



Great Ideas in Theoretical CS

Lecture 20:

Approximation Algorithms

Anil Ada

Ariel Procaccia (this time)

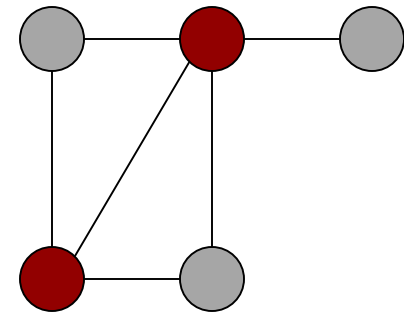
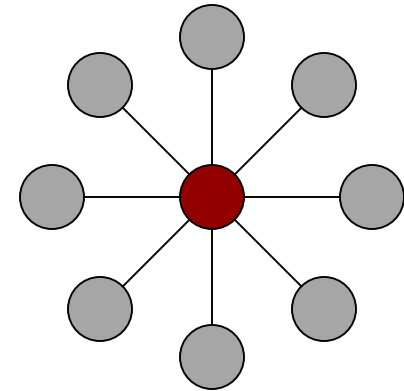
COMPUTATIONAL HARDNESS

- We saw that NP-hardness can be a force for good (preventing manipulation)
- But typically it just gets in the way of solving problems we want to solve!
- What can we do?
 - In practice: Heuristics often work well
 - In theory: Run in polynomial time and provide formal guarantees wrt the quality of the solution



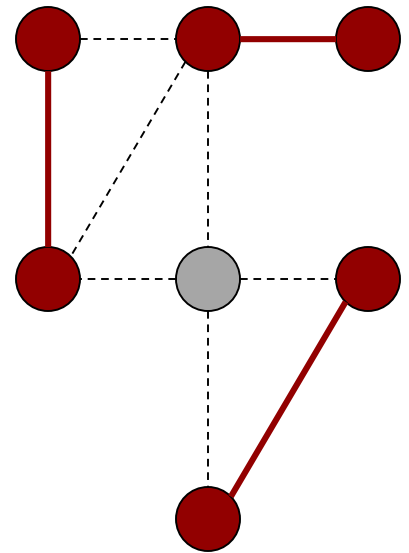
VERTEX COVER

- **VERTEX COVER:** Given a graph $G = (V, E)$ find the smallest $S \subseteq V$ such that every edge in E is incident on a vertex in S
- Decision version of the problem is NP-complete



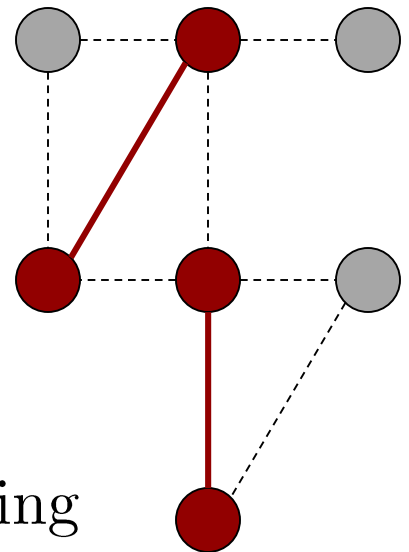
VERTEX COVER

- We don't know the size of the optimal vertex cover, but...
- **Lemma:** Let M be a matching in G , and S be a vertex cover. Then $|S| \geq |M|$
- **Proof:** S must cover at least one vertex for each edge in M ; this covers no other edges in M ■



VERTEX COVER

- Reminder: A matching M is **maximal** if \nexists matching $M' \neq M$ such that $M \subseteq M'$
- **Poll 1:** Which of the following algs would find a maximal matching:
 1. Greedily add edges that are disjoint from the edges added so far, while such edges exist
 2. Compute a **maximum cardinality** matching
 3. Both
 4. Neither



VERTEX COVER

APPROX-VC(G)

$M \leftarrow$ maximal matching on G

$S \leftarrow$ all vertices incident on M

Return S

- **Theorem:** Given a graph G , let $OPT(G)$ be the size of the optimal vertex cover and $S = APPROX-VC(G)$; S is a valid cover with $|S| \leq 2 \cdot OPT(G)$

We can say
this even
though we
don't know
OPT!



VERTEX COVER

- **Theorem:** Given a graph G , let $OPT(G)$ be the size of the optimal vertex cover and $S = APPROX-VC(G)$; S is a valid cover with $|S| \leq 2 \cdot OPT(G)$
- **Proof:**
 - For each $e \in E$, at least one vertex is in M , so S is a valid vertex cover
 - By the lemma, $|S| = 2|M| \leq 2 \cdot OPT$ ■

Can we
replace the 2
factor with
 $\alpha < 2$?



APPROXIMATION

- For a **minimization problem** instance I and algorithm ALG , let $ALG(I)$ be the quality of the algorithm's output and $OPT(I)$ be the quality of the optimal solution
- For $c > 1$, ALG is a **c -approximation alg** if for **every** I , $ALG(I) \leq c \cdot OPT(I)$
- APPROX-VC is a polytime 2-approximation algorithm for VERTEX-COVER

APPROXIMATION

- For a maximization problem and $c < 1$, ALG is a **c -approximation algorithm** if for every I , $ALG(I) \geq c \cdot OPT(I)$

These notions allow us to circumvent NP-hardness by designing polynomial-time algs with formal worst-case guarantees!



APPROXIMATION

- Algorithm STUPID-APPROX(G): Return all vertices of G (assume G is not empty)
- **Poll 2:** What is the smallest value of α for which STUPID-APPROX is an α -approx algorithm for VERTEX COVER?
 1. $\alpha = 3$
 2. $\alpha = \log n$
 3. $\alpha = \lceil n/2 \rceil$
 4. $\alpha = n$



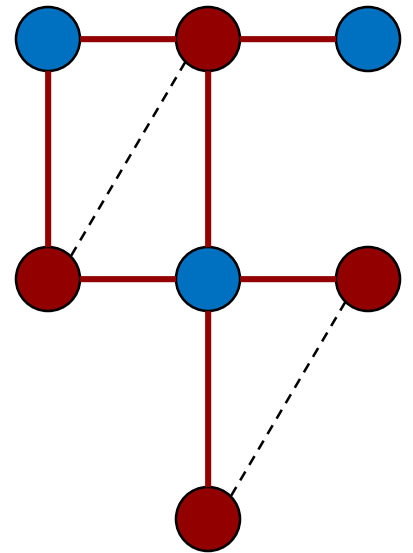
MAX CUT



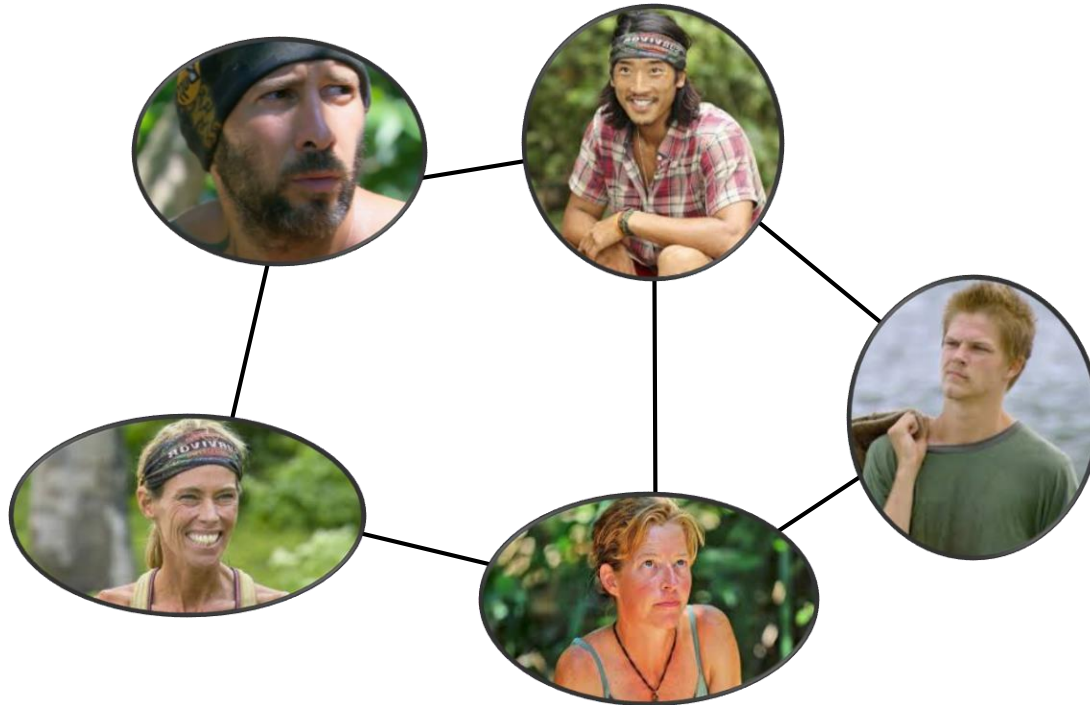
Ryan's favorite problem!

MAX CUT

- Given a coloring of vertices in red and blue, an edge is a **cut edge** if and only if its endpoints have different colors
- **MAX CUT**: Given a graph $G = (V, E)$, find a coloring of V in red and blue that maximizes the number of cut edges



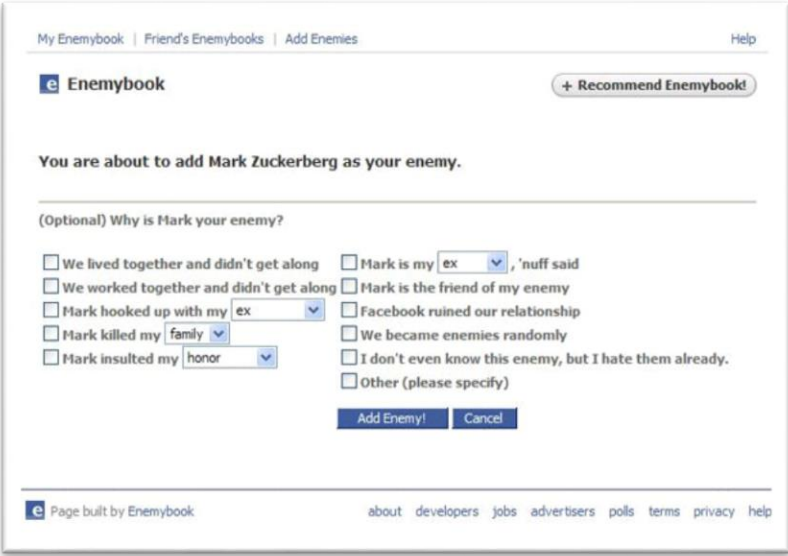
MAX CUT



Partition into two tribes to break as many friendships as possible (to maximize drama)

MAX CUT

More natural if the social network recorded “enemyships” instead of friendships



The screenshot shows the 'Enemybook' interface. At the top, there are navigation links: 'My Enemybook', 'Friend's Enemybooks', 'Add Enemies', and 'Help'. Below this is the 'Enemybook' logo and a '+ Recommend Enemybook' button. The main heading reads 'You are about to add Mark Zuckerberg as your enemy.' Below this is a section titled '(Optional) Why is Mark your enemy?' with several radio button options and dropdown menus. The options include: 'We lived together and didn't get along', 'We worked together and didn't get along', 'Mark hooked up with my ex', 'Mark killed my family', 'Mark insulted my honor', 'Mark is my ex, 'nuff said', 'Mark is the friend of my enemy', 'Facebook ruined our relationship', 'We became enemies randomly', and 'I don't even know this enemy, but I hate them already.' There is also an 'Other (please specify)' option. At the bottom of the form are 'Add Enemy!' and 'Cancel' buttons. The footer contains the text 'Page built by Enemybook' and a list of links: 'about', 'developers', 'jobs', 'advertisers', 'polls', 'terms', 'privacy', 'help'.



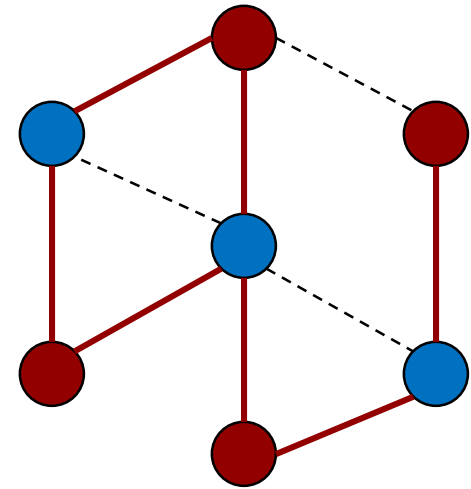
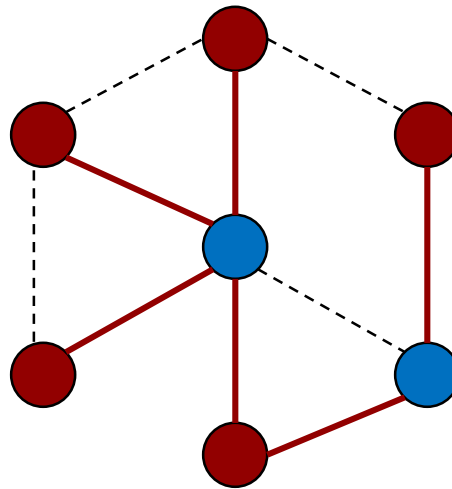
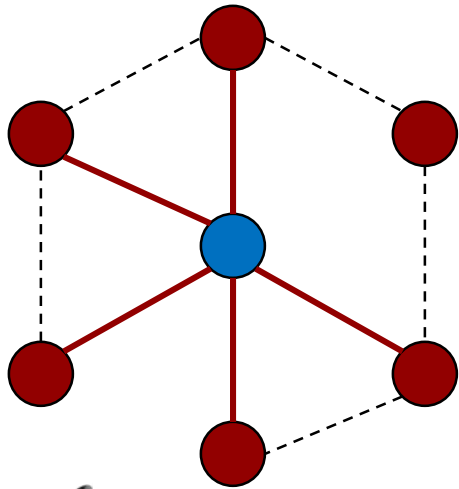
MAX CUT

APPROX-MC(G)

Start from arbitrary coloring

While \exists vertex v such that changing its color increases the number of cut edges

Change the color of v



MAX CUT

APPROX-MC(G)

Start from arbitrary coloring

Loop

If \exists vertex such that changing its color increases the number of cut edges, change its color

- **Poll 3:** What is the maximum number of iterations in the worst case?

1. $\Theta(m)$
2. $\Theta(mn)$
3. $\Theta(m^2)$
4. $\Theta(m^2n)$

Polynomial
time!

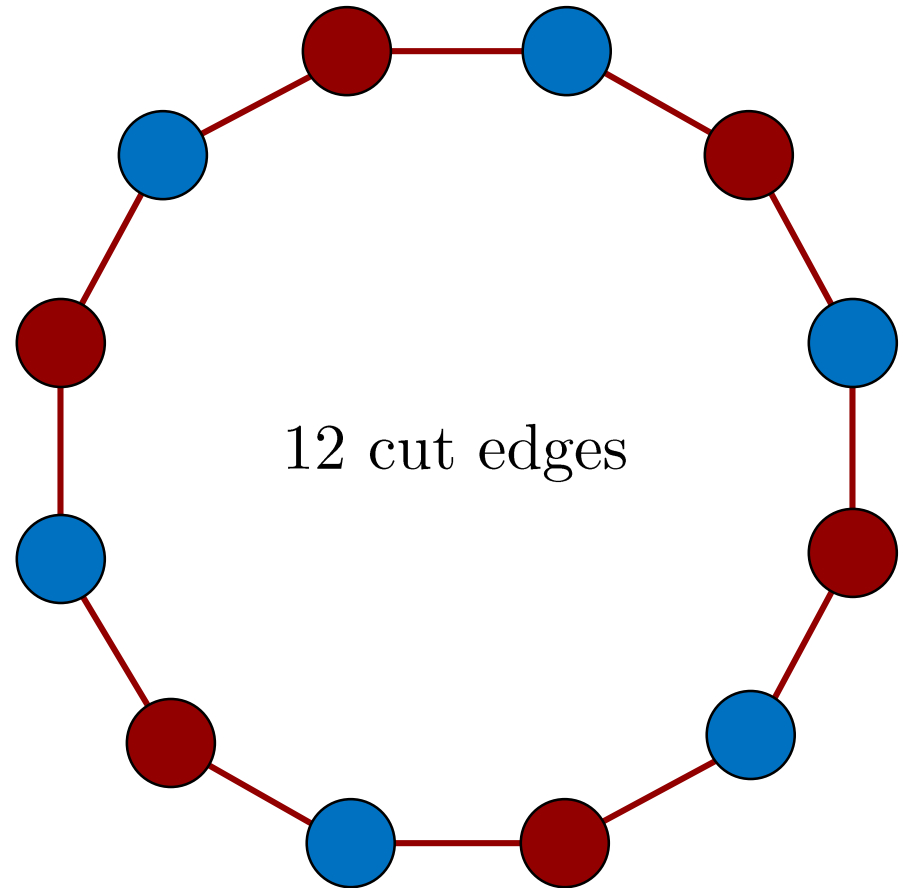
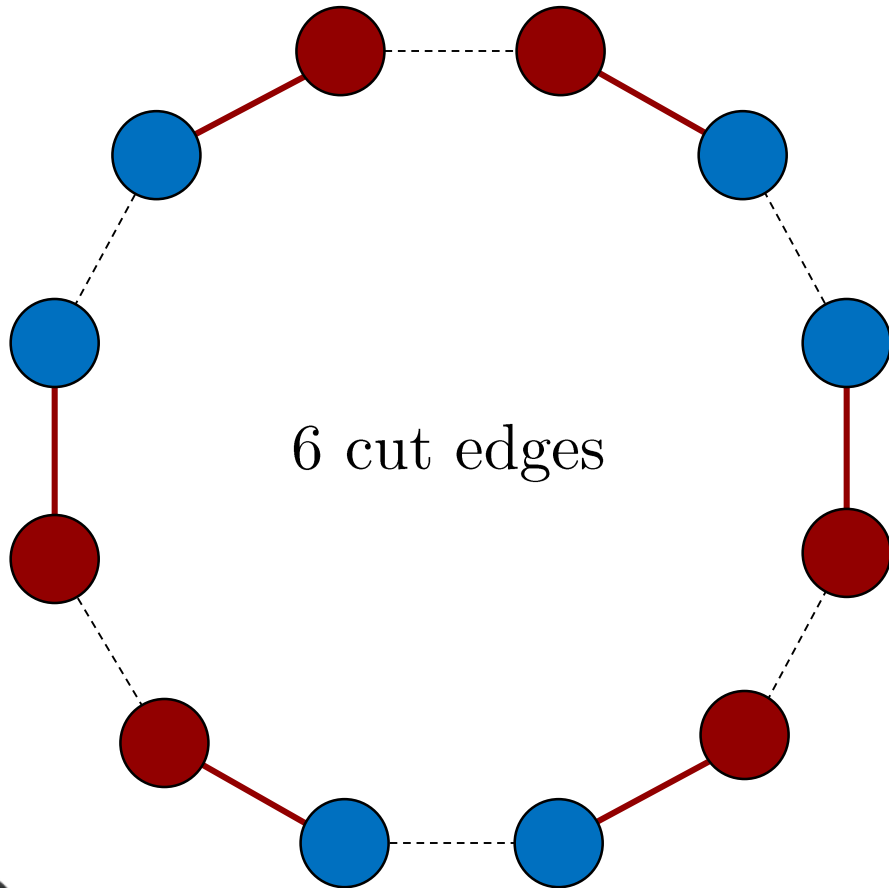


MAX CUT

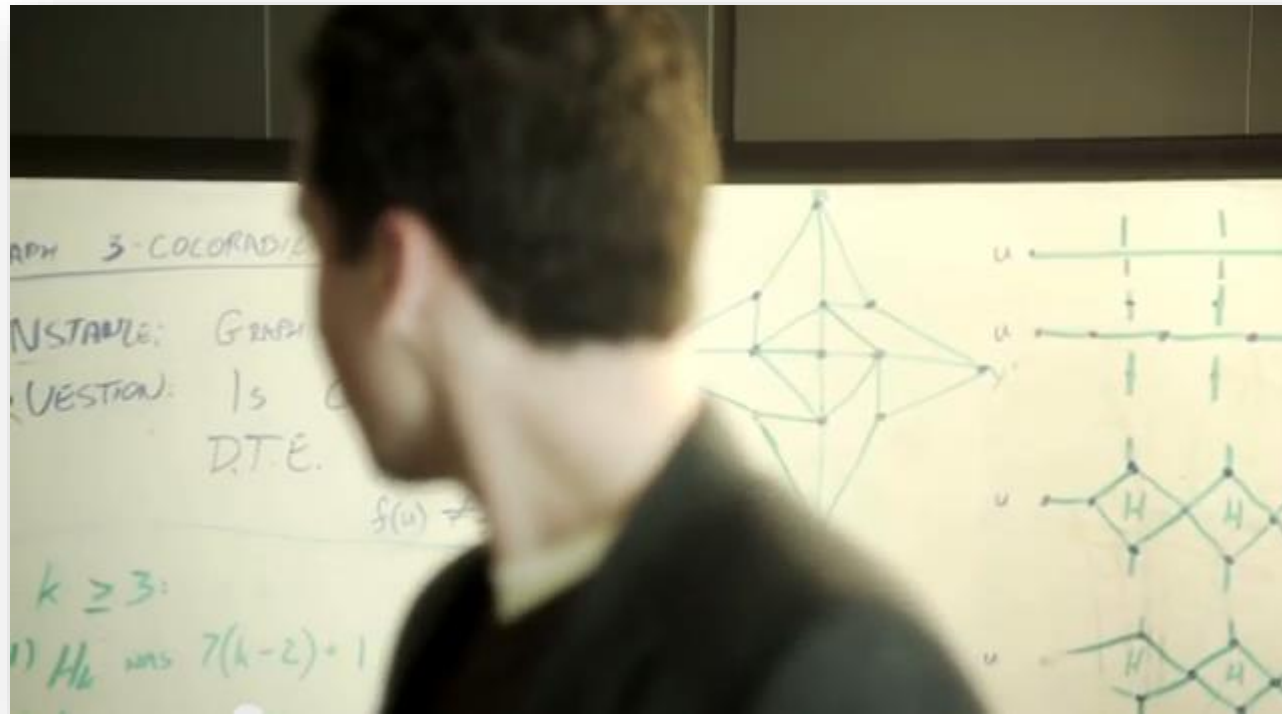
- **Theorem:** APPROX-MC is a $\frac{1}{2}$ -approximation algorithm for MAX CUT
- **Proof:**
 - When the algorithm returns, each $v \in V$ has at least $\deg(v)/2$ of its edges cut (**why?**)
 - Therefore, the solution is guaranteed to have at least $m/2$ cut edges (**exercise**)
 - $OPT \leq m$ ■



MAX CUT



INTERLUDE



<https://youtu.be/6ybd5rbQ5rU>

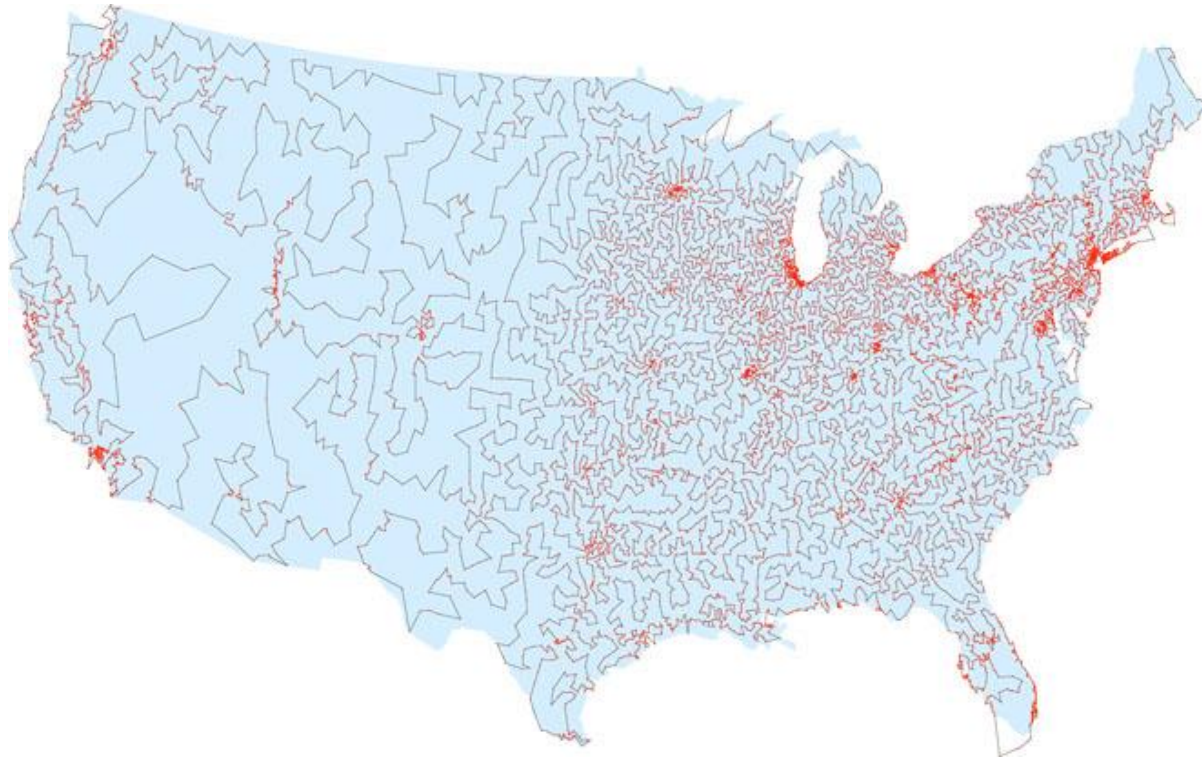


TRAVELING SALESMAN

- **TRAVELING SALESMAN (TSP)**: Given a graph $G = (V, E)$ with edge costs $c: E \rightarrow \mathbb{N}$, find a minimum cost tour that visits each vertex exactly once
- NP-complete by reduction from HAMILTONIAN CYCLE: Given an instance, assign $c(e) = 1$ for each $e \in E$ and ask whether there is a tour of cost n
- **Metric TSP**: can visit vertices multiple times (also NP-complete)



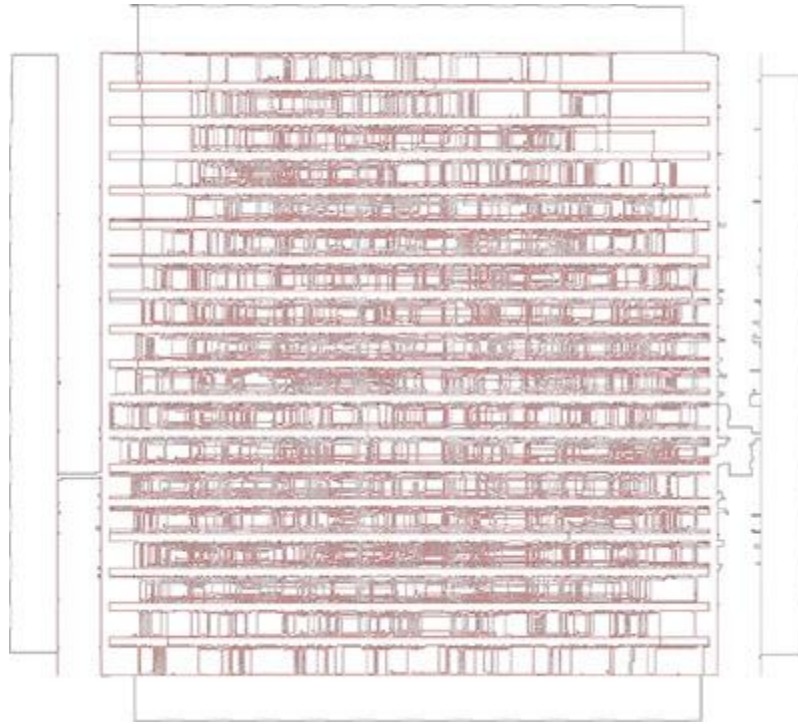
TRAVELING SALESMAN



Shortest traveling salesman route going through all 13,509 cities in the United States with a population of at least 500 (as of 1998)



TRAVELING SALESMAN



An 85,900-vertex route. The graph corresponds to the design of a customized computer chip created at Bell Laboratories, and the solution exhibits the shortest path for a laser to follow as it sculpts the chip.

TRAVELING SALESMAN

Finding long chains in kidney exchange using the traveling salesman problem

Ross Anderson¹, Itai Ashlagj¹, David Gamarnik², and Alvin E. Roth^{1*}

¹Operations Research Center and ²Slater School of Management, Massachusetts Institute of Technology, Cambridge, MA 02142; and ³Department of Economics, Stanford University, Stanford, CA 94305

Contributed by Alvin E. Roth, November 18, 2014; accepted for review July 15, 2014; reviewed by Peter Bräu

As of May 2014 there were more than 100,000 patients on the waiting list for a kidney transplant from a deceased donor. Although the preferred treatment is a kidney transplant, every year there are fewer donors than new patients, so the wait for a transplant continues to grow. To address this shortage, kidney paired donation (KPD) programs allow patients with living but biologically incompatible donors to exchange donors through cycles or chains initiated by altruistic nondirected donors, thereby increasing the supply of kidneys to recipients. In every KPD program, an optimization algorithm determines which exchanges will take place to maximize the total number of transplants performed. This optimization problem has proven challenging both in theory, because it is NP-hard, and in practice, because the algorithms previously used were unable to optimally search over all long chains. We give two new algorithms that use integer programming to optimally solve this problem, one of which is helped by the techniques used to solve the traveling salesman problem. These algorithms provide the tools needed to find optimal solutions in practice.

kidney exchange • kidney paired donation • transplantation • algorithms • computation

As of May 2014 there were over 100,000 patients on the waiting list for a kidney transplant from a deceased donor. Many of these patients have a friend or family member willing to be a living kidney donor, but the donor is biologically incompatible. Kidney exchange, also called kidney paired donation (KPD), arose to allow these patients with willing donors (referred to as patient-donor pairs) to exchange kidneys, thus increasing the number of living donor transplants and reducing the size of the waiting list.

In KPD, incompatible patient-donor pairs can exchange kidneys in cycles with other such pairs so that every patient receives a kidney from a compatible donor (Fig. S1). Additionally, there are a small number of altruistic donors who are willing to donate their kidney to any pair without asking for anything in return. In KPD, these donors initiate a chain of transplants with incompatible pairs, ending with a patient on the waiting list that has no associated donor (Fig. S2).

Integrating both cycles and chains in KPD was proposed in ref. 1, allowing both chains and cycles to be of arbitrary size. To ensure that every patient receives a kidney before her associated donor donates her kidney, cycles are executed simultaneously (otherwise, if the intended donor is unable to donate to the patient whose associated donor already gave her kidney the pair not only did not receive a kidney, but also could not participate in future exchanges). Because organizing many surgeries simultaneously is logistically very complex, the first implementations of KPD by the New England Program for Kidney Exchange and other clearinghouses used only two-way cyclic exchanges. After a short period, clearinghouses have moved to allow three-way exchanges as well.

In ref. 2 it was proposed to relax the requirement of simultaneously to the weaker requirement that every patient-donor pair receive a kidney before they give a kidney. Although for cycles this restriction is still required all surgeries be performed

simultaneously, it did allow for nonsimultaneous chains. Note that these nonsimultaneous chains still protected patient-donor pairs from irrevocable harm but allowed for the possibility of donors' backing out after their patient had received a transplant. Since the first nonsimultaneous chain was arranged (3) chain-type exchanges have accounted for a majority of the transplants in kidney exchange clearinghouses. [Approximately 75% of the transplants in the National Kidney Registry (NKR) and the Alliance for Paired Donations (APD) are done through chains.] Chains involving as many as 30 pairs have been performed in practice, capturing significant public interest. (4). Very long chains are often planned in segments, in which the donor from the final pair in a segment is used to begin another segment after some patients arrive. Such donors are called "bridge donors." Once the segment is executed, the bridge donors and altruistic donors are essentially identical for the purpose of planning future transplants and are collectively referred to here as "non-directed donors" (NDDs). The problem of determining how to optimally select a set of cycles and chains to maximize the number of transplants performed is the focus of this work.

We refer to the problem of finding the maximum (possibly weighted) number of transplants for a pool of incompatible patient-donor pairs and NDDs as the kidney exchange problem (KEP). Optimally included in the problem are a maximum chain length and a maximum cycle length. Weights can be used to prioritize difficult-to-match patients. An example of a KEP instance is shown in Fig. S3. When there is no bound on the chain or cycle length, the problem can be solved efficiently by reducing it to a maximum weighted perfect matching problem on a bipartite graph, as described in ref. 5. The special case in which only cycles

Significance

There are currently more than 100,000 patients on the waiting list in the United States for a kidney transplant from a deceased donor. To address this shortage, kidney exchange programs allow patients with living incompatible donors to exchange donors through cycles and chains initiated by altruistic nondirected donors. To determine which exchanges will take place, kidney exchange programs use algorithms for maximizing the number of transplants under constraints about the size of feasible exchanges. This problem is NP-hard, and algorithms previously used were unable to optimize when chains could be long. We developed two algorithms that use integer programming to solve this problem, one of which is inspired by the traveling salesman, that together can find optimal solutions in practice.

Author contributions: R.A., I.A., D.G., and A.E.R. designed research; R.A. and I.A. performed research; A.E.R. analyzed data; R.A., I.A., D.G., and A.E.R. wrote the paper.

Reviewers included A. Institute of Economics, Hungarian Academy of Sciences. The authors declare no conflict of interest.

*To whom correspondence should be addressed. Email: alvin@mit.edu.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1424281112.

www.pnas.org/cgi/doi/10.1073/pnas.1424281112

PNAS | January 20, 2015 | vol. 112 | no. 3 | 460–468

Anderson et al., PNAS 2015

15251 Fall 2017: Lecture 20

Carnegie Mellon University 23

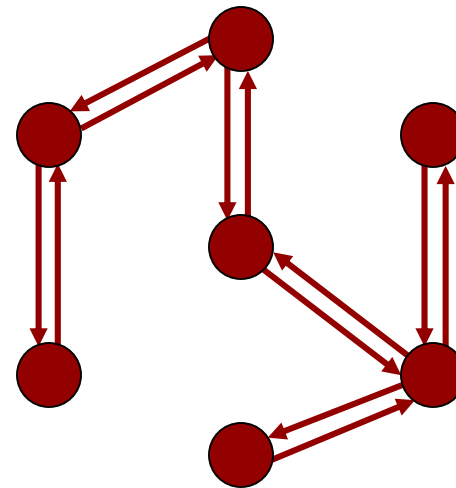
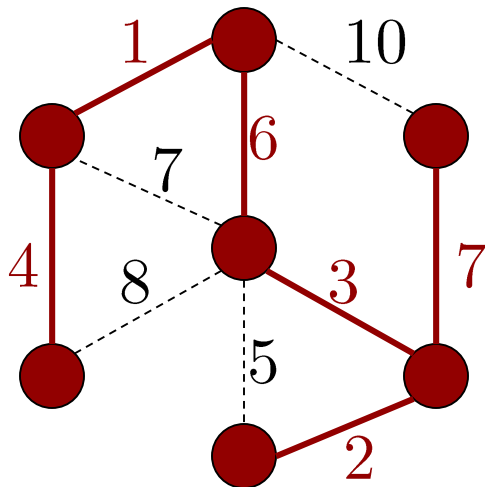
TRAVELING SALESMAN

APPROX-TSP(G)

$T \leftarrow$ Minimum spanning tree of G

$2T \leftarrow$ double edges of T

Return Eulerian tour of $2T$



TRAVELING SALESMAN

- **Theorem:** APPROX-TSP is a 2-approximation algorithm for Metric TSP
- **Proof:**
 - A TSP tour can be converted into a lower cost spanning tree (**how?**), therefore

$$c(T) = \sum_{e \in E(T)} c(e) \leq OPT$$

- Clearly $c(2T) = 2c(T)$
- It follows that $c(2T) \leq 2OPT$ ■



TRAVELING SALESMAN*

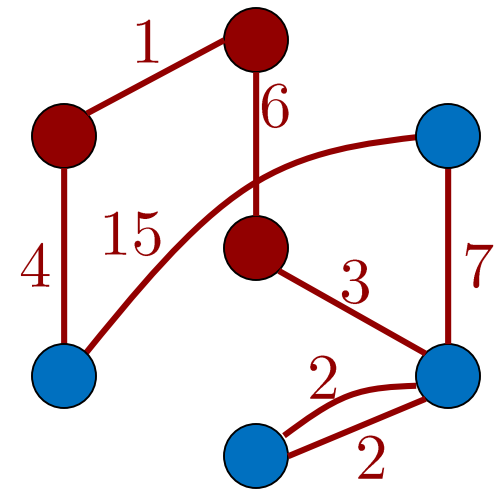
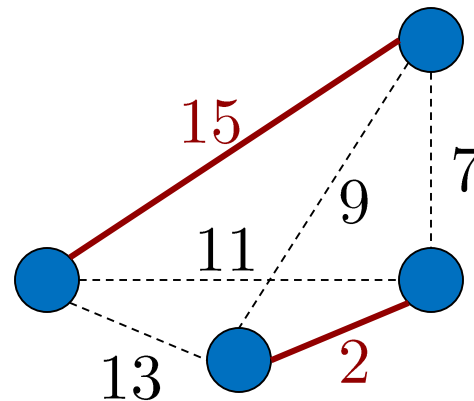
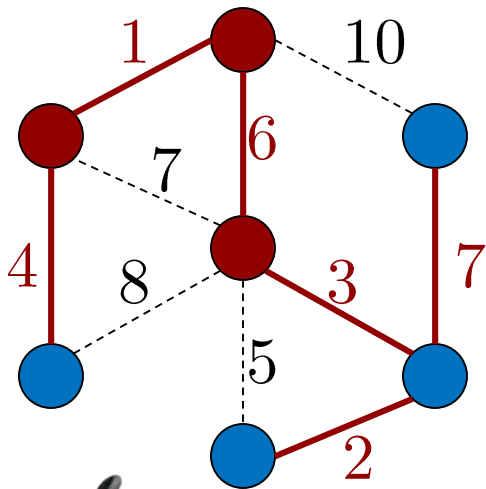
CHRISTOFIDES(G)

$T \leftarrow$ Minimum spanning tree of G

$S \leftarrow$ Vertices of odd degree in T ($|S|$ is even, **why?**)

$M \leftarrow$ Min cost **perfect** matching on S in G

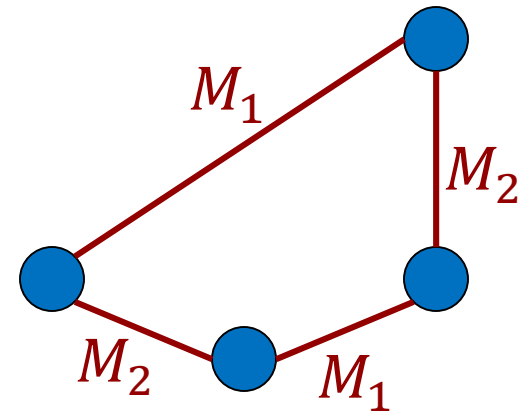
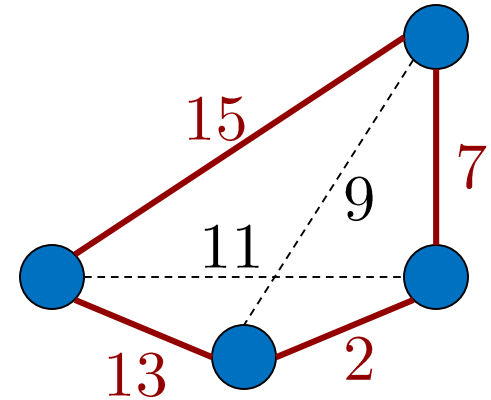
Return Eulerian tour of $T \cup M$ (it exists, **why?**)



* Just for fun

TRAVELING SALESMAN*

- Lemma: $C(M) \leq \frac{1}{2} OPT$
- Proof:
 - \exists tour of S of cost at most OPT (because $S \subseteq V$)
 - Decompose into two matchings M_1 and M_2
 - $c(M_1) + c(M_2) \leq OPT$, but $c(M) \leq c(M_1)$ and $c(M) \leq c(M_2) \Rightarrow c(M) \leq \frac{1}{2} OPT$ ■



TRAVELING SALESMAN*

- **Theorem:** CHRISTOFIDES is a $\frac{3}{2}$ -approximation algorithm for Metric TSP
- **Proof:** Using the lemma,

$$\begin{aligned}ALG &= c(M) + c(T) \\ &\leq \frac{1}{2} OPT + OPT \\ &= \frac{3}{2} OPT \quad \blacksquare\end{aligned}$$

SUMMARY

- Definitions
 - Approximation algorithm
 - VERTEX COVER, MAX CUT, TRAVELING SALESMAN
- Algorithms
 - 2-approximation for VERTEX COVER
 - $\frac{1}{2}$ -approximation for MAX CUT
 - 2-approximation for Metric TSP

