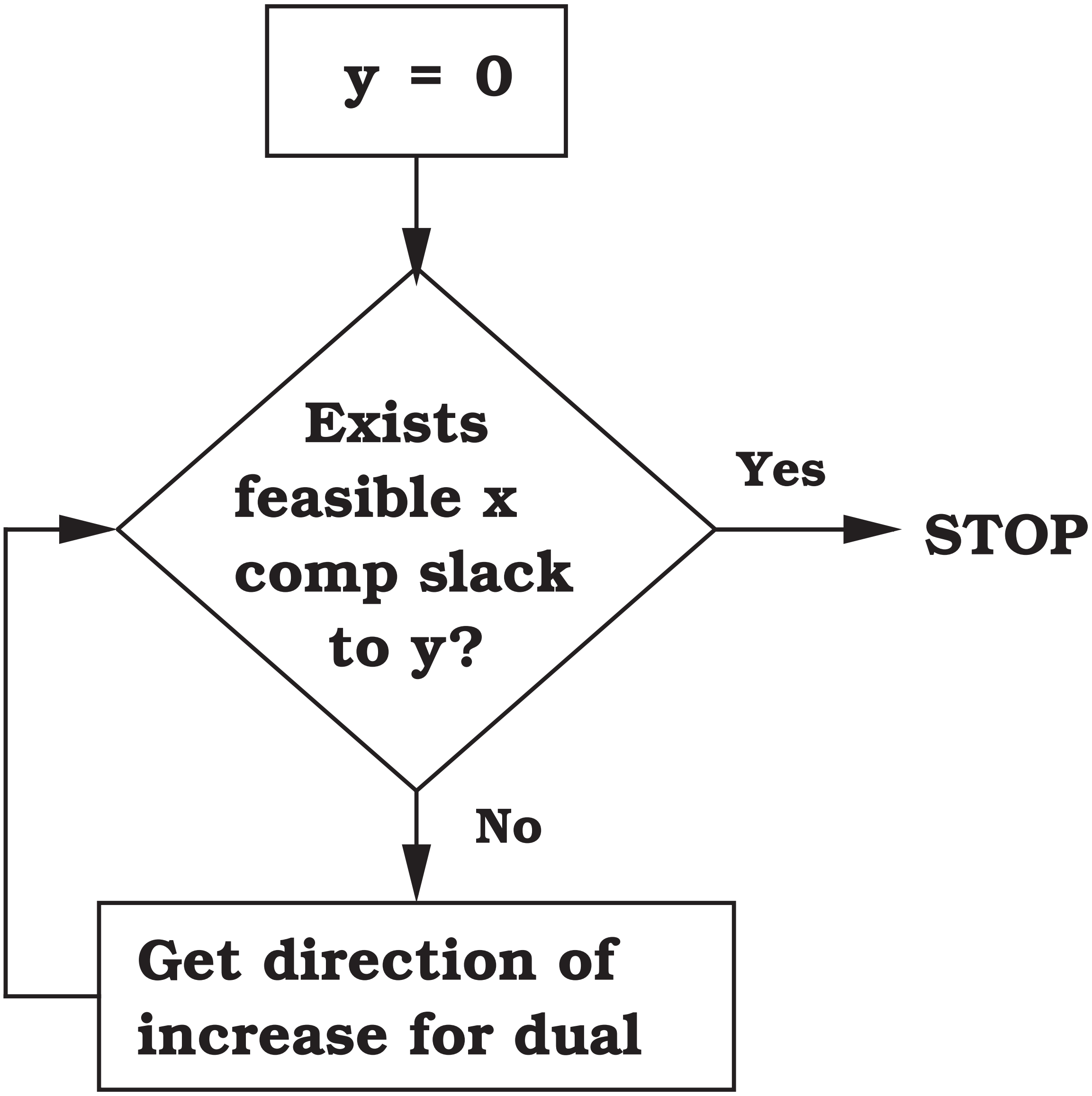
Primal complementary slackness

$$x_i > 0 \Rightarrow \sum_j a_{ij} y_j = c_i$$

Dual complementary slackness

$$y_j > 0 \Rightarrow \sum_i a_{ij} x_i = b_j$$

If the conditions are obeyed for feasible solutions x, y, then solutions are optimal.



y = 0
Exists feasible x comp slack to y?
Yes
STOP
No
Get direction of increase for dual

Long known

The primal-dual method can be used to solve optimization problems that can be formulated as linear programs.

Recent years

A variation of the primal-dual method can be used to create good approximation algorithms.

Today

A single generic primal-dual algorithm and analysis that encompasses many known algorithms

$\underline{\alpha = 1}$

Shortest s-t path	Dijkstra '59
Min-cost branching	Edmonds '67

$\underline{\alpha = 2}$

Vertex cover	Bar-Yehuda, Even '81
Steiner tree	various
Generalized Steiner tree	Agrawal, Klein, Ravi '91
Matching (w/ triangle inequality)	Goemans, W '92
T-joins	$\vdots$
Non-fixed point-to-point connection	
Partition into trees/paths/cycles of at least k vertices	
Location/design problems	
Prize-collecting TSP	
$\vdots$	
Multicuts in trees	Garg, Vazirani, Yannakakis '93

Other α

Survivable network design	W, Goemans, Mihail, Vazirani '93
	Goemans, Goldberg, Plotkin, Shmoys, Tardos, W '94

The generic algorithm is for the hitting set problem.

Input:	Ground set E
	Costs $c_e \geq 0$, $\forall e \in E$.
	Sets $T_i \subseteq E$, $i = 1, \ldots, p$.
Output:	Minimum-cost $A \subseteq E$ s.t. $A \cap T_i \neq \emptyset$

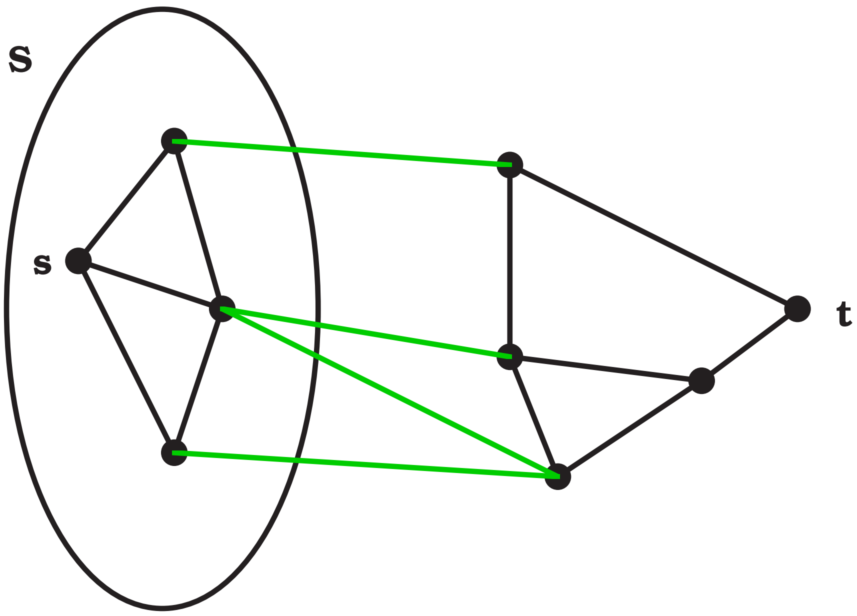
Example Minimum-weight vertex cover on $G = (V, E)$

Given weights w_i on $i \in V$, find a min-cost subset S of V such that at least one endpoint of every edge is in S.

Ground set: V
Costs: w_i, $i \in V$
Sets: $\{u, v\}$, where $(u, v) \in E$

Example Shortest s-t path in $G = (V, E)$

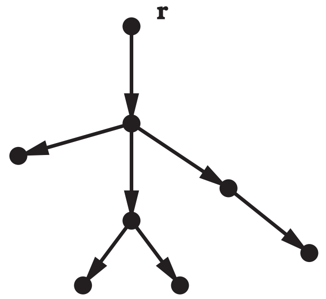
Ground set: E
Costs: $c_e \geq 0$, $\forall e \in E$
Sets: $\delta(S)$, $\quad \forall S \subset V$ such that $s \in S, t \notin S$
$\delta(S) \equiv \{(u, v) \in E : u \in S, v \notin S\}$



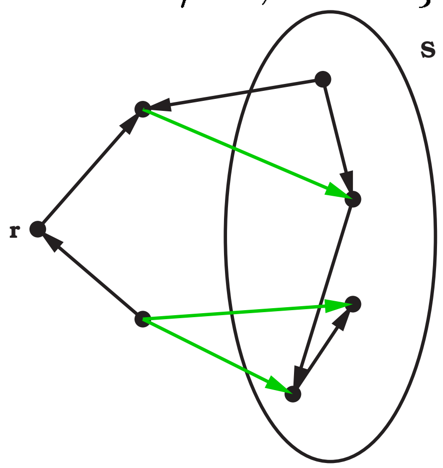
The hitting set models this problem via a max-flow/min-cut argument.

Example Minimum-cost branching

Given a directed graph $G = (V, A)$, root $r \in V$, find a min-cost subgraph such that there is a directed path from r to every other vertex.



Ground set: A
Costs: c_a, $\forall a \in A$
Sets: $\delta^-(S)$, $\quad \forall S \subset V, r \notin S$
$\delta^-(S) = \{(u, v) \in A : u \notin S, v \in S\}$



A formulation of the hitting set problem:

$$\text{Min} \quad \sum_{e \in E} c_e x_e$$

subject to:

$$\sum_{e \in T_i} x_e \geq 1 \qquad i = 1, \ldots, p$$

$$x_e \in \{0, 1\} \qquad e \in E,$$

The dual of the relaxation:

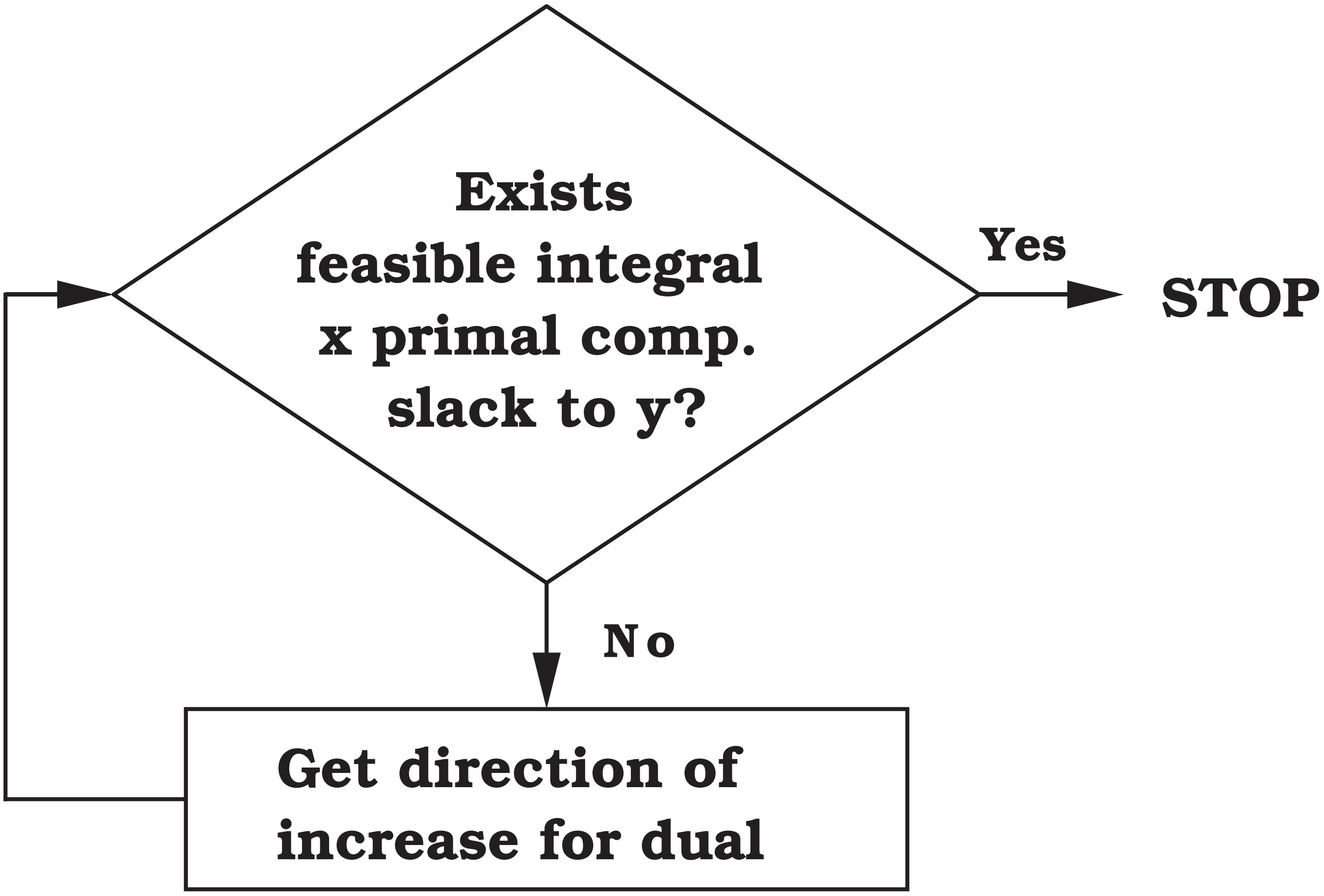
$$\text{Max} \quad \sum_{i=1}^{p} y_i$$

subject to:

$$\sum_{i : e \in T_i} y_i \leq c_e \qquad e \in E$$

$$y_i \geq 0 \qquad i = 1, \ldots, p.$$

The Main Idea for primal-dual approximation algorithms: relax dual complementary slackness conditions.



Appropriate relaxations of the dual conditions lead to good approximation algorithms.

An easy way to get an integral solution obeying primal complementary slackness conditions:

$$A = \{e \in E : \sum_{i : e \in T_i} y_i = c_e\}.$$

If A is not feasible, then there exists a violated set T_j (i.e., $A \cap T_j = \emptyset$).

$$\Rightarrow \forall e \in T_j, \sum_{i : e \in T_i} y_i < c_e.$$

Therefore, we can increase the dual objective by increasing y_j.

Algorithm (Bar-Yehuda, Even)

$y \leftarrow 0$

$A \leftarrow \emptyset$

While A is not feasible

Pick violated set T_j

Increase y_j until $\sum_{i:e \in T_i} y_i = c_e$ for some $e \in T_j$

$A \leftarrow A \cup \{e\}$

Analysis of the Algorithm

Let A_f denote the output of the algorithm.

$$\sum_{e \in A_f} c_e = \sum_{e \in A_f} \sum_{i : e \in T_i} y_i = \sum_i |A_f \cap T_i| y_i$$

If we can ensure that $y_i > 0$ implies $|A_f \cap T_i| \leq \alpha$, then

$$\sum_{e \in A_f} c_e \leq \alpha \sum_i y_i \leq \alpha \cdot OPT,$$

i.e., we have an α-approximation algorithm.

Example For vertex cover $|A_f \cap T_i| \leq 2$, which implies a 2-approximation alg. (Bar-Yehuda, Even).

In the '90s, two new ideas are introduced.

Idea #1: "Clean up" final solution by deleting elements in reverse of the order added.

- Saran, Vazirani, Young '92
  Minimum-cost 2-connected graphs

- Klein, Ravi '93
  Minimum-cost generalized 2-connectivity problems

- Williamson, Goemans, Mihail, Vazirani '93
  Minimum-cost general connectivity problems

- Becker, Geiger '94 and Berman, Fujito, Bafna '95
  Minimum feedback vertex sets in undirected graphs

Algorithm (Take 2)

$y \leftarrow 0$

$A \leftarrow \emptyset$

$l \leftarrow 0$

While A is not feasible

$l \leftarrow l + 1$

Pick violated set T_j

Increase y_j until $\sum_{i: e \in T_i} y_i = c_e$ for some $e_l \in T_j$

$A \leftarrow A \cup \{e_l\}$

For $j \leftarrow l$ downto 1

If $A - \{e_j\}$ feasible

$A \leftarrow A - \{e_j\}$

Theorem If for any infeasible A and any *minimal* augmentation B of A to a feasible solution,

$$|B \cap T(A)| \leq \beta,$$

where $T(A)$ is the violated set chosen by the algorithm, the algorithm has a performance guarantee of β.

The condition of the theorem implies that

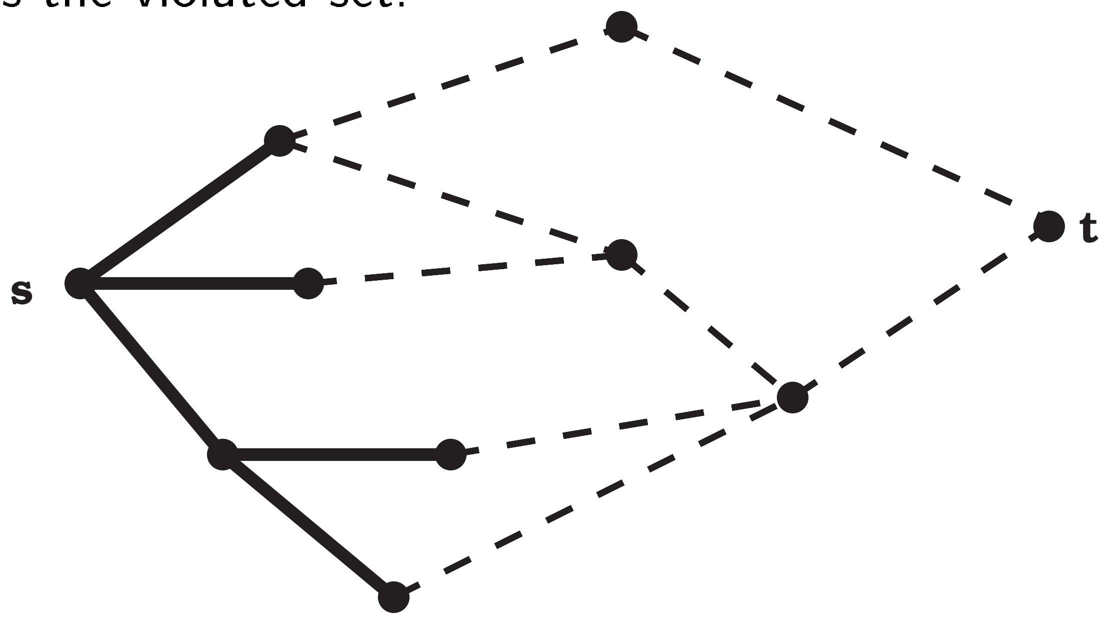
$$y_i > 0 \Rightarrow |A_f \cap T_i| \leq \beta$$

for the final solution A_f (after clean-up). So then it follows that

$$\sum_{e \in A_f} c_e = \sum_i |A_f \cap T_i| y_i \leq \beta \sum_i y_i \leq \beta \cdot OPT.$$

Example: Shortest s-t path problem

Choose the cut of the connected component containing s as the violated set.

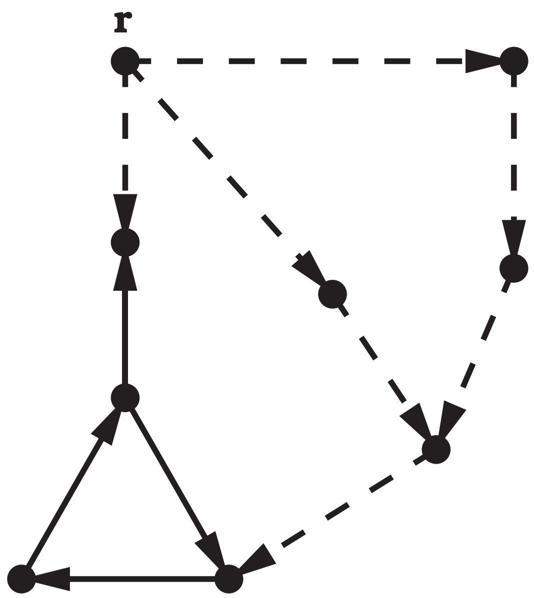


Then any minimal augmentation to an s-t path hits the cut once.

This implies a 1-approximation algorithm.
In fact, it's just Dijkstra's algorithm.

Example: Minimum-cost branchings

Choose the cut of any strongly-connected component not containing r as the violated set.



Then any minimal augmentation to a branching hits the cut once.

This implies a 1-approximation algorithm.
In fact, it's just Edmonds' algorithm.

Improving the primal-dual method

Idea #2: Increase duals on several violated sets simultaneously.

- Agrawal, Klein, Ravi '91
  - Generalized Steiner trees
- Goemans, Williamson '92
  - T-joins
  - Matching (with triangle inequality)
  - Non-fixed point-to-point connection problem
  - Location/design problem
  - Prize-collection TSP
  - Various partitioning problems
  - others...

Algorithm (Take 3)

$y \leftarrow 0$

$A \leftarrow \emptyset$

$l \leftarrow 0$

While A is not feasible

  $l \leftarrow l + 1$

  $\mathcal{V} \leftarrow \text{Violated}(A)$

  Increase y_j for $T_j \in \mathcal{V}$ uniformly until $\sum_{i:e \in T_i} y_i = c_e$ for some $e_l \in T_j \in \mathcal{V}$

  $A \leftarrow A \cup \{e_l\}$

For $j \leftarrow l$ downto 1

  If $A - \{e_j\}$ feasible

    $A \leftarrow A - \{e_j\}$

Theorem If for any infeasible A and any *minimal* augmentation B of A to a feasible solution,

$$\sum_{T_i \in \mathrm{Violated}(A)} |B \cap T_i| \leq \rho \cdot |\mathrm{Violated}(A)|,$$

where $\mathrm{Violated}(A)$ are the violated sets chosen by the algorithm given A, the performance guarantee of the algorithm is ρ.

Example: Network design problems on undirected graphs corresponding to a class of functions $f : 2^V \rightarrow \{0,1\}$.

Ground set:	E
Costs:	$c_e \quad \forall e \in E$
Sets:	$\delta(S) \quad \forall S \subseteq V$ s.t. $f(S) = 1$

For this class of functions, one can get $O(n^2 \log n)$ time 2-approximation algs. (Goemans, Williamson '92)

- Matching (given triangle inequality)
- T-joins
- Steiner tree
- Generalized Steiner tree
- Non-fixed point-to-point connection
- Exact partitioning problems
- ...

We will focus for now on the minimum-cost perfect matching problem given the triangle inequality.

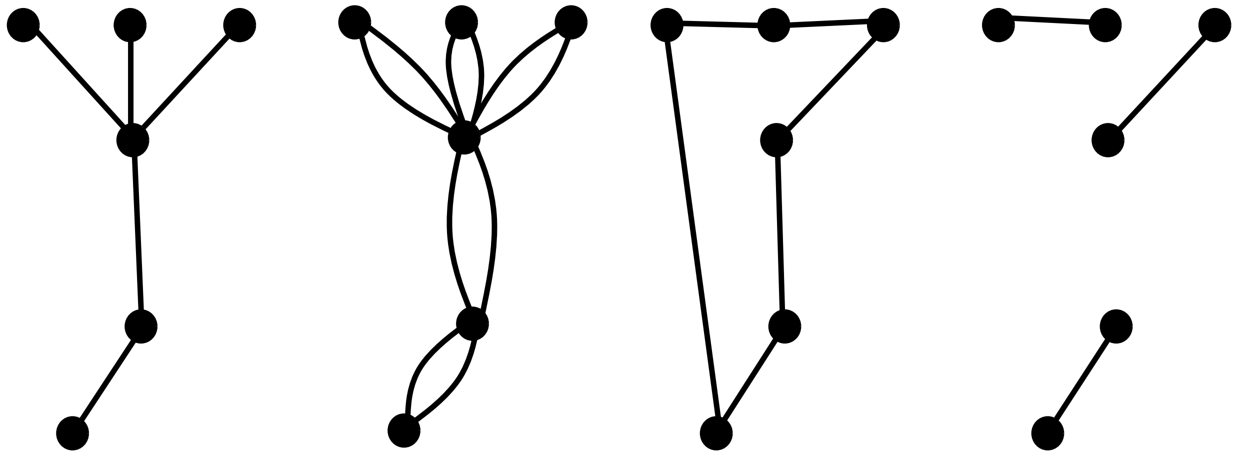
- The problem is known to be polynomial-time solvable (Edmonds '65).

- The best known running times are $O(n(m+n \log n))$ (Gabow '90) and $O(m\sqrt{n\alpha(m,n)\log n} \log nC)$ (Gabow, Tarjan '89).

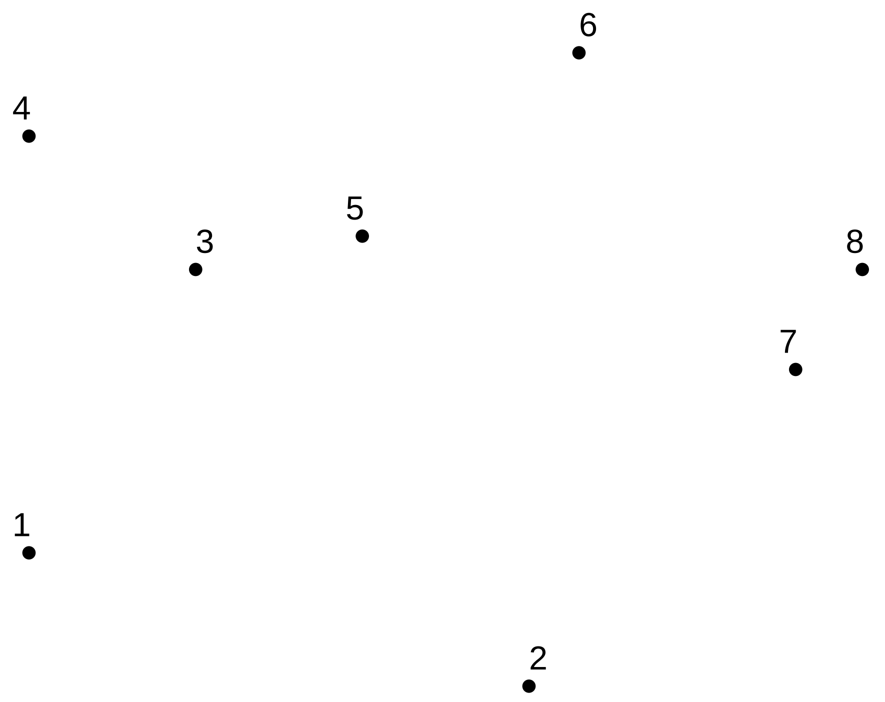
- The best previously known approximation algorithm has a performance guarantee of $3+2\epsilon$ and a running time of $O(n^2 \log^{2.5} n \log \frac{1}{\epsilon})$ (Vaidya '89).

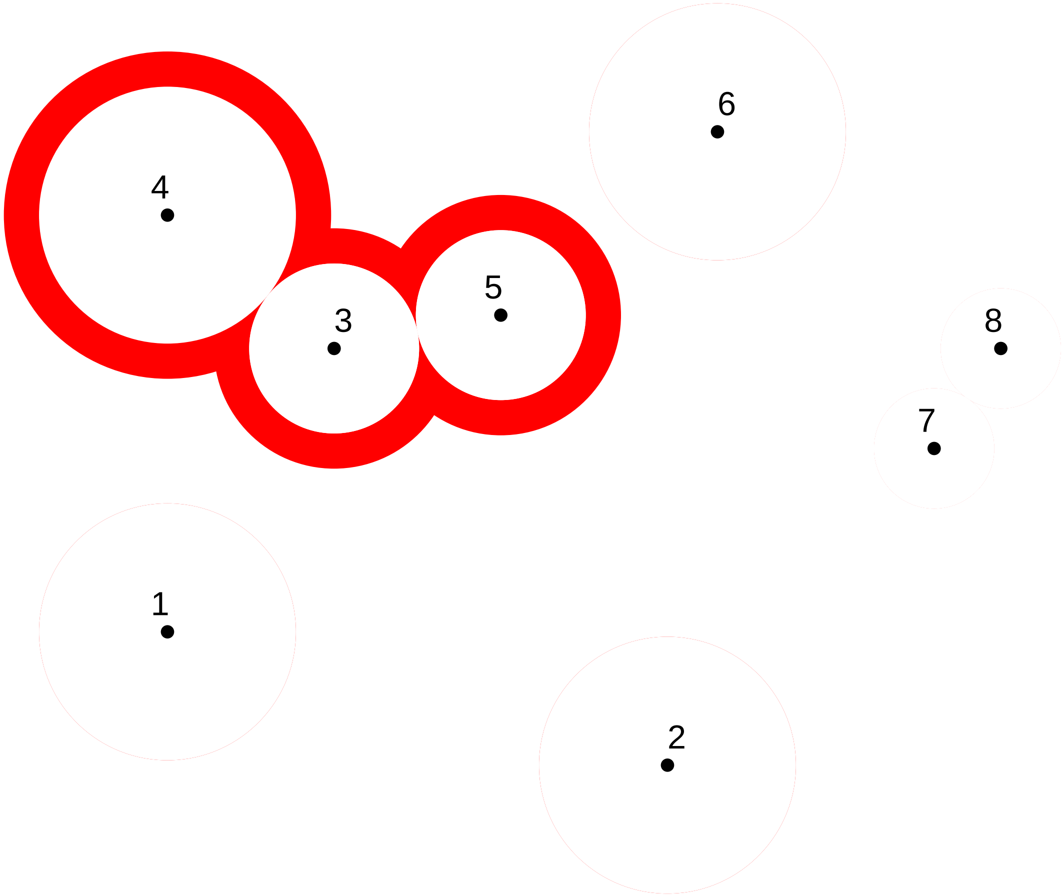
To get a 2-approximation algorithm for this problem, we consider hitting all sets $\delta(S)$ such that $|S|$ is odd. This gives a solution such that every connected component has even size.

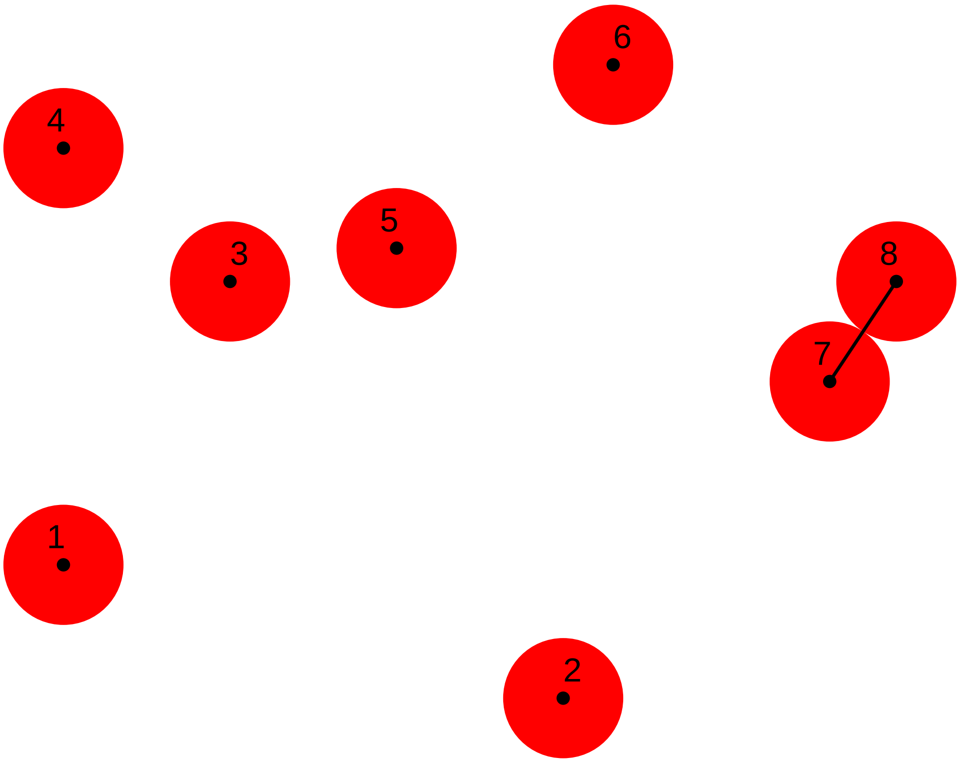
To get a matching, duplicate edges, shortcut it to a tour, and take the best of the two induced matchings.

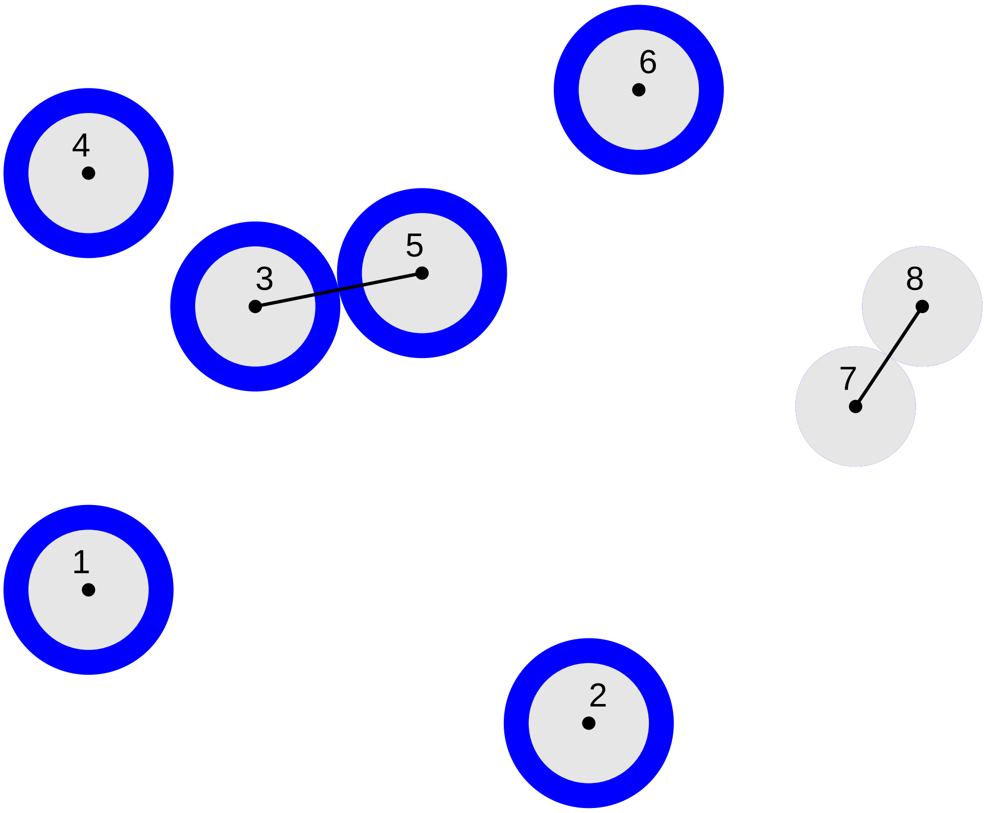


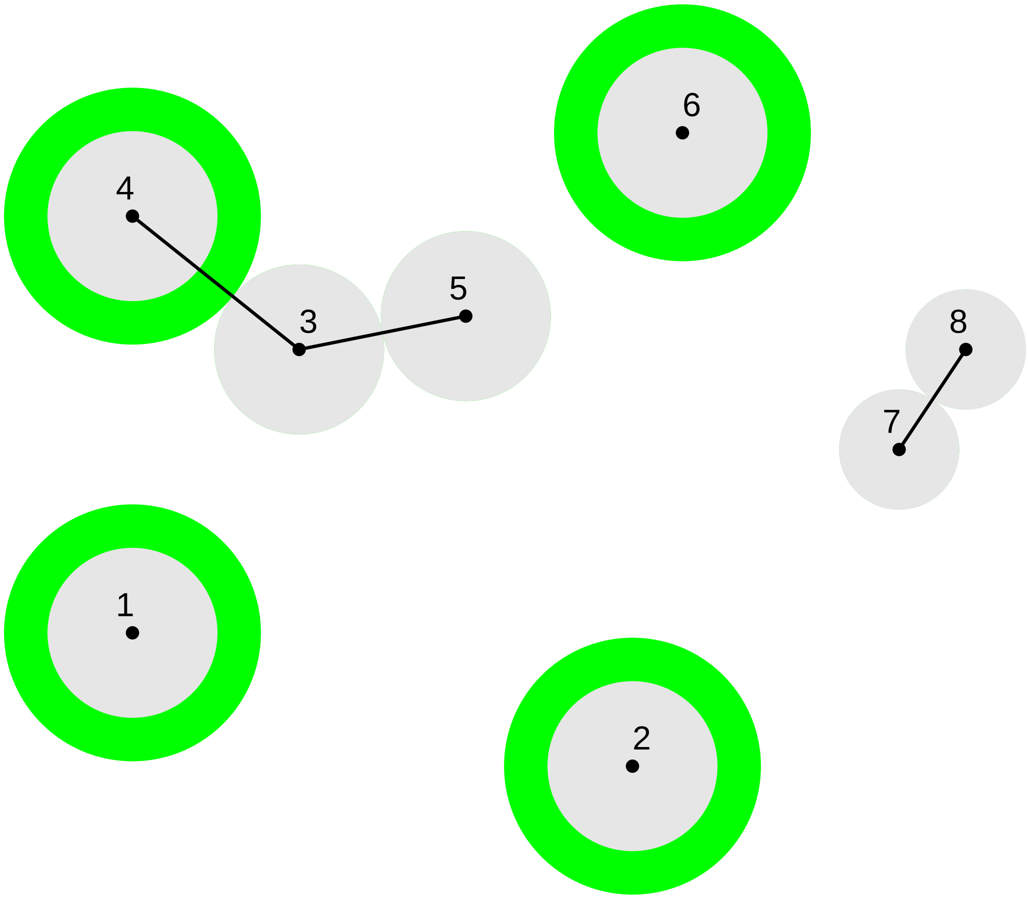
To apply the algorithm, we set $\mathrm{Violated}(A)$ to be the cuts corresponding to odd-sized components of (V, A).

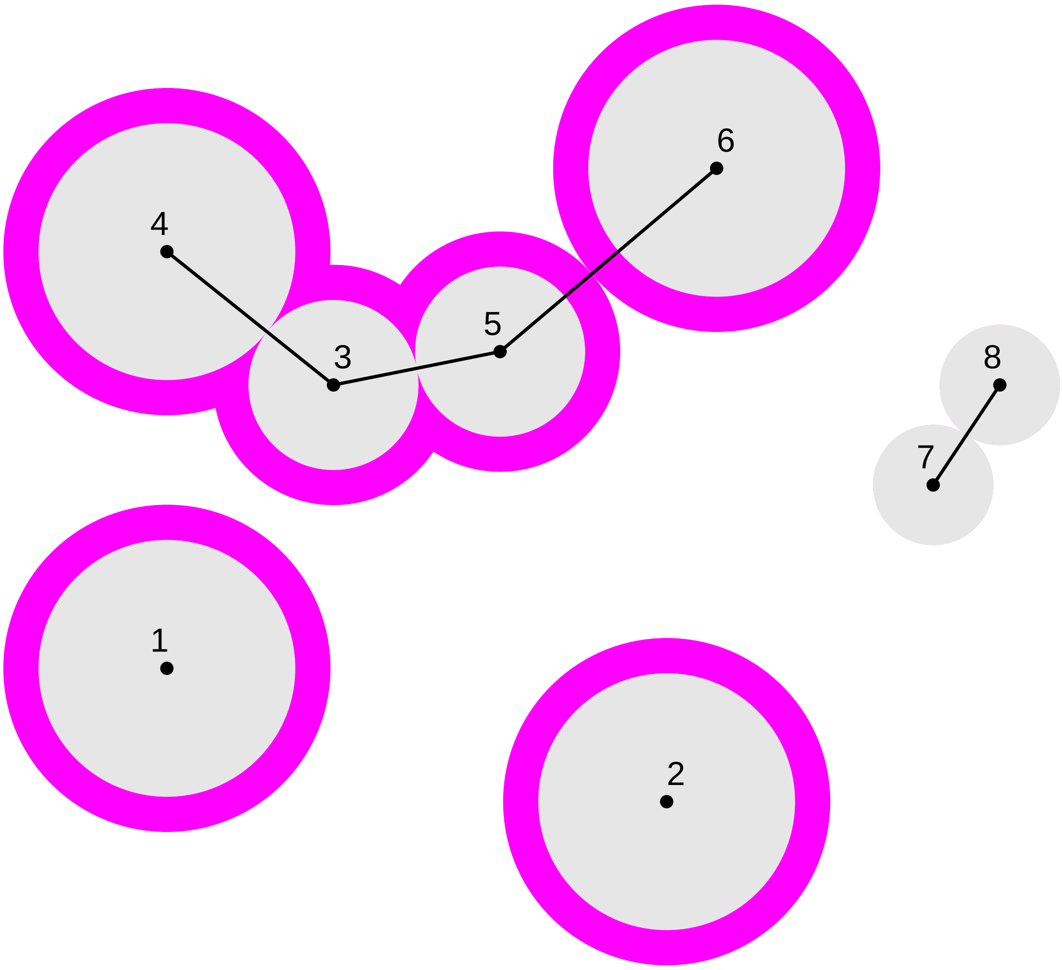

6
4
5
3
8
7
1
2

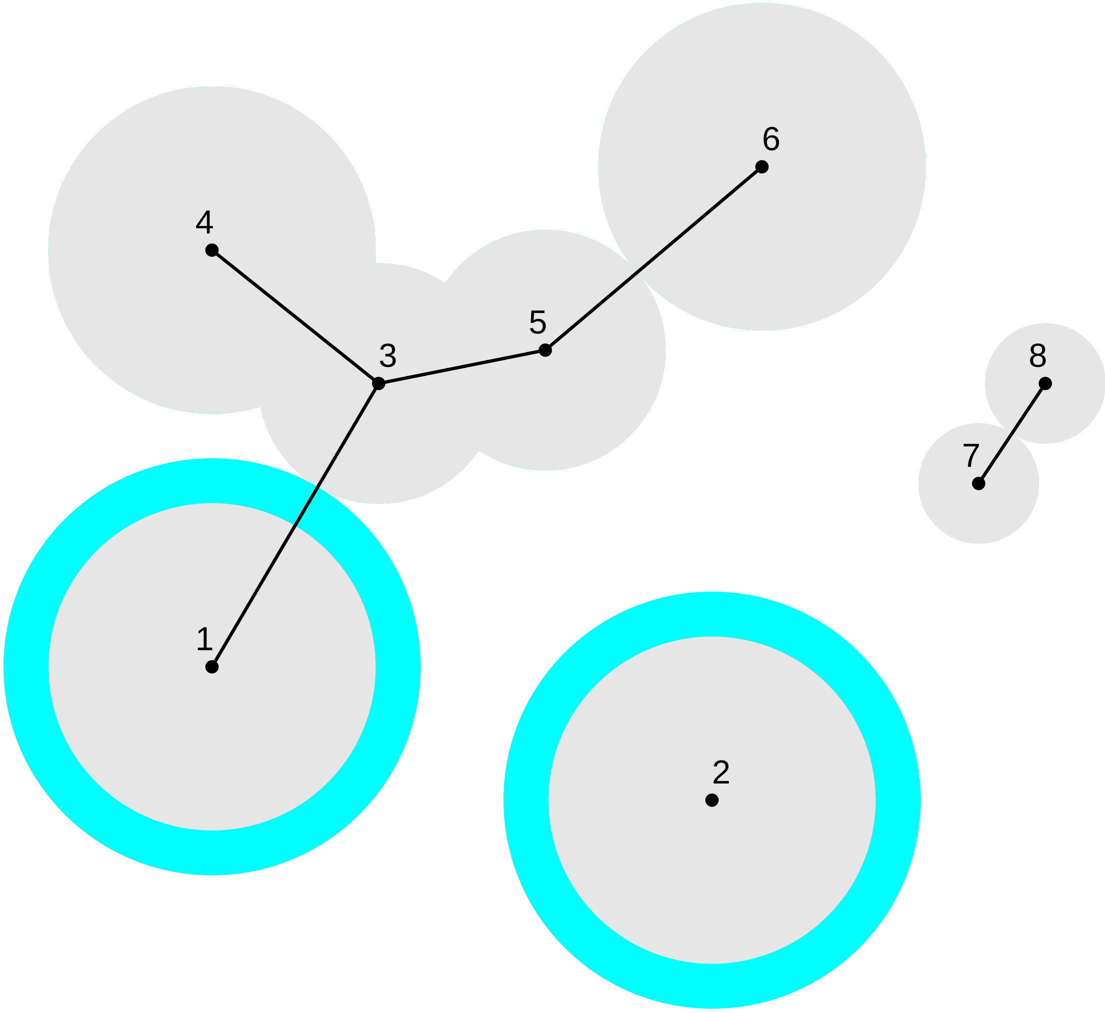

4
5
3
6
8
7
1
2

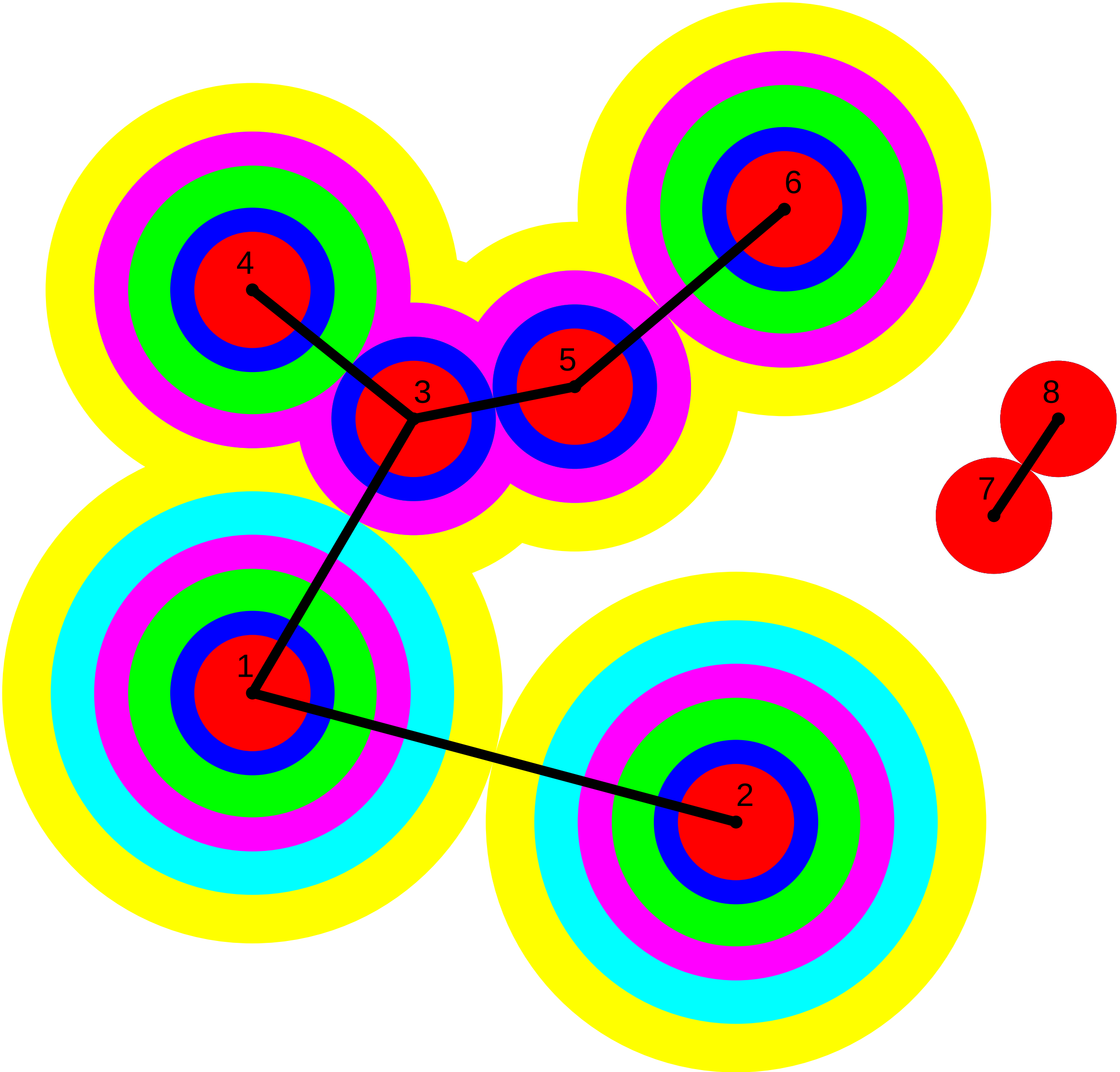

6
4
5
3
8
7
1
2

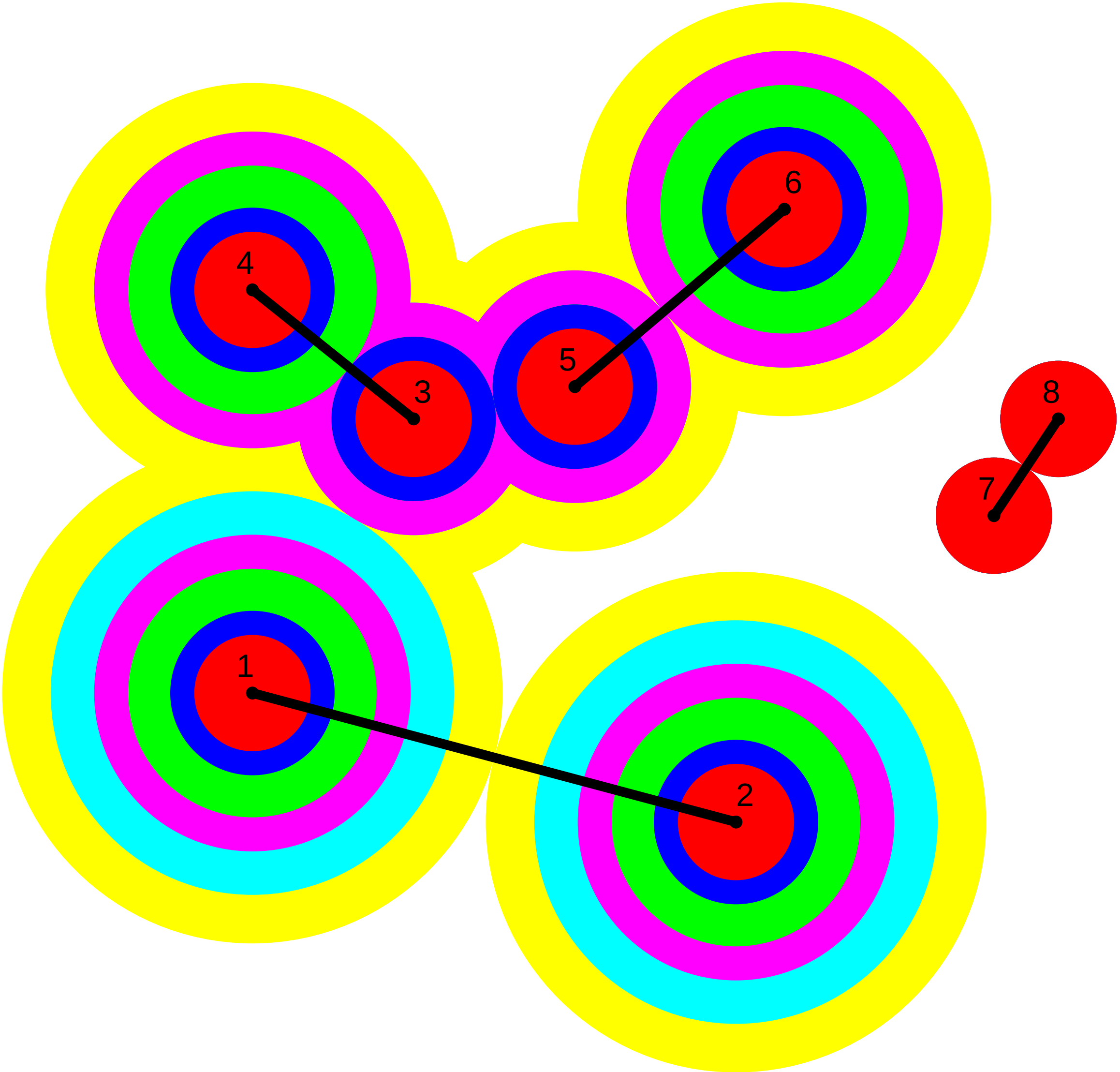

6
4
5
3
8
7
1
2


4
6
5
3
8
7
1
2


6
4
5
3
8
7
1
2


6
4
5
3
8
7
1
2

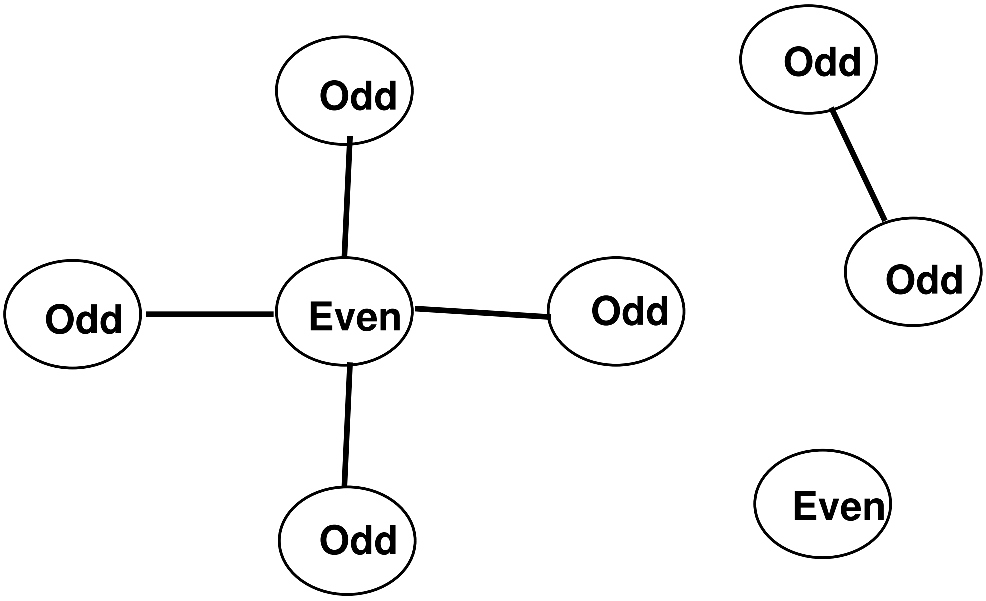

4
6
3
5
8
7
1
2


6
4
5
3
8
7
1
2

Proof

We want to show that for any infeasible A and any minimal augmentation B of A,

$$\sum_{\text{odd comps } C} |B \cap \delta(C)| \leq 2(\# \text{ odd comps}).$$



Lemma No "leaf" component is even.

Proof cont.

$$
\begin{aligned}
& \sum_{\text{odd comps } C} |B \cap \delta(C)| \\
= & \sum_{\text{all comps } C} |B \cap \delta(C)| - \sum_{\text{even comps } C} |B \cap \delta(C)| \\
\leq & \ 2(\#\text{ all comps} - 1) - 2(\#\text{ even comps}) \\
= & \ 2(\#\text{ odd comps}) - 2.
\end{aligned}
$$

Therefore, the algorithm is a 2-approximation algorithm.

The algorithm can be used as a building block for solving more sophisticated problems.

Example: Survivable Network Design Problem
Given a graph $G = (V, E)$, costs c_e, for all $e \in E$, and r_{ij} for all $i, j \in V$, find a minimum-cost set of edges such that there are at least r_{ij} edge-disjoint paths between i and j.

The problem is NP-hard; Steiner trees are a special case (Karp '72).

It is equivalent to a hitting set problem in which we must hit a cut $\delta(S)$

$$f(S) = \max_{i \in S, j \notin S} r_{ij}$$

times.

To get an algorithm for this problem, we use the original hitting set algorithm in phases, 1 to $f_{\max} = \max_S f(S)$.

Algorithm

$F \leftarrow \emptyset$

For $p \leftarrow 1$ to $f_{\max}$

$\Delta(S) = f(S) - |\delta(S) \cap F|$

$M \leftarrow \max_S \Delta(S)$

Apply the hitting set algorithm to $G = (V, E - F)$ with sets S such that $\Delta(S) = M$.

$F \leftarrow F \cup A$.

Theorem This is a $2\mathcal{H}(f_{\max})$-approximation algorithm for the survivable network design problem, which can be made to run in $O(f_{\max} n^2 (f_{\max} + \sqrt{\log\log n}))$ time.

$$\mathcal{H}(k) = 1 + \frac{1}{2} \cdots + \frac{1}{k} \approx \ln k$$

The same algorithm can also be used to approximate more general problems in $O(f_{\max}^2 n^3)$ time.

(W, Goemans, Mihail, Vazirani '93)
(Goemans, Goldberg, Plotkin, Shmoys, Tardos, W '94)
(Gabow, Goemans, W '93)

Exercises

1. Use the primal-dual algorithm to obtain a 2-approximation algorithm for multicuts in trees (Garg, Vazirani, Yannakakis '93). (*Hint:* Root the tree at an arbitrary vertex).

2. Consider the prize-collecting Steiner tree problem: given an undirected graph $G = (V, E)$, edge costs c_e, vertex costs π_i, and a root vertex r, find a tree containing r that minimizes the cost of the edges in the tree plus the cost of the vertices not in the tree.

   (a) Formulate this problem as a hitting set problem.
   (b) (Hard) Show that the primal-dual algorithm gives a 2-approximation algorithm for this problem (Goemans, W '92).

Research Problems

1. More applications of the primal-dual algorithm.

2. Find a constant approximation algorithm for the survivable network design problem. Easier problem: find such an algorithm if you are allowed to take multiple copies of edges.