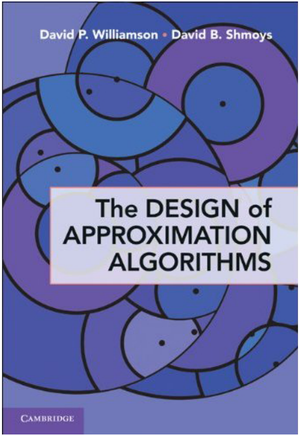
Some Open Problems in Approximation Algorithms

David P. Williamson

School of Operations Research and Information Engineering
Cornell University

August 18, 2011
APPROX 2011

The book



Electronic version at www.designofapproxalgs.com.

Outline

- Three FAQs about the book
- Our ten open problems (Chapter 17)
- Some thoughts about the field

FAQ #1

FAQ #1: How long did it take to write the book?

FAQ #1

FAQ #1: How long did it take to write the book?

Answer: 13-14 years, depending on how you count.

FAQ #1

FAQ #1: How long did it take to write the book?

Answer: 13-14 years, depending on how you count.

Fax from July 16, 1997 with book outline.

The first outline

SENT BY: 7-16-97 ; 11:45 ; 914 945 3434;# 1/ 2

Co.	Co.
Dept.	Phone #
Fax #	Fax #

I. Introduction
  A. What are approximation algorithms? Why approximation algorithms?
  B. An introduction to approximation algorithms: 2-approximation algorithms for vertex cover
    1. Unweighted via matchings
    2. Linear programming and deterministic rounding (Hochbaum; Nemhauser-Trotter?)
    3. Constructing a dual (Bar-Yehuda and Even)
    4. Randomization (Pitt)
  C. Set cover
    1. Extending arguments: Hochbaum, Bar-Yehuda & Even, Pitt
    2. A Better Algorithm for Unweighted (Johnson, Lovasz)
    3. A Better Algorithm for Weighted (Chvatal)

(Part →) II. Classic Approximation Algorithms for Classic Problems
  (Chapter) A. Traveling Salesman Problem
    (Section) 1. Doubling a min-cost spanning tree — Nearest addition
    2. Christofides' Algorithm
    3. The Prize-Collecting TSP
      i. Solving large LPs: the ellipsoid method
      ii. Deterministic Rounding (Bienstock et al.)
  B. Steiner Trees and a simple 2-approximation algorithm
  C. Knapsack and Dynamic Programming (Ibarra-Kim)

  (Contrast between — complete enumeration + augmentation; — dynamic prog. + augmentation)

  D. Bin-Packing
    1. Simple proof of a Johnson algorithm? (2OPT+1)
    2. de la Vega and Lueker
    3. Karmarkar and Karp
  E. Cliques, Independent Sets, and Colorings
    1. Johnson's algorithm
    2. Widgerson's O(\sqrt{n}) algorithm for 3-colorable graphs
    3. Boppana-Halldorsson (~~how hard is this~~?)
    4. Edge-coloring (Vizing)
  F. Scheduling
    1. Graham for P||C_max (+ PTAS for fixed m) LPT?
    2. Dynamic programming: PTAS for P||C_max (Hochbaum & Shmoys)

III. The Power of Randomization
  A. Flipping Coins and Easy Randomization
    1. MAX CUT (Sahni-Gonzalez) and MAX SAT (Johnson)
    2. Flipping Bent Coins: MAX SAT (Lieberherr and Specker)
    3. Derandomization: the Method of Conditional Expectations
  B. Randomized Rounding
    1. MAX SAT (Goemans-Williamson)
    2. Multicommodity Flows (Raghavan-Thompson)
    3. Prize-collecting TSP revisited (Goemans)
    4. Set cover and oversampling? VC dimension?
    5. MAX CUT in dense graphs (Arora-Karger-Karpinski)
  C. Rounding Semidefinite Programs

JUL 16 '97 12:07 PAGE.01

Cornell University

The first outline (2)

SENT BY: 7-16-97 ; 11:45 ; → 914 945 3434;# 2/ 2

1. MAX CUT (Goemans-Williamson)
2. Coloring (Karger-Motwani-Sudan)

IV. Finding cuts in graphs & metric methods & applications
- A. A warm-up: multiway cuts (Dahlhaus et al.)
- B. The metric method: multicuts (Garg-Vazirani-Yannakakis) LRU?
- C. Balanced separators (Even-Naor-Rao-Schieber)
- D. Applications of balanced separators (linear arrangement, etc.)

V. Network Design Problems, filtering, and the primal-dual method
- A. Facility location and filtering
  1. Early results (Dyer-Frieze, Hochbaum-Shmoys)
  2. Filtering (Shmoys-Tardos-Aardal) ??
- B. The primal-dual method
  1. 0-1 proper functions (Goemans-Williamson)
  2. Survivable Network Design with multiple edge copies (Agrawal-Klein-Ravi, Goemans-Williamson)
  3. Prize-collecting TSP revisited again (Goemans-Williamson)?
  4. k-MST (Garg)?
  5. A non-network design application: feedback vertex sets (Bafna-Berman-Fujito as interpreted by Chudak-Goemans-Hochbaum-Williamson)
- C. More network design
  1. Steiner trees (Zelikovsky)
  2. Low-degree trees (Furer-Raghavachari)

VI. Dynamic Programming
- A. Baker's PTAS for planar graphs ←
- B. Arora's PTAS for ~~Euclidean problems~~ TSP → Mitchell for Steiner Tree

→ VII. More Scheduling

VIII. Hardness results and implications
- A. Statement of ALMSS and proof of hardness of MAX SAT
- B. The label-cover theorem and implications (?)
- C. Statement of hardness of clique, coloring, Hastad's result for equations over GF[2]

VII

A. Simple Rules with Simple Analyses
1. EDD for $1|r_j|L_{max}$ (in head-body-tail form)
2. [illegible] EPT for $1|r_j|\Sigma C_j$

B. LP-based rules
1. $\bar{C}_j$ for $1|prec|\Sigma w_j C_j$
2. Generalized assignment
3. [illegible] 3. SRR for $P||\Sigma w_j C_j$

C. Randomized Improvements
1. e/e-1 for $1|r_j|\Sigma C_j$
2. 2-for $R|r_{ij}|\Sigma w_j C_j$

JUL 16 '97 12:07 PAGE.02

Cornell University

A later outline

I. Introduction – the book in miniature
  A. The what/why of approx. algs.
  B. An introduction to approx. algs.: Set cover
    1. Unweighted, greedy
    2. Deterministic rounding (Hochbaum)
    3. Introducing a dual: Hochbaum's dual rounding
    4. Primal-dual (Bar-Yehuda & Even)
    5. Weighted greedy (Chvátal)

Any way to introduce a randomized alg here?

II TSP & deterministic rounding??
  A. TSP
    1. Doubling an MST & nearest addition
    2. Christofides
    3. The PCST
      a. Ellipsoid method
      b. Deterministic rounding (but need parsimonious property)
  B. k-median – easy det. rounding from CGST ??

Steiner tree as exercise

III Dynamic Programming & Bin packing
  A. Knapsack
  B. P|| C_{max}
    1. List scheduling
    2. PTAS (HS)
  C. Bin packing
    1. First fit
    2. DelaVega & Lueker's $(1+\epsilon)OPT+1$
    3. Karmarkar & Karp

LPT as exercise

IV Randomization
  A. Max Sat: a case study
    1. Randomized Johnson
    2. Derandomization: conditional expectations
    3. Bent coins: Lieberherr & Specker
    4. Randomized rounding: GW
    5. Non-uniform rand round: GW
  B. More non-uniform rand round: PCST (Goemans)

Alternate GW algs. as ex.

Cornell University

A later outline (2)

C. Max Cut: a case study
  1. Easy $\frac{1}{2}$- (SG)
  2. PTAS in dense graphs (AKK)
     Intro to Chernoff bounds

Rg-Th mc flows as ex.

~~VI. Non-unifor~~

D. SDP
  1. Max Cut
  2. Quadratic programming (Nesterov)
  3. Coloring
     a. Widgerson's $O(\sqrt{n})$ for 3-colorable graphs
     b. MKS alg. 1
     a. MKS alg. 2
  4. Betweeness? Bisection?

V Cuts & Metrics

A. Multiway cuts
  1. Easy 2 (Dalhaus et al.)
  2. Improved LP + L_1 metric view (CKR)

B. Multicuts (GVY)

C. Balanced separators (ENRS)
  1. applications thereof

D. Sparsest cut & LLR low distortion embeddings

(Kahale as ex.)

VI The Primal-dual method

A. Intro → Generalized Steiner trees

B. Facility Location (GV)

C. PCST ??

D. Lagrangean relaxation
  1. k-median
  2. k-MST (5)

E. FVS?

A later outline (3)

VII Local search
- A. Facility location (Guha & Charikar)
- B. Low degree spanning tree (Furer & Raghavachari)

VIII Advanced deterministic rounding: Jain

IX Advanced dynamic programming: Arora/Mitchell

X. Advanced scheduling
- A. Simple rules w/ simple analyses
  1. EDD for $1|r_j|L_{max}$
  2. EPT for $1|r_j|\Sigma w_j C_j$
- B. LP based rules
  1. $\bar{C}_j$ for $1|prec|\Sigma w_j C_j$
  2. Generalized assignment
  3. SRR for $P||\Sigma w_j C_j$
- C. Randomization
  1. $e/e-1$ for $1|r_j|\Sigma C_j$
  2. 2 for $R||\Sigma w_j C_j$

XI Hardness bounds
- A. Statement of ALMSS ⇒ no PTAS for MAX SAT
- B. Label cover?
- C. Hardness of clique, coloring, eqs. over GF[2]

Some principles

Guiding principles

- Presentation of simple, elegant algorithms and analysis
- Techniques widely applicable
  (over exhaustive, but hodge-podge coverage of particular problems & results)
- Illustrate the power of new techiques, formulations by revisiting core problems many times:
  PCST, k-median/fac loc, FVS, Max Sat, Max Cut, Multiway cut

Final structure

Intro: Set cover	
Greedy and local search	Further greedy and local search
Dynamic programming	Further dynamic programming
Deterministic rounding	Further deterministic rounding
Randomized rounding	Further randomized rounding
SDP	Further SDP
Primal-dual	Further primal-dual
Cuts and metrics	Further cuts and metrics
	Hardness
	Open problems

Some nice things about the construction

Uncapacitated facility location

- Deterministic rounding: 4 (Chudak and Shmoys)

Some nice things about the construction

Uncapacitated facility location

- Deterministic rounding: 4 (Chudak and Shmoys)
- Randomized rounding: 3 (Chudak and Shmoys)

Some nice things about the construction

Uncapacitated facility location

- Deterministic rounding: 4 (Chudak and Shmoys)
- Randomized rounding: 3 (Chudak and Shmoys)
- Primal-dual: 3 (Jain and Vazirani)

Some nice things about the construction

Uncapacitated facility location

- Deterministic rounding: 4 (Chudak and Shmoys)
- Randomized rounding: 3 (Chudak and Shmoys)
- Primal-dual: 3 (Jain and Vazirani)
- Further greedy and local search: 3, $1 + \sqrt{2}$ (Charikar, Guha), 2 (Jain et al.)

Some nice things about the construction

Uncapacitated facility location

- Deterministic rounding: 4 (Chudak and Shmoys)
- Randomized rounding: 3 (Chudak and Shmoys)
- Primal-dual: 3 (Jain and Vazirani)
- Further greedy and local search: 3, $1 + \sqrt{2}$ (Charikar, Guha), 2 (Jain et al.)
- Further randomized rounding: $1 + \frac{2}{e}$ (Chudak and Shmoys)

Cornell University

Problems or techniques?

A pedagogical issue: teach problems or techniques?

Problems or techniques?

A pedagogical issue: teach problems or techniques?

Hard because historically the two are interwined; for example:

- Deterministic rounding/primal-dual and set cover/vertex cover (Hochbaum, Bar-Yehuda and Even)
- Randomized rounding and integer multicommodity flow (Raghavan and Thompson)
- SDP and max cut (Goemans and W)
- Region-growing and multicut (Garg, Vazirani, Yannakakis)

Problems or techniques?

A pedagogical issue: teach problems or techniques?

Hard because historically the two are interwined; for example:

- Deterministic rounding/primal-dual and set cover/vertex cover (Hochbaum, Bar-Yehuda and Even)
- Randomized rounding and integer multicommodity flow (Raghavan and Thompson)
- SDP and max cut (Goemans and W)
- Region-growing and multicut (Garg, Vazirani, Yannakakis)

If techniques, then some algorithms are hard to categorize; e.g. what is Christofides' algorithm?

Cornell University

Problems or techniques?

A pedagogical issue: teach problems or techniques?

Hard because historically the two are interwined; for example:

- Deterministic rounding/primal-dual and set cover/vertex cover (Hochbaum, Bar-Yehuda and Even)
- Randomized rounding and integer multicommodity flow (Raghavan and Thompson)
- SDP and max cut (Goemans and W)
- Region-growing and multicut (Garg, Vazirani, Yannakakis)

If techniques, then some algorithms are hard to categorize; e.g. what is Christofides' algorithm?
If problems, then what is the main takeaway of the course?

FAQ #2

FAQ #2: Why didn't you cover:

FAQ #2

FAQ #2: Why didn't you cover:

- Online algorithms

FAQ #2

FAQ #2: Why didn't you cover:

- Online algorithms
- Streaming algorithms

FAQ #2

FAQ #2: Why didn't you cover:

- Online algorithms
- Streaming algorithms
- Geometric approximation algorithms (e.g. coresets)

Cornell University

FAQ #2

FAQ #2: Why didn't you cover:

- Online algorithms
- Streaming algorithms
- Geometric approximation algorithms (e.g. coresets)
- The multiplicative update algorithm

FAQ #2

FAQ #2: Why didn't you cover:

- Online algorithms
- Streaming algorithms
- Geometric approximation algorithms (e.g. coresets)
- The multiplicative update algorithm
- Directed multicut
- ...

Cornell University

FAQ #2

FAQ #2: Why didn't you cover:

- Online algorithms
- Streaming algorithms
- Geometric approximation algorithms (e.g. coresets)
- The multiplicative update algorithm
- Directed multicut
- ...

Answers (choose one at random):

Cornell University

FAQ #2

FAQ #2: Why didn't you cover:

- Online algorithms
- Streaming algorithms
- Geometric approximation algorithms (e.g. coresets)
- The multiplicative update algorithm
- Directed multicut
- ...

Answers (choose one at random):

- It would have taken another 13-14 years...

FAQ #2

FAQ #2: Why didn't you cover:

- Online algorithms
- Streaming algorithms
- Geometric approximation algorithms (e.g. coresets)
- The multiplicative update algorithm
- Directed multicut
- . . .

Answers (choose one at random):

- It would have taken another 13-14 years...
- ..and another 500+ pages...

FAQ #2

FAQ #2: Why didn't you cover:

- Online algorithms
- Streaming algorithms
- Geometric approximation algorithms (e.g. coresets)
- The multiplicative update algorithm
- Directed multicut
- ...

Answers (choose one at random):

- It would have taken another 13-14 years...
- ..and another 500+ pages...
- Luckily, Sariel Har-Peled just wrote a book on geometric approximation algorithms (362 pages).

Cornell University

FAQ #2

FAQ #2: Why didn't you cover:

- Online algorithms
- Streaming algorithms
- Geometric approximation algorithms (e.g. coresets)
- The multiplicative update algorithm
- Directed multicut
- ...

Answers (choose one at random):

- It would have taken another 13-14 years...
- ..and another 500+ pages...
- Luckily, Sariel Har-Peled just wrote a book on geometric approximation algorithms (362 pages).
- We consciously decided not to write about approximating problems in P.

A definitional question

Some of the items taken from the following blog post by David Eppstein (7 Nov 2010):

Approximate book

There's a new book out by Williamson and Shmoys on approximation algorithms, *The Design of Approximation Algorithms*, available electronically for free as a pdf download.

For a book that claims to be a comprehensive reference on approximation algorithms, suitable for a general-purpose graduate course on the subject, it seems to me to have some strange lacunae. It has nothing about core-sets, for instance, and more generally very very little about approximation in geometric algorithms: the only such problem appearing in the table of contents is the Euclidean TSP. It also has similarly scanty coverage of competitive analysis of online algorithms, and absolutely nothing on streaming algorithms.

But if you want a book more specifically about how to bound the approximation ratio of linear and semidefinite programming relaxations to integer programming problems, this may be a worthwhile one to consider, despite the misleadingly general title. And the price is definitely right.

A definitional question

Some of the items taken from the following blog post by David Eppstein (7 Nov 2010):

Approximate book

There's a new book out by Williamson and Shmoys on approximation algorithms, *The Design of Approximation Algorithms*, available electronically for free as a pdf download.

For a book that claims to be a comprehensive reference on approximation algorithms, suitable for a general-purpose graduate course on the subject, it seems to me to have some strange lacunae. It has nothing about core-sets, for instance, and more generally very very little about approximation in geometric algorithms: the only such problem appearing in the table of contents is the Euclidean TSP. It also has similarly scanty coverage of competitive analysis of online algorithms, and absolutely nothing on streaming algorithms.

But if you want a book more specifically about how to bound the approximation ratio of linear and semidefinite programming relaxations to integer programming problems, this may be a worthwhile one to consider, despite the misleadingly general title. And the price is definitely right.

All these types of algorithms do compute things approximately, but is that what the field means by an approximation algorithm? Should these topics get covered in grad courses on approximation algorithms?

FAQ #3

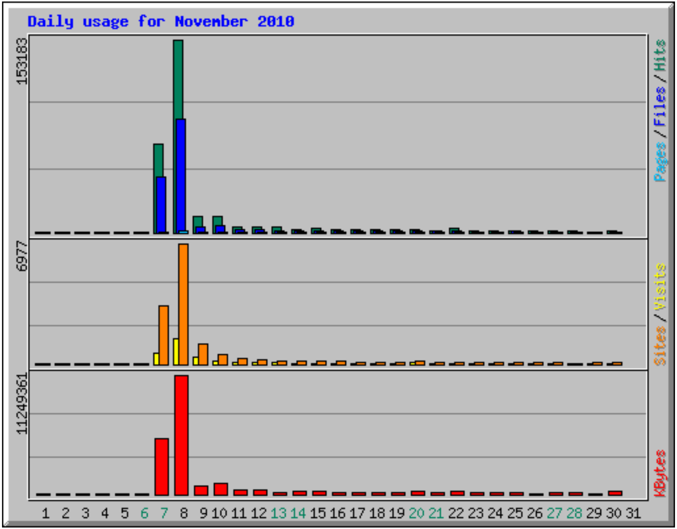
FAQ #3: Are you going to leave the PDF up on the website?

FAQ #3

FAQ #3: Are you going to leave the PDF up on the website?

Answer: Yes, with the agreement of the publisher (Cambridge University Press).

An effect


Daily usage for November 2010
153183
6977
11249361
Pages / Files / Hits
Sites / Visits
KBytes
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

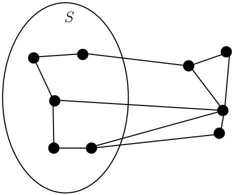
Next: ten open problems from our book.

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

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

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

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

  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

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

  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?

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.

Lightweight approximation algorithms

Lightweight approximation: can we replace more expensive computational primitives with cheaper ones and still get the same guarantees?

SDP $\rightarrow$ SOCP $\rightarrow$ LP $\rightarrow$ Network flow/primal-dual $\rightarrow$ greedy
Ellipsoid $\rightarrow$ polysized LP $\rightarrow \cdots$

Cornell University

Lightweight approximation algorithms

Lightweight approximation: can we replace more expensive computational primitives with cheaper ones and still get the same guarantees?

SDP $\rightarrow$ SOCP $\rightarrow$ LP $\rightarrow$ Network flow/primal-dual $\rightarrow$ greedy
Ellipsoid $\rightarrow$ polysized LP $\rightarrow \cdots$

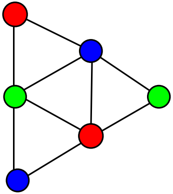
Lots of work already done in this direction (e.g. Poloczek and Schnitger (SODA 2010), randomized $\frac{3}{4}$-approximation algorithm for MAX SAT without solving LP or network flow), but let's do more.

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.



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)

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).

Cornell University

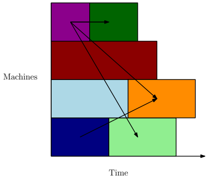
Problem 8: Scheduling related machines with precedence constraints ($Q|prec|C_{\max}$)

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.



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).

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.

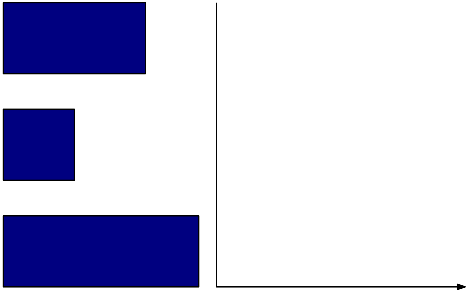
Problem 7: Scheduling unrelated machines ($R||C_{\max}$)

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.



Problem 7: Scheduling unrelated machines

What's known?

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

Cornell University

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).

Cornell University

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 $\mathrm{P} = \mathrm{NP}$ (Lenstra, Shmoys, Tardos 1990).

Cornell University

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 $\mathrm{P} = \mathrm{NP}$ (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

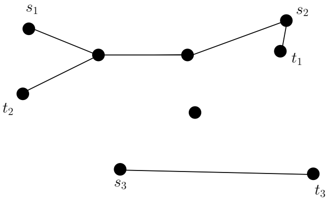
Problem 6: Generalized Steiner tree

Generalized Steiner tree (aka Steiner forest)

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, $\ldots$, 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).



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).

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).
- 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 $\alpha < 2$.

Cornell University

A belief about approximation algorithms

A proof of approximation guarantee α for algorithm A is always a proof about a polytime-computable relaxation R:

$$R \leq OPT \leq A \leq \alpha R.$$

A belief about approximation algorithms

A proof of approximation guarantee α for algorithm A is always a proof about a polytime-computable relaxation R:

$$R \leq OPT \leq A \leq \alpha R.$$

> *The aim of this paper is to look for one or two guiding principles [in analyzing heuristics], and in particular principles relating the analysis of heuristics to such traditional preoccupations of operations researchers as linear programming and branch and bound... We assume problem can be formulated as a linear integer program, and the essential step is to relate the heuristic solution to a dual feasible solution of the given integer problem.*

Wolsey, Heuristic analysis, linear programming and branch and bound *(1980)*

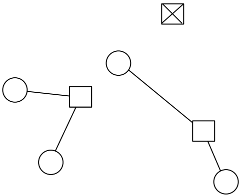
Problem 5: Capacitated facility location

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 $\sigma : D \rightarrow S$ such that $|\sigma^{-1}(i)| \leq U$ for all $i \in S$ that minimizes $\sum_{i \in S} f_i + \sum_{j \in D} c_{\sigma(j),j}$.



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 $i \notin S$, $j \in S$.

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 $i \notin S$, $j \in S$.

- 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)

Cornell University

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?

$$R \leq \text{OPT} \leq A \leq \alpha R.$$

Cornell University

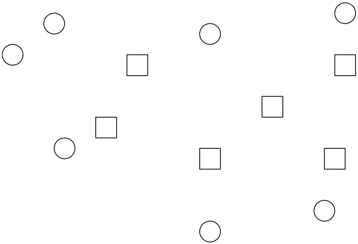
Problem 4: Survivable network design

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).



Cornell University

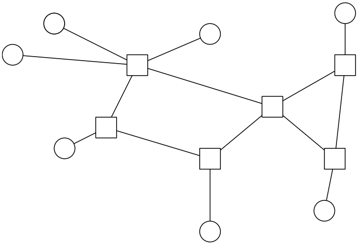
Problem 4: Survivable network design

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:

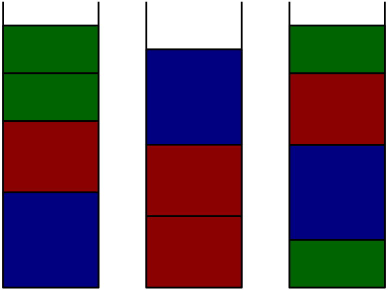
Is there a lightweight 2-approximation algorithm? E.g. a primal-dual algorithm?

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



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)

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.

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{N} x_j \\ \text{subject to} \quad & \sum_{j=1}^{N} a_{ij} x_j \geq b_i, \qquad i = 1, \ldots, m, \\ & x_j \text{ integer}, \qquad j = 1, \ldots, N. \end{aligned}$$

Cornell University

Problem 3: Bin packing

The problem:

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

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

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

but currently no known instances for which

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

Possibly

$$\text{OPT} \leq \lceil LP \rceil + 1.$$

Cornell University

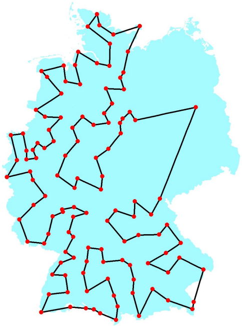
Problems 1 and 2: the traveling salesman problem

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



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 $\mathrm{P} = \mathrm{NP}$ (Papadimitriou, Vempala 2006)

Cornell University

Problems 1 and 2: the traveling salesman problem

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i,j \in V} c_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{j \in V} x_{ij} = \sum_{j \in V} x_{ji} \qquad i \in V, \\
& \sum_{i \in S, j \notin S} x_{ij} \geq 1 \qquad \forall S \subset V \\
& x_{ij} \geq 0 \qquad \forall i, j \in V.
\end{aligned}
$$

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

Problems 1 and 2: the traveling salesman problem

$$\begin{aligned}
\text{minimize} \quad & \sum_{i,j \in V} c_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{j \in V} x_{ij} = \sum_{j \in V} x_{ji} \qquad i \in V, \\
& \sum_{i \in S, j \notin S} x_{ij} \geq 1 \qquad \forall S \subset V \\
& x_{ij} \geq 0 \qquad \forall i, j \in V.
\end{aligned}$$

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.

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)

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)

Graphical case: given graph $G = (V, E)$, c_{ij} is shortest-length path between i and j in G

- Oveis Gharan, Saberi, Singh (December 2010): $\frac{3}{2} - 10^{-12}$
- Mömke, Svensson (April 2011): $\frac{14(\sqrt{2}-1)}{12\sqrt{2}-13} \approx 1.461$
- Mucha (August 2011): $\frac{35}{24} \approx 1.458$

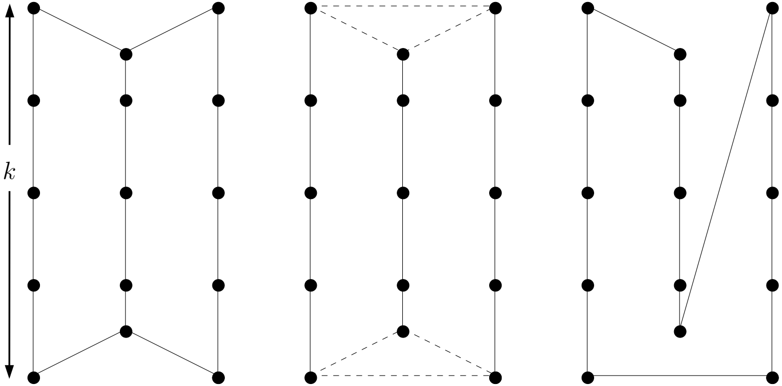
Cornell University

Problems 1 and 2: the traveling salesman problem

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

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

Cornell University


k

Problems 1 and 2: the traveling salesman problem

The problem:

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

A hard, simple case

Suppose LP solution is a fractional 2-matching (all $x_{ij} \in \{0, 1/2, 1\}$).
Can we do better than $3/2$ whenever this is the case?

A hard, simple case

Suppose LP solution is a fractional 2-matching (all $x_{ij} \in \{0, 1/2, 1\}$).
Can we do better than $3/2$ whenever this is the case?

Conjecture (Schalekamp, W, van Zuylen 2011):
Such instances give the worst-case integrality gap.

Problems that didn't make the cut

Problems that didn't make the cut:

- Directed Steiner tree
- LP-based Steiner tree (then Byrka et al. came out)
- Feedback arc set in directed graphs (improve $O(\log n \log\log n)$)
- $P|prec|C_{\max}$ (then Svensson came out)
- Edge coloring multigraphs (+1 result)
- Flow shop, job shop scheduling
- Minimum-cost k-connected subgraph
- Subset feedback vertex set (better than 8)

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 successfully generated interesting algorithmic ideas and mathematical understandings of approximate computation.

Success in computation?

The field has 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

Success in computation?

The field has 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)

Success in computation?

The field has 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 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?

Lightweight approximation algorithms (again)

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

- Jain's algorithm for survivable network design requires solving LP via ellipsoid method
- Goemans-W algorithm for max cut requires solving semidefinite program

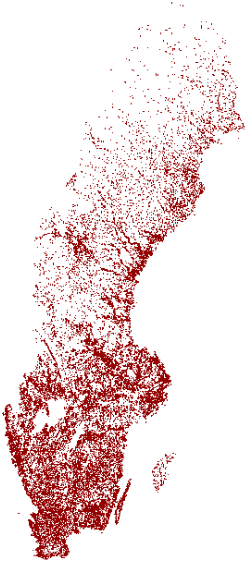
Lightweight approximation algorithms (again)

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

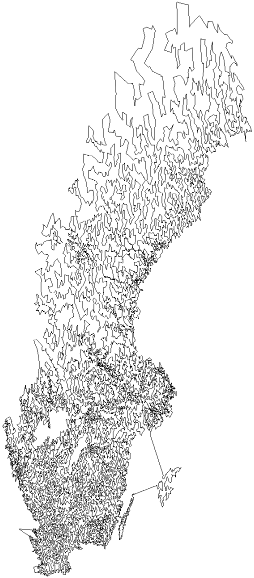
- Jain's algorithm for survivable network design requires solving LP via ellipsoid method
- Goemans-W algorithm for max cut requires solving semidefinite program

Hence lightweight, *implementable*, versions of these algorithms give us a chance to compete with heuristics more often used in practice.

How hard are problems really?



How hard are problems really?



A quest for theory?

Can we explain theoretically why solvers for NP-hard real-world problems work so well on “real-life” instances? Possible directions:

- A more nuanced notion of efficient computation than polynomial time?
- Some empirically justifiable notion of “real-life” instances?

One reason I like the field

Once I start thinking "maybe all the really interesting stuff has been done," someone proves me wrong.

One reason I like the field

Once I start thinking "maybe all the really interesting stuff has been done," someone proves me wrong.

Just in the last year or so

- The Asadpour et al. $O(\log n / \log\log n)$-approximation algorithm for asymmetric traveling salesman problem
- The Byrka et al. 1.39-approximation algorithm for Steiner tree
- All the progress in graphical TSP (mostly using "old" techniques!)

One reason I like the field

Once I start thinking "maybe all the really interesting stuff has been done," someone proves me wrong.

Just in the last year or so

- The Asadpour et al. $O(\log n / \log\log n)$-approximation algorithm for asymmetric traveling salesman problem
- The Byrka et al. 1.39-approximation algorithm for Steiner tree
- All the progress in graphical TSP (mostly using "old" techniques!)

And perhaps your work will be next!

The End

Thanks for your attention.