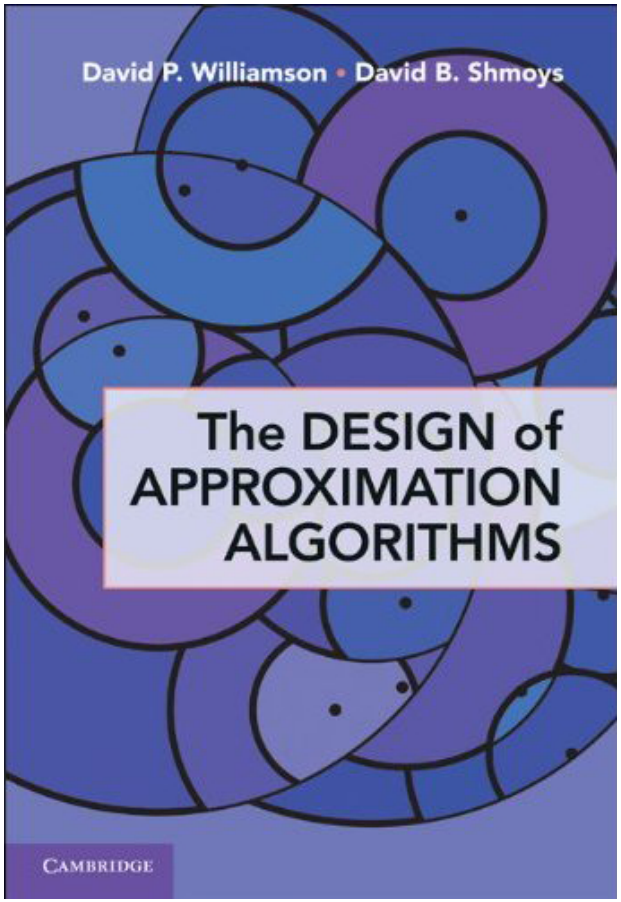
Some Open Problems in Approximation Algorithms

David P. Williamson

School of Operations Research and Information Engineering
Cornell University

February 28, 2011
University of Bonn
Bonn, Germany

The book



Electronic version at `www.designofapproxalgs.com`.


Cornell University

Outline

- Introduction
- A brief history and some early results
- Ten open problems
- Some concluding thoughts and issues

Cornell University

The Problem

The problem: How should we go about solving NP-hard discrete optimization problems?

The Problem

The problem: How should we go about solving NP-hard discrete optimization problems?

An old engineering slogan: “Fast. Cheap. Reliable. Choose any two.”

Similarly, if P $\neq$ NP, then for any NP-hard problem, choose two:

The Problem

The problem: How should we go about solving NP-hard discrete optimization problems?

An old engineering slogan: "Fast. Cheap. Reliable. Choose any two."

Similarly, if P $\neq$ NP, then for any NP-hard problem, choose two:

- a polynomial-time algorithm

The Problem

The problem: How should we go about solving NP-hard discrete optimization problems?

An old engineering slogan: "Fast. Cheap. Reliable. Choose any two."

Similarly, if P $\neq$ NP, then for any NP-hard problem, choose two:

- a polynomial-time algorithm
- that for every instance

The Problem

The problem: How should we go about solving NP-hard discrete optimization problems?

An old engineering slogan: "Fast. Cheap. Reliable. Choose any two."

Similarly, if P $\neq$ NP, then for any NP-hard problem, choose two:

- a polynomial-time algorithm
- that for every instance
- finds the optimal solution.

All work on these problems relaxes at least one of these conditions.

Some Approaches

1. Drop the polynomial-time requirement: integer programming, A^* search, constraint programming, . . .

Some Approaches

1. Drop the polynomial-time requirement: integer programming, A^* search, constraint programming, . . .
   - Pro: Often fast enough

Some Approaches

1. Drop the polynomial-time requirement: integer programming, A^* search, constraint programming, . . .
   - Pro: Often fast enough
   - Con: Sometimes not fast enough if we need solutions in seconds, or instance is very large; may not terminate in time

Some Approaches

1. Drop the polynomial-time requirement: integer programming, A^* search, constraint programming, ...
   - Pro: Often fast enough
   - Con: Sometimes not fast enough if we need solutions in seconds, or instance is very large; may not terminate in time
2. Drop the every instance requirement: special cases (e.g. planar graphs)

Some Approaches

1. Drop the polynomial-time requirement: integer programming, A^* search, constraint programming, ...
   - Pro: Often fast enough
   - Con: Sometimes not fast enough if we need solutions in seconds, or instance is very large; may not terminate in time
2. Drop the every instance requirement: special cases (e.g. planar graphs)
   - Pro: Sometimes the instances we have fall in these cases

Some Approaches

1. Drop the polynomial-time requirement: integer programming, A^* search, constraint programming, ...
   - Pro: Often fast enough
   - Con: Sometimes not fast enough if we need solutions in seconds, or instance is very large; may not terminate in time
2. Drop the every instance requirement: special cases (e.g. planar graphs)
   - Pro: Sometimes the instances we have fall in these cases
   - Con: Often they don't

Some Approaches

1. Drop the polynomial-time requirement: integer programming, A^* search, constraint programming, ...
   - Pro: Often fast enough
   - Con: Sometimes not fast enough if we need solutions in seconds, or instance is very large; may not terminate in time
2. Drop the every instance requirement: special cases (e.g. planar graphs)
   - Pro: Sometimes the instances we have fall in these cases
   - Con: Often they don't
3. Drop the optimality requirement: heuristics, metaheuristics, ...

Some Approaches

1. Drop the polynomial-time requirement: integer programming, A^* search, constraint programming, . . .
   - Pro: Often fast enough
   - Con: Sometimes not fast enough if we need solutions in seconds, or instance is very large; may not terminate in time
2. Drop the every instance requirement: special cases (e.g. planar graphs)
   - Pro: Sometimes the instances we have fall in these cases
   - Con: Often they don't
3. Drop the optimality requirement: heuristics, metaheuristics, . . .
   - Pro: Often we only need a solution that is "good enough"

Some Approaches

1. Drop the polynomial-time requirement: integer programming, A^* search, constraint programming, . . .
   - Pro: Often fast enough
   - Con: Sometimes not fast enough if we need solutions in seconds, or instance is very large; may not terminate in time
2. Drop the every instance requirement: special cases (e.g. planar graphs)
   - Pro: Sometimes the instances we have fall in these cases
   - Con: Often they don't
3. Drop the optimality requirement: heuristics, metaheuristics, . . .
   - Pro: Often we only need a solution that is "good enough"
   - Con: How good is the solution we get?

A Definition

Defined by David S. Johnson in 1974 paper.

Definition

An α-approximation algorithm for a discrete optimization problem Π, for any instance of Π, runs in polynomial time and produces a solution of cost within α times the cost of an optimal solution to the instance.

A Definition

Defined by David S. Johnson in 1974 paper.

Definition

An α-approximation algorithm for a discrete optimization problem Π, for any instance of Π, runs in polynomial time and produces a solution of cost within α times the cost of an optimal solution to the instance.

- Randomized approximation algorithms: expected value is within α of optimal.

A Definition

Defined by David S. Johnson in 1974 paper.

Definition

An α-approximation algorithm for a discrete optimization problem Π, for any instance of Π, runs in polynomial time and produces a solution of cost within α times the cost of an optimal solution to the instance.

- Randomized approximation algorithms: expected value is within α of optimal.
- Additive approximation algorithms: solution is within additive error of optimal.

A Definition

Defined by David S. Johnson in 1974 paper.

Definition

An α-approximation algorithm for a discrete optimization problem Π, for any instance of Π, runs in polynomial time and produces a solution of cost within α times the cost of an optimal solution to the instance.

- Randomized approximation algorithms: expected value is within α of optimal.
- Additive approximation algorithms: solution is within additive error of optimal.
- Polynomial-time approximation scheme (PTAS): for any $\epsilon > 0$, a $(1 + \epsilon)$-approximation algorithm.

Prehistory

But existed prior to Johnson's paper (even prior to P and NP!)

Prehistory

But existed prior to Johnson's paper (even prior to P and NP!)

- Erdős (1967): a random cut has expected value at least half of all edges in graph.

Prehistory

But existed prior to Johnson's paper (even prior to P and NP!)

- Erdős (1967): a random cut has expected value at least half of all edges in graph.
- Graham (1966, 1967): 2-approximation algorithm for a scheduling problem, and a PTAS in case the number of machines is fixed.

Prehistory

But existed prior to Johnson's paper (even prior to P and NP!)

- Erdős (1967): a random cut has expected value at least half of all edges in graph.
- Graham (1966, 1967): 2-approximation algorithm for a scheduling problem, and a PTAS in case the number of machines is fixed.
- Vizing (1964): edge coloring with additive error of 1.

Prehistory

But existed prior to Johnson's paper (even prior to P and NP!)

- Erdős (1967): a random cut has expected value at least half of all edges in graph.
- Graham (1966, 1967): 2-approximation algorithm for a scheduling problem, and a PTAS in case the number of machines is fixed.
- Vizing (1964): edge coloring with additive error of 1.

Johnson's 1974 paper gives:

- $(\ln n)$-approximation algorithm for (unweighted) set cover problem (see also Lovász (1975))
- $1/2$-approximation algorithm for maximum satisfiability problem
- approximation algorithms for vertex coloring and maximum clique problem

Johnson's paper

Paper ends with:

> *The results described in this paper indicate a possible classification of optimization problems as to the behavior of their approximation algorithms. Such a classification must remain tentative, at least until the existence of polynomial-time algorithms for finding optimal solutions has been proved or disproved. In the meantime, many questions can be asked. Are there indeed* $O(\log n)$ *coloring algorithms? Are there any clique finding algorithms better than* $O(n^{\epsilon})$ *for all* $\epsilon > 0$*? Where do other optimization problems fit into the scheme of things? What is it that makes algorithms for different problems behave in the same way? Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results, or are they due to some structural similarity between the problems as we define them? And what other types of behavior and ways of analyzing and measuring it are possible?*

Johnson's paper

Paper ends with:

> *The results described in this paper indicate a possible classification of optimization problems as to the behavior of their approximation algorithms. Such a classification must remain tentative, at least until the existence of polynomial-time algorithms for finding optimal solutions has been proved or disproved. In the meantime, many questions can be asked. Are there indeed* $O(\log n)$ *coloring algorithms? Are there any clique finding algorithms better than* $O(n^{\epsilon})$ *for all* $\epsilon > 0$*? Where do other optimization problems fit into the scheme of things? What is it that makes algorithms for different problems behave in the same way? Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results, or are they due to some structural similarity between the problems as we define them? And what other types of behavior and ways of analyzing and measuring it are possible?*

The community spent the next few decades trying to answer some of these questions.

Some Early, Easy Results

Maximum Satisfiability Problem

Input:

- n boolean variables $x_1, x_2, \ldots, x_n$
- m clauses of disjunctions of variables or negations, e.g. $x_1 \vee \bar{x}_5 \vee x_{12}$.
- clause weights $w_j \geq 0$ for $1 \leq j \leq m$.

Goal: Find a setting of the x_i that maximizes the weight of satisfied clauses.

Some Early, Easy Results

Maximum Satisfiability Problem

Input:

- n boolean variables $x_1, x_2, \ldots, x_n$
- m clauses of disjunctions of variables or negations, e.g. $x_1 \vee \bar{x}_5 \vee x_{12}$.
- clause weights $w_j \geq 0$ for $1 \leq j \leq m$.

Goal: Find a setting of the x_i that maximizes the weight of satisfied clauses.

An easy algorithm: Set each x_i true with probability 1/2.

Some Early, Easy Results

Maximum Satisfiability Problem

$$\begin{aligned}\text{Expected weight of satisfied clauses} &= \sum_{j=1}^{m} w_j \cdot \Pr[\text{Clause } j \text{ satisfied}] \\ &\geq \frac{1}{2} \sum_{j=1}^{m} w_j \\ &\geq \frac{1}{2} \text{OPT}\,.\end{aligned}$$

Cornell University

Some Early, Easy Results

Maximum Satisfiability Problem

$$\begin{aligned}\text{Expected weight of satisfied clauses} &= \sum_{j=1}^{m} w_j \cdot \Pr[\text{Clause } j \text{ satisfied}] \\ &\geq \frac{1}{2} \sum_{j=1}^{m} w_j \\ &\geq \frac{1}{2} \text{OPT}\,.\end{aligned}$$

If each clause has exactly three literals (MAX E3SAT), can show a $\frac{7}{8}$-approximation algorithm (Johnson 1974).

Some Easy, Early Results

Vertex Cover Problem

Input:

- Undirected graph $G = (V, E)$
- weights $w_v \geq 0$ for all $v \in V$

Goal: Find $S \subseteq V$ of minimum weight so that each edge has at least one endpoint in S.

Some Easy, Early Results

Vertex Cover Problem

Input:

- Undirected graph $G = (V, E)$
- weights $w_v \geq 0$ for all $v \in V$

Goal: Find $S \subseteq V$ of minimum weight so that each edge has at least one endpoint in S.

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} w_v x_v \\ \text{subject to} \quad & x_u + x_v \geq 1, \qquad \forall (u, v) \in E, \\ & x_v \geq 0, \qquad \forall v \in V. \end{aligned}$$

An easy algorithm (Hochbaum 1982): Solve LP, let $S = \{v \in V : x_v^* \geq 1/2\}$.

Some Easy, Early Results

Algorithm: Solve LP, let $S = \{v \in V : x_v^* \geq 1/2\}$.

Clearly $\sum_{v \in S} w_j \leq 2 \sum_{v \in V} w_v x_v^* \leq 2\,\mathrm{OPT}$.

Also, since $x_u^* + x_v^* \geq 1$ for all $(u, v) \in V$, either $u \in S$ or $v \in S$ (or both).

Thus, this is a 2-approximation algorithm.

Some Easy, Early Results

Consider also the dual of LP relaxation:

$$\begin{aligned} \text{maximize} \quad & \sum_{(u,v)\in E} y_{(u,v)} \\ \text{subject to} \quad & \sum_{u:(u,v)\in E} y_{(u,v)} \leq w_v, \qquad \forall v \in V, \\ & y_{(u,v)} \geq 0, \qquad \forall (u,v) \in E. \end{aligned}$$

A *primal-dual* algorithm (Bar-Yehuda, Even 1981):

- Start with $S = \emptyset$, $y = 0$
- While S is not a vertex cover since $(u, v) \in E$ uncovered
  - Increase $y_{(u,v)}$ until $\sum_{a:(a,b)\in E} y_{(a,b)} = w_b$ for some $b \in V$
  - Add b to S

Some Easy, Early Results

$$\begin{aligned}
\text{maximize} \quad & \sum_{(u,v)\in E} y_{(u,v)} \\
\text{subject to} \quad & \sum_{u:(u,v)\in E} y_{(u,v)} \leq w_v, \qquad \forall v \in V, \\
& y_{(u,v)} \geq 0, \qquad \forall (u,v) \in E.
\end{aligned}$$

This algorithm is also a 2-approximation algorithm since

$$\begin{aligned}
\sum_{v\in S} w_v &= \sum_{v\in S} \sum_{u:(u,v)\in E} y_{(u,v)} \\
&\leq 2 \sum_{(u,v)\in E} y_{(u,v)} \leq 2\,\mathrm{OPT}
\end{aligned}$$

Cornell University

Approximate Min-Max Theorems

Most (not all!) approximation algorithms are approximate min-max theorems:

For minimization problems, we have some polytime computable bound R such that

$$R \leq \text{OPT} \leq \text{algorithm's soln} \leq \alpha R.$$

For vertex cover, R is the value of the linear programming relaxation.

Approximate Min-Max Theorems

Most (not all!) approximation algorithms are approximate min-max theorems:

For minimization problems, we have some polytime computable bound R such that

$$R \leq \text{OPT} \leq \text{algorithm's soln} \leq \alpha R.$$

For vertex cover, R is the value of the linear programming relaxation.

For maximization problems,

$$R \geq \text{OPT} \geq \text{algorithm's soln} \geq \alpha R.$$

For maximum satisfiability, R is total sum of clause weights.

Cornell University

Hardness Results

Starting in the 1990s, important progress in showing the nonexistence of approximation algorithms (if P $\neq$ NP).

- The breakthrough: Arora, Lund, Motwani, Sudan, and Szegedy (1998) use a new definition of NP in terms of *probabilistically checkable proofs* to show that no PTAS can exist for a large class of problems unless P $=$ NP.

Cornell University

Hardness Results

Starting in the 1990s, important progress in showing the nonexistence of approximation algorithms (if P $\neq$ NP).

- The breakthrough: Arora, Lund, Motwani, Sudan, and Szegedy (1998) use a new definition of NP in terms of *probabilistically checkable proofs* to show that no PTAS can exist for a large class of problems unless P $=$ NP.
- Lund and Yannakakis (1994) show for some $c < 1$, no $(c \ln n)$-approximation algorithm for set cover unless P $=$ NP.

Hardness Results

Starting in the 1990s, important progress in showing the nonexistence of approximation algorithms (if P $\neq$ NP).

- The breakthrough: Arora, Lund, Motwani, Sudan, and Szegedy (1998) use a new definition of NP in terms of *probabilistically checkable proofs* to show that no PTAS can exist for a large class of problems unless P $=$ NP.
- Lund and Yannakakis (1994) show for some $c < 1$, no $(c \ln n)$-approximation algorithm for set cover unless P $=$ NP.
- Feige (1998) improves to show that for *all* $c < 1$, no $(c \ln n)$-approximation for set cover unless NP has $O(n^{\log \log n})$ time algoritms.

Hardness Results

- Håstad (1999) (together with Zuckerman (2007)) shows that for any $\epsilon > 0$, no $O(n^{\epsilon-1})$-approximation algorithm for maximum clique problem unless P $=$ NP.

Hardness Results

- Håstad (1999) (together with Zuckerman (2007)) shows that for any $\epsilon > 0$, no $O(n^{\epsilon-1})$-approximation algorithm for maximum clique problem unless P $=$ NP.
- Håstad (2001) shows that for all $\epsilon > 0$, no $(\frac{7}{8} + \epsilon)$-approximation algorithm for MAX E3SAT unless P $=$ NP.

Hardness Results

- Håstad (1999) (together with Zuckerman (2007)) shows that for any $\epsilon > 0$, no $O(n^{\epsilon-1})$-approximation algorithm for maximum clique problem unless P $=$ NP.
- Håstad (2001) shows that for all $\epsilon > 0$, no $(\frac{7}{8} + \epsilon)$-approximation algorithm for MAX E3SAT unless P $=$ NP.
- Dinur and Safra (2002) show no α-approximation algorithm for vertex cover with $\alpha < 10\sqrt{5} - 21 \approx 1.36$ unless P $=$ NP.

Hardness result: Unique Games

Recent work based on *unique games conjecture*.

Unique Games Problem

Input:

- An undirected graph $G = (V, E)$
- Labels L
- Permutations $\pi_{uv} : L \to L$ for all $(u, v) \in E$

Goal: Find an assignment $\sigma : L \to V$ to maximize the number of satisfied edges; (u, v) satisfied if u labelled with $i \in L$, v labelled with $j \in L$ and $\pi_{uv}(i) = j$.

Conjecture (Unique Games Conjecture (UGC), Khot (2002))

For every $\delta > 0$, there exists some k such that for $|L| = k$ it is NP*-hard to decide whether*

- *At least $(1 - \delta)|E|$ edges are satisfiable*
- *or at most $\delta|E|$ edges are satisfiable*

Hardness results: Unique Games

Some examples:

- Khot and Regev (2008) show that given UGC, there is no α-approximation algorithm for vertex cover with $\alpha < 2$ unless P $=$ NP.

Hardness results: Unique Games

Some examples:

- Khot and Regev (2008) show that given UGC, there is no α-approximation algorithm for vertex cover with $\alpha < 2$ unless P $=$ NP.
- Raghavendra (2008), Raghavendra and Steurer (2009) show that the approximability of constraint satisfaction problems tied to integrality gap of semidefinite program, and given UGC, cannot do better than integrality gap unless P $=$ NP.

Hardness results: Unique Games

Some examples:

- Khot and Regev (2008) show that given UGC, there is no α-approximation algorithm for vertex cover with $\alpha < 2$ unless P $=$ NP.
- Raghavendra (2008), Raghavendra and Steurer (2009) show that the approximability of constraint satisfaction problems tied to integrality gap of semidefinite program, and given UGC, cannot do better than integrality gap unless P $=$ NP.
- Svensson (2010) shows that given a variant of the UGC, there is no α-approximation algorithm for a scheduling problem of Graham with $\alpha < 2$ unless P $=$ NP.

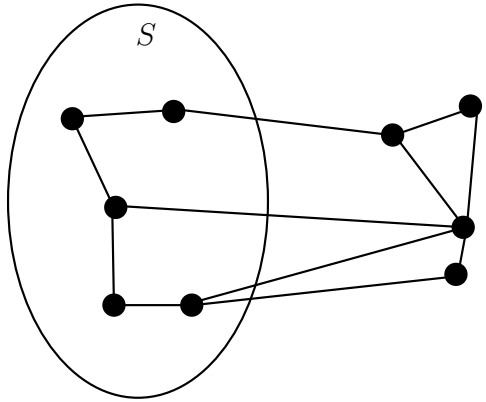
Next: ten open problems from our book.

Problem 10: A primal-dual algorithm for the maximum cut problem

Maximum Cut Problem

Input: An undirected graph $G = (V, E)$ with nonnegative edge weights $w_{ij} \geq 0$ for all $i, j \in V$.

Goal: Find a set of vertices $S \subseteq V$ that maximizes $\sum_{i \in S, j \notin S} w_{ij}$.



Problem 10: A primal-dual algorithm for the maximum cut problem

What's known?

- an $(\alpha - \epsilon)$-approximation algorithm using semidefinite programming (Goemans, W 1995) for

$$\alpha = \min_{-1 \le x \le 1} \frac{\frac{1}{\pi} \arccos(x)}{\frac{1}{2}(1-x)} \approx .87856,$$

and any $\epsilon > 0$.

Problem 10: A primal-dual algorithm for the maximum cut problem

What's known?

- an $(\alpha - \epsilon)$-approximation algorithm using semidefinite programming (Goemans, W 1995) for

$$\alpha = \min_{-1 \le x \le 1} \frac{\frac{1}{\pi} \arccos(x)}{\frac{1}{2}(1-x)} \approx .87856,$$

  and any $\epsilon > 0$.
- Assuming the unique games conjecture, no $(\alpha + \epsilon)$-approximation algorithm is possible unless P $=$ NP (Khot, Kindler, Mossel, O'Donnell 2007; Mossel, O'Donnell, Oleszkiewicz 2010)

Problem 10: A primal-dual algorithm for the maximum cut problem

What's known?

- an $(\alpha - \epsilon)$-approximation algorithm using semidefinite programming (Goemans, W 1995) for

$$\alpha = \min_{-1 \le x \le 1} \frac{\frac{1}{\pi} \arccos(x)}{\frac{1}{2}(1-x)} \approx .87856,$$

  and any $\epsilon > 0$.
- Assuming the unique games conjecture, no $(\alpha + \epsilon)$-approximation algorithm is possible unless P $=$ NP (Khot, Kindler, Mossel, O'Donnell 2007; Mossel, O'Donnell, Oleszkiewicz 2010)
- No β-approximation algorithm possible for constant $\beta > \frac{16}{17} \approx .941$ unless P $=$ NP (Håstad 1997).

Cornell University

Problem 10: A primal-dual algorithm for the maximum cut problem

The problem:

Solving the semidefinite program is computationally expensive. Can one obtain an $(\alpha - \epsilon)$-approximation algorithm for the problem via computationally easier means? E.g. a primal-dual algorithm?

Cornell University

Problem 10: A primal-dual algorithm for the maximum cut problem

The problem:

Solving the semidefinite program is computationally expensive. Can one obtain an $(\alpha - \epsilon)$-approximation algorithm for the problem via computationally easier means? E.g. a primal-dual algorithm?

A potential start:

(Trevisan, STOC 2009) gives a .531-approximation algorithm via an eigenvalue computation.

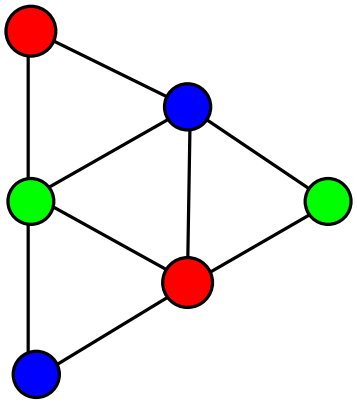
Cornell University

Problem 9: Coloring 3-colorable graphs

Coloring 3-colorable graphs

Input: An undirected, 3-colorable graph $G = (V, E)$.

Goal: Find a k-coloring of the graph with k as small as possible.



Problem 9: Coloring 3-colorable graphs

What's known?

- A poly-time algorithm using semidefinite programming that uses at most $\tilde{O}(n^{0.211})$ colors (Arora, Chlamtac, Charikar 2006)

Problem 9: Coloring 3-colorable graphs

What's known?

- A poly-time algorithm using semidefinite programming that uses at most $\tilde{O}(n^{0.211})$ colors (Arora, Chlamtac, Charikar 2006)
- It is NP-hard to decide if a graph needs only 3 colors or at least 5 colors (Khanna, Linial, Safra 2000)

Cornell University

Problem 9: Coloring 3-colorable graphs

What's known?

- A poly-time algorithm using semidefinite programming that uses at most $\tilde{O}(n^{0.211})$ colors (Arora, Chlamtac, Charikar 2006)
- It is NP-hard to decide if a graph needs only 3 colors or at least 5 colors (Khanna, Linial, Safra 2000)
- Assuming a variant of the unique games conjecture, for any constant $k > 3$, it is NP-hard to decide if a graph needs only 3 colors or at least k colors (Dinur, Mossel, Regev 2009)

Problem 9: Coloring 3-colorable graphs

What's known?

- A poly-time algorithm using semidefinite programming that uses at most $\tilde{O}(n^{0.211})$ colors (Arora, Chlamtac, Charikar 2006)
- It is NP-hard to decide if a graph needs only 3 colors or at least 5 colors (Khanna, Linial, Safra 2000)
- Assuming a variant of the unique games conjecture, for any constant $k > 3$, it is NP-hard to decide if a graph needs only 3 colors or at least k colors (Dinur, Mossel, Regev 2009)

The problem:

Give an algorithm that uses $O(\log n)$ colors for 3-colorable graphs (or show this is not possible modulo some complexity theoretic condition).

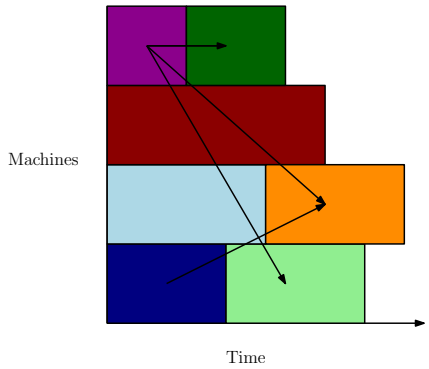
Problem 8: Scheduling related machines with precedence constraints

Scheduling related machines with precedence constraints

Input:

- n jobs with processing requirements $p_1, \ldots, p_n \geq 0$.
- m machines with speeds $s_1 \geq s_2 \geq \cdots \geq s_m > 0$.
- A precedence relation $\prec$ on jobs.

Goal: Find a schedule of minimum length in which all jobs are completely scheduled and if $j \prec j'$, then job j completes before job j' starts. Job j on machine i takes p_j/s_i units of time.



Cornell University

Problem 8: Scheduling related machines with precedence constraints

What's known?

- If machines are identical ($s_1 = s_2 = \cdots = s_m$) then there is a 2-approximation algorithm (Graham 1966).

Cornell University

Problem 8: Scheduling related machines with precedence constraints

What's known?

- If machines are identical ($s_1 = s_2 = \cdots = s_m$) then there is a 2-approximation algorithm (Graham 1966).
- For general case, an $O(\log m)$-approximation algorithm is known (Chudak and Shmoys 1999; Chekuri and Bender 2001).

Cornell University

Problem 8: Scheduling related machines with precedence constraints

What's known?

- If machines are identical ($s_1 = s_2 = \cdots = s_m$) then there is a 2-approximation algorithm (Graham 1966).
- For general case, an $O(\log m)$-approximation algorithm is known (Chudak and Shmoys 1999; Chekuri and Bender 2001).
- If machines are identical, and given a variant of the unique games conjecture, then no α-approximation algorithm is possible for $\alpha < 2$ unless P $=$ NP. (Svensson STOC 2010).

Problem 8: Scheduling related machines with precedence constraints

What's known?

- If machines are identical ($s_1 = s_2 = \cdots = s_m$) then there is a 2-approximation algorithm (Graham 1966).
- For general case, an $O(\log m)$-approximation algorithm is known (Chudak and Shmoys 1999; Chekuri and Bender 2001).
- If machines are identical, and given a variant of the unique games conjecture, then no α-approximation algorithm is possible for $\alpha < 2$ unless P $=$ NP. (Svensson STOC 2010).

The problem:

Give an α-approximation algorithm for some constant α, or show that $O(\log m)$ is the best possible modulo the unique games conjecture.

Cornell University

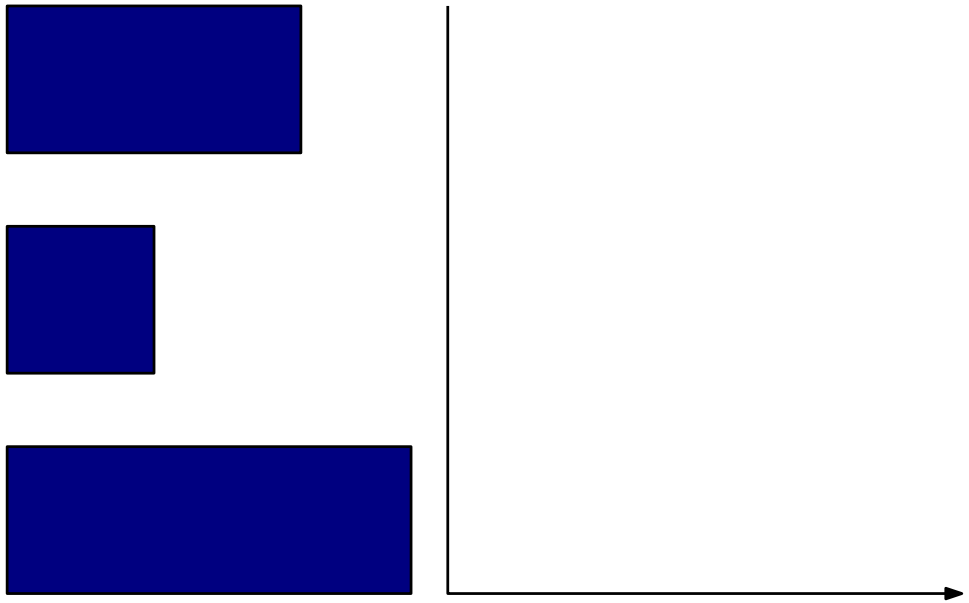
Problem 7: Scheduling unrelated machines

Scheduling unrelated machines

Input:

- m machines.
- n jobs with processing requirements p_{ij} for scheduling job j on machine i.

Goal: Find a schedule of minimum length.



Cornell University

Problem 7: Scheduling unrelated machines

What's known?

- A 2-approximation algorithm via LP rounding (Lenstra, Shmoys, Tardos 1990)

Problem 7: Scheduling unrelated machines

What's known?

- A 2-approximation algorithm via LP rounding (Lenstra, Shmoys, Tardos 1990)
- A 1.94-approximation algorithm if running time is $p_{ij} \in \{p_j, \infty\}$ for all i, j (Svensson STOC 2011).

Problem 7: Scheduling unrelated machines

What's known?

- A 2-approximation algorithm via LP rounding (Lenstra, Shmoys, Tardos 1990)
- A 1.94-approximation algorithm if running time is $p_{ij} \in \{p_j, \infty\}$ for all i, j (Svensson STOC 2011).
- No α-approximation algorithm with $\alpha < 3/2$ is possible unless P $=$ NP (Lenstra, Shmoys, Tardos 1990).

Problem 7: Scheduling unrelated machines

What's known?

- A 2-approximation algorithm via LP rounding (Lenstra, Shmoys, Tardos 1990)
- A 1.94-approximation algorithm if running time is $p_{ij} \in \{p_j, \infty\}$ for all i, j (Svensson STOC 2011).
- No α-approximation algorithm with $\alpha < 3/2$ is possible unless P $=$ NP (Lenstra, Shmoys, Tardos 1990).

The problem:

Give an α-approximation algorithm for $3/2 \leq \alpha < 2$, or show that this is not possible.

Cornell University

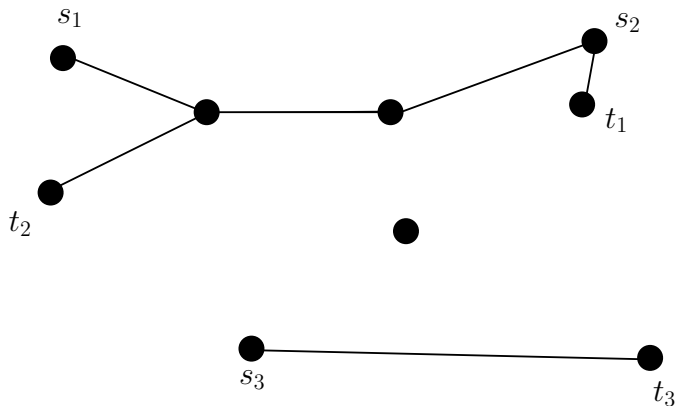
Problem 6: Generalized Steiner tree

Generalized Steiner tree

Input:

- Undirected graph $G = (V, E)$.
- Nonnegative edge costs $c_e \geq 0$ for all $e \in E$.
- k source-sink pairs s_1-t_1, s_2-t_2, ..., s_k-t_k.

Goal: Find edges F of minimum cost so that for each i, s_i and t_i are connected in (V, F).



Cornell University

Problem 6: Generalized Steiner tree

What's known?

- A primal-dual 2-approximation algorithm (Agrawal, Klein, Ravi 1995; see also Goemans and W 1995).

Problem 6: Generalized Steiner tree

What's known?

- A primal-dual 2-approximation algorithm (Agrawal, Klein, Ravi 1995; see also Goemans and W 1995).
- If $s_i = s$ for all i, have the Steiner tree problem; then a 1.39-approximation algorithm known using LP rounding (Byrka, Grandoni, Rothvoß, Sanità STOC 2010).

Problem 6: Generalized Steiner tree

What's known?

- A primal-dual 2-approximation algorithm (Agrawal, Klein, Ravi 1995; see also Goemans and W 1995).
- If $s_i = s$ for all i, have the Steiner tree problem; then a 1.39-approximation algorithm known using LP rounding (Byrka, Grandoni, Rothvoß, Sanità STOC 2010).
- No α-approximation algorithm possible for Steiner tree for $\alpha < \frac{96}{95} \approx 1.01$ unless P $=$ NP (Chlebík, Chlebíková 2008)

Problem 6: Generalized Steiner tree

What's known?

- A primal-dual 2-approximation algorithm (Agrawal, Klein, Ravi 1995; see also Goemans and W 1995).
- If $s_i = s$ for all i, have the Steiner tree problem; then a 1.39-approximation algorithm known using LP rounding (Byrka, Grandoni, Rothvoß, Sanità STOC 2010).
- No α-approximation algorithm possible for Steiner tree for $\alpha < \frac{96}{95} \approx 1.01$ unless P $=$ NP (Chlebík, Chlebíková 2008)

The problem

Find an α-approximation algorithm for the generalized Steiner tree problem for constant $\alpha < 2$.

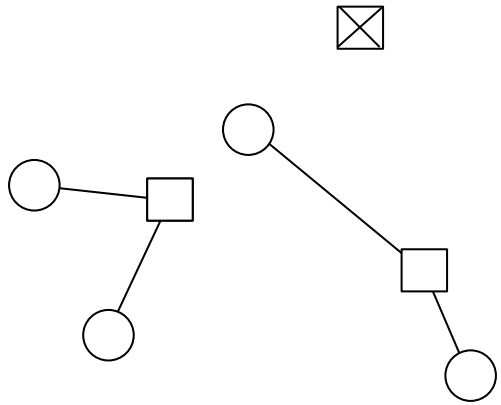
Problem 5: Capacitated facility location

Capacitated facility location

Input:

- A set F of facilities; each $i \in F$ has facility cost $f_i \geq 0$.
- A set D of clients.
- A metric c_{ij} on locations $i, j \in F \cup D$.
- A capacity U on each facility.

Goal: Find $S \subset F$ and assignment $\sigma : D \to S$ such that $|\sigma^{-1}(i)| \leq U$ for all $i \in S$ that minimizes $\sum_{i \in S} f_i + \sum_{j \in D} c_{\sigma(j),j}$.



Cornell University

Problem 5: Capacitated facility location

What's known?

A local search algorithm: Let S be a set of currently open facilities. As long as it improves the overall cost,

- **Add**: $S \leftarrow S \cup \{i\}$ for $i \notin S$;
- **Drop**: $S \leftarrow S - \{i\}$ for $i \in S$; or
- **Swap**: $S \leftarrow S \cup \{i\} - \{j\}$ for $i \notin S$, $j \in S$.

Problem 5: Capacitated facility location

What's known?

A local search algorithm: Let S be a set of currently open facilities. As long as it improves the overall cost,

- **Add**: $S \leftarrow S \cup \{i\}$ for $i \notin S$;
- **Drop**: $S \leftarrow S - \{i\}$ for $i \in S$; or
- **Swap**: $S \leftarrow S \cup \{i\} - \{j\}$ for $i \notin S$, $j \in S$.

- Can show this gives an $(\alpha + \epsilon)$-approximation algorithm for
  - $\alpha = 8$ (Koropolu, Plaxton, Rajaraman 2000)
  - $\alpha = 6$ (Chudak, W 2005)
  - $\alpha = 3$ (Aggarwal et al. 2010)

Problem 5: Capacitated facility location

Let S be local optimal solution, with assignment σ; let S^* be optimal solution with assignment σ^*.
For any $i \in O$, since locally optimal,

$$f_i + \sum_{j \in \sigma^{*-1}(i)} (c_{ij} - c_{\sigma(j),j}) \geq 0.$$

Problem 5: Capacitated facility location

Let S be local optimal solution, with assignment σ; let S^* be optimal solution with assignment σ^*.
For any $i \in O$, since locally optimal,

$$f_i + \sum_{j \in \sigma^{*-1}(i)} (c_{ij} - c_{\sigma(j),j}) \geq 0.$$

Summing over all $i \in O$,

$$\sum_{i \in O} f_i + \sum_{j \in D} (c_{\sigma^*(j),j} - c_{\sigma(j),j}) \geq 0,$$

or

$$\sum_{j \in D} c_{\sigma(j),j} \leq \sum_{i \in O} f_i + \sum_{j \in D} c_{\sigma^*(j),j}.$$

Problem 5: Capacitated facility location

The problem:

Is there a polytime-computable relaxation R of the problem within a constant factor of the optimal?

Problem 5: Capacitated facility location

The problem:

Is there a polytime-computable relaxation R of the problem within a constant factor of the optimal?

Or, what's the approximate min-max relaxation?

$$R \leq \text{OPT} \leq \text{algorithm's soln} \leq \alpha R.$$

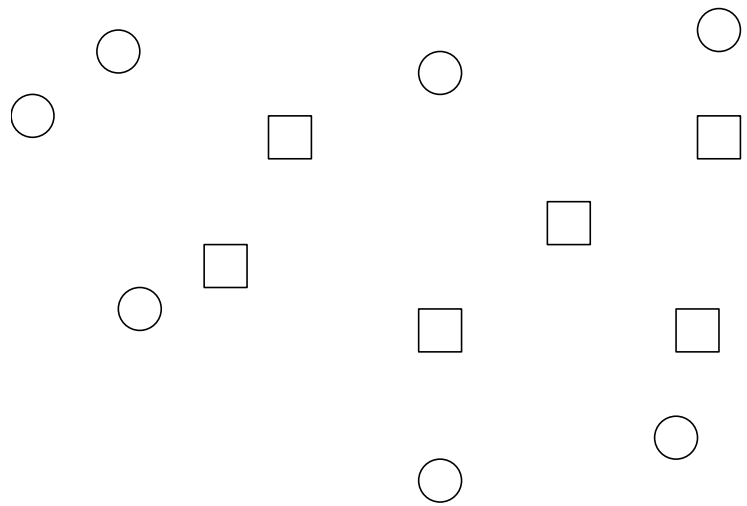
Problem 4: Survivable network design

Survivable network design

Input:

- An undirected graph $G = (V, E)$
- Costs $c_e \geq 0$ for all $e \in E$
- Integer connectivity requirements r_{ij} for all $i, j \in V$

Goal: Find a minimum-cost set of edges F so that for all $i, j \in V$, there are at least r_{ij} edge-disjoint paths between i and j in (V, F).



Cornell University

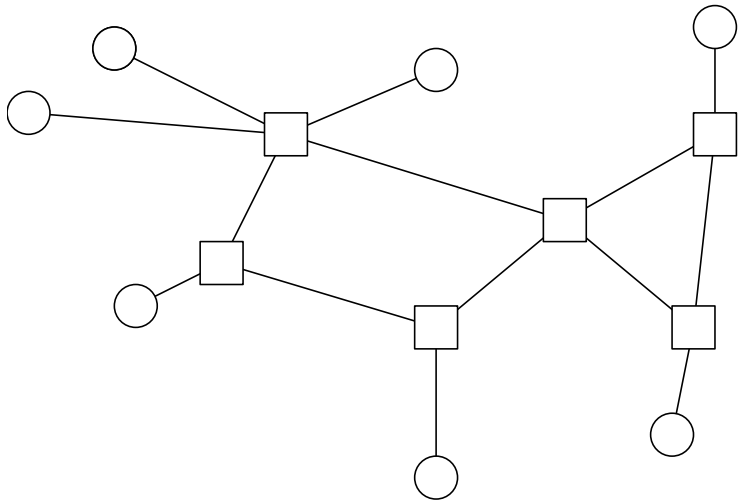
Problem 4: Survivable network design

Survivable network design

Input:

- An undirected graph $G = (V, E)$
- Costs $c_e \geq 0$ for all $e \in E$
- Integer connectivity requirements r_{ij} for all $i, j \in V$

Goal: Find a minimum-cost set of edges F so that for all $i, j \in V$, there are at least r_{ij} edge-disjoint paths between i and j in (V, F).



Problem 4: Survivable network design

What's known?

- A primal-dual $2H_R$-approximation algorithm (Goemans, Goldberg, Plotkin, Shmoys, Tardos, W '94), where $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ and $R = \max_{i,j} r_{ij}$.
- An LP rounding 2-approximation algorithm (Jain 2001)

Cornell University

Problem 4: Survivable network design

What's known?

- A primal-dual $2H_R$-approximation algorithm (Goemans, Goldberg, Plotkin, Shmoys, Tardos, W '94), where $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ and $R = \max_{i,j} r_{ij}$.
- An LP rounding 2-approximation algorithm (Jain 2001)

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e \\ \text{subject to} \quad & \sum_{e \in \delta(S)} x_e \geq \max_{i \in S, j \notin S} r_{ij}, \qquad \forall S \subset V, \\ & 0 \leq x_e \leq 1, \qquad \forall e \in E. \end{aligned}$$

Theorem (Jain 2001)

For any basic feasible solution x^* of the LP relaxation, there exists some edge $e \in E$ such that $x^*_e \geq 1/2$.

Problem 4: Survivable network design

The problem:

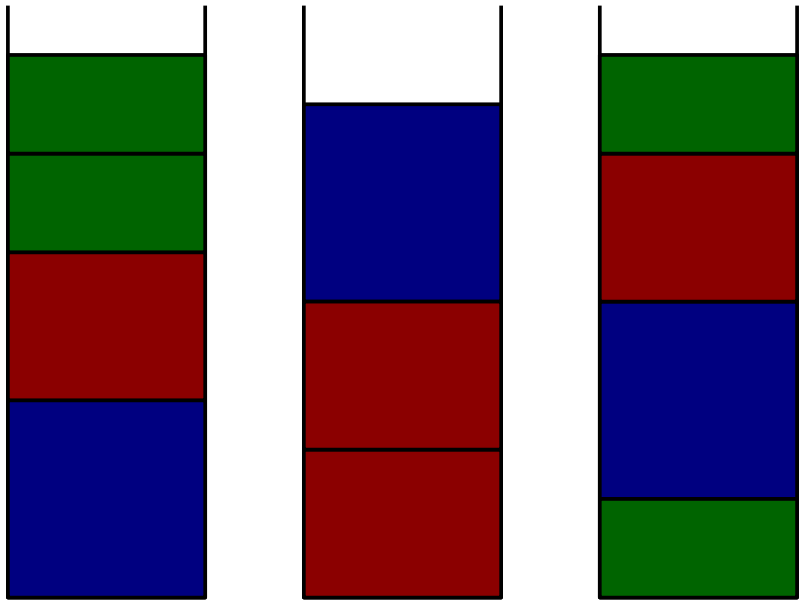
Is there a 2-approximation algorithm that doesn't require solving the LP? E.g. a primal-dual algorithm?

Problem 3: Bin packing

Bin packing

Input: b_i pieces of size s_i, $0 < s_i < 1$, for $i = 1, \ldots, m$

Goal: Find a packing of pieces into bins of size 1 that minimizes the total number of bins used



Problem 3: Bin packing

What's known?

An LP-rounding algorithm that uses $\mathrm{OPT} + O(\log^2 \mathrm{OPT})$ bins (Karmarkar, Karp 1982)

Cornell University

Problem 3: Bin packing

What's known?

An LP-rounding algorithm that uses $\mathrm{OPT} + O(\log^2 \mathrm{OPT})$ bins (Karmarkar, Karp 1982)

Enumerate all N possible ways of packing a bin. jth configuration uses a_{ij} pieces of size i.

$$\text{minimize} \quad \sum_{j=1}^{N} x_j$$

$$\text{subject to} \quad \sum_{j=1}^{N} a_{ij} x_j \geq b_i, \qquad i = 1, \ldots, m,$$

$$x_j \text{ integer}, \qquad j = 1, \ldots, N.$$

Cornell University

Problem 3: Bin packing

The problem:

Find a polytime algorithm that uses at most OPT $+c$ bins for some constant c.

Note that there are instances known for which

$$\text{OPT} > \lceil LP \rceil + 1,$$

but currently no known instances for which

$$\text{OPT} > \lceil LP \rceil + 2.$$

Cornell University

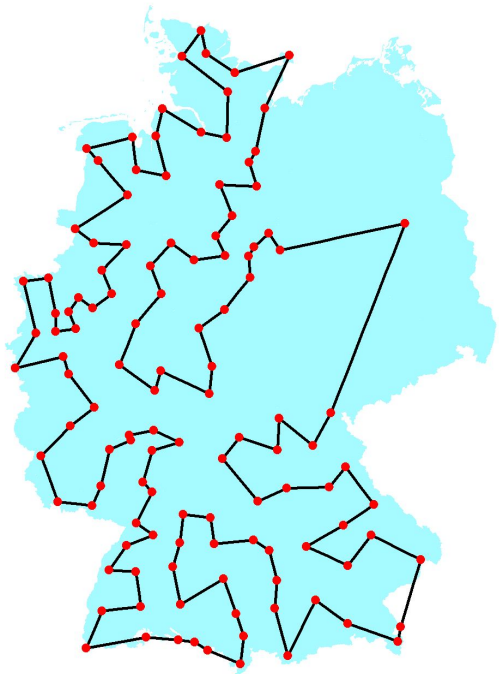
Problems 1 and 2: the traveling salesman problem

Traveling salesman problem

Input:

- Set of cities V
- Travel costs c_{ij} such that $c_{ij} \leq c_{ik} + c_{kj}$ for all $i, j, k \in V$

Goal: Find a minimum-cost tour of all the cities



Cornell University

Problems 1 and 2: the traveling salesman problem

Problem 2: the asymmetric case ($c_{ij} \neq c_{ji}$)

What's known?

- An $O(\log n)$-approximation algorithm (Frieze, Galbiati, Maffioli 1982)
- An LP rounding $O(\log n / \log \log n)$-approximation algorithm (Asadpour, Goemans, Madry, Oveis Gharan, Saberi 2010)
- Can't approximate better than $\frac{117}{116} \approx 1.008$ unless P $=$ NP (Papadimitriou, Vempala 2006)

Problems 1 and 2: the traveling salesman problem

$$\text{minimize} \quad \sum_{i,j \in V} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{j \in V} x_{ij} = \sum_{j \in V} x_{ji} \qquad i \in V,$$

$$\sum_{i \in S, j \notin S} x_{ij} \geq 1 \qquad \forall S \subset V$$

$$x_{ij} \geq 0 \qquad \forall i, j \in V.$$

No instance known for which the integrality gap is worse than 2 (Charikar, Goemans, Karloff 2006)

Problems 1 and 2: the traveling salesman problem

$$\text{minimize} \quad \sum_{i,j \in V} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{j \in V} x_{ij} = \sum_{j \in V} x_{ji} \qquad i \in V,$$

$$\sum_{i \in S, j \notin S} x_{ij} \geq 1 \qquad \forall S \subset V$$

$$x_{ij} \geq 0 \qquad \forall i, j \in V.$$

No instance known for which the integrality gap is worse than 2 (Charikar, Goemans, Karloff 2006)

The problem:

Find an α-approximation algorithm for α constant for the asymmetric case.

Cornell University

Problems 1 and 2: the traveling salesman problem

Problem 1: the symmetric case $c_{ij} = c_{ji}$ for all $i, j \in V$

What's known?

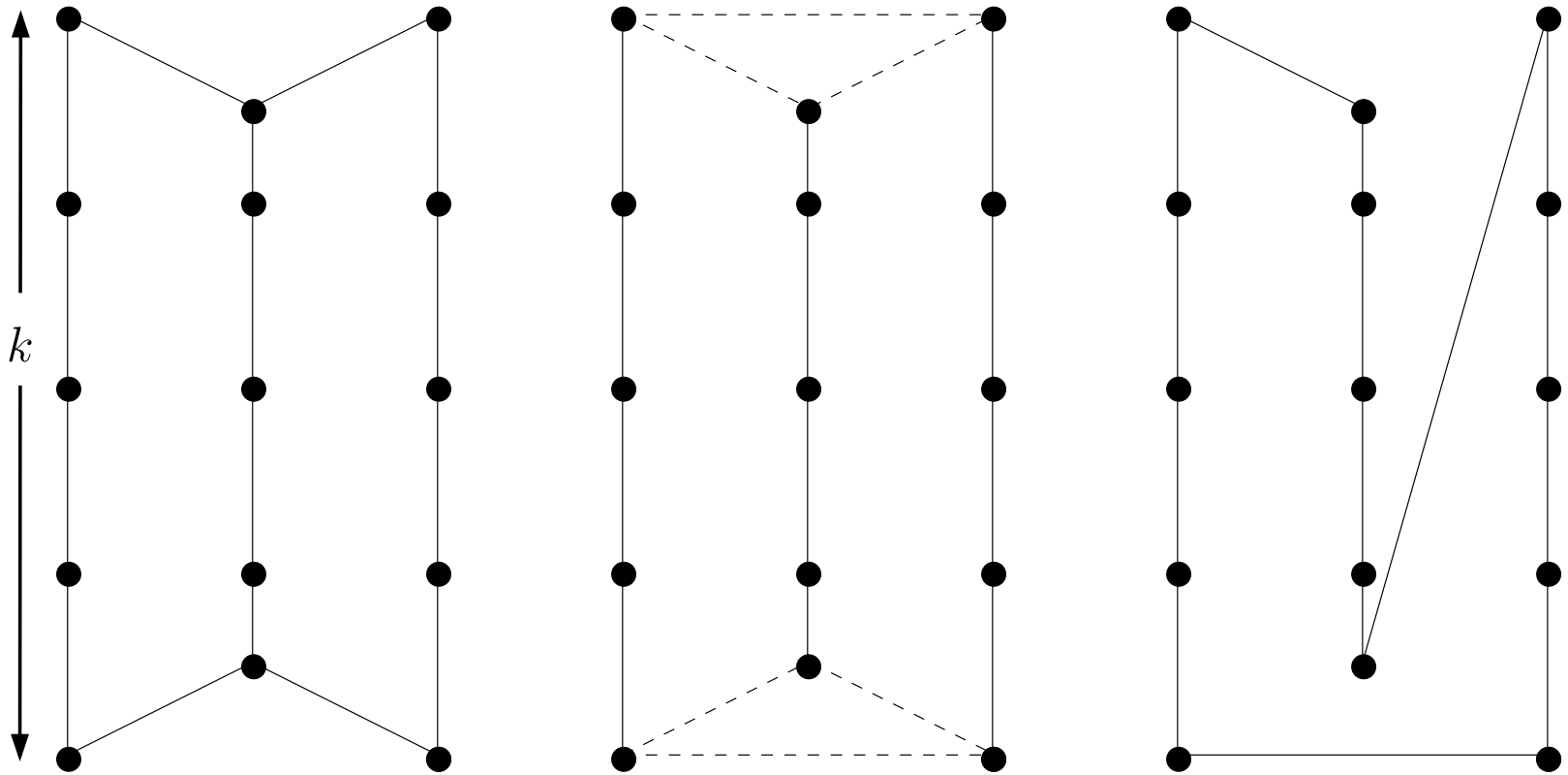
- A $\frac{3}{2}$-approximation algorithm (Christofides 1976)
- Can't approximate better than $\frac{220}{219} \approx 1.004$ unless P = NP (Papadimitriou, Vempala 2006)

Cornell University

Problems 1 and 2: the traveling salesman problem

$$\begin{aligned}
\text{minimize} \quad & \sum_{i,j \in V : i<j} c_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{j \in V : i<j} x_{ij} + \sum_{j \in V : i>j} x_{ji} = 2 && i \in V \\
& \sum_{i \in S, j \notin S \text{ or } i \notin S, j \in S} x_{ij} \geq 2 && \forall S \subset V \\
& x_{ij} \geq 0 && \forall i, j \in V, i < j.
\end{aligned}$$

Integrality gap at most $\frac{3}{2}$ (Wolsey 1980). No instance known with gap worse than $\frac{4}{3}$.


k

Problems 1 and 2: the traveling salesman problem

The problem:

Find an α-approximation algorithm for constant $\alpha < \frac{3}{2}$.

Cornell University

An observation

No open problem of the form "this problem has an α-approximation algorithm for constant α, find a PTAS."

Success in computation?

The field has certainly successfully generated interesting algorithmic ideas and mathematical understandings of approximate computation.

Success in computation?

The field has certainly successfully generated interesting algorithmic ideas and mathematical understandings of approximate computation.

But how much effect on actual computational practice?

Some cases in network design codes:

- Mihail, Shallcross, Dean, Mostrel (1996): Use primal-dual survivable network design algorithm
- Johnson, Minkoff, Phillips (2000): Use primal-dual prize-collecting Steiner tree algorithm

Also cases for problems that are theoretically solvable in polytime, but for which approximation algorithms are much faster: e.g. Müller, Radke, Vygen (2010)

But in graph partitioning and traveling salesman problem, most used codes and ideas are from outside the area.

Cornell University

Success in computation?

The field has certainly successfully generated interesting algorithmic ideas and mathematical understandings of approximate computation.

But how much effect on actual computational practice?

Some cases in network design codes:

- Mihail, Shallcross, Dean, Mostrel (1996): Use primal-dual survivable network design algorithm
- Johnson, Minkoff, Phillips (2000): Use primal-dual prize-collecting Steiner tree algorithm

Also cases for problems that are theoretically solvable in polytime, but for which approximation algorithms are much faster: e.g. Müller, Radke, Vygen (2010)

But in graph partitioning and traveling salesman problem, most used codes and ideas are from outside the area.

Can the theory help explain the realities of practice?

Cornell University

Lightweight approximation algorithms

Perhaps part of the problem of adopting approximation algorithms is that the theoretically best algorithms are too computationally demanding compared to heuristics.

Examples:

- Best approximation algorithm for survivable network design requires solving LP via ellipsoid method
- Best approximation algorithm for max cut requires solving semidefinite program

Can we create *lightweight* approximation algorithms that deliver the same performance guarantees but with more practical computational requirements?

Cornell University

Maturity as a field

The area has become relatively mature, so substantial progress on open problems is likely to be difficult.

Maturity as a field

The area has become relatively mature, so substantial progress on open problems is likely to be difficult.

But still possible! For example,

- $O(\log n / \log \log n)$-approximation algorithm for asymmetric traveling salesman problem (Asadpour et al. SODA 2010)
- A 1.39-approximation algorithm for the minimum-cost Steiner tree problem (Byrka et al. STOC 2010).
- Some progress announced on symmetric TSP (when metric is from an unweighted graph) (Oveis Gharan, Saberi, Singh, December 2010)

Maturity as a field

The area has become relatively mature, so substantial progress on open problems is likely to be difficult.

But still possible! For example,

- $O(\log n / \log \log n)$-approximation algorithm for asymmetric traveling salesman problem (Asadpour et al. SODA 2010)
- A 1.39-approximation algorithm for the minimum-cost Steiner tree problem (Byrka et al. STOC 2010).
- Some progress announced on symmetric TSP (when metric is from an unweighted graph) (Oveis Gharan, Saberi, Singh, December 2010)

And perhaps you!

The End

Thanks for your attention.

www.designofapproxalgs.com

Cornell University