Discrete Optimization 2010
Lecture 12
TSP, SAT & Outlook

Marc Uetz
University of Twente

m.uetz@utwente.nl

Outline

1. Approximation Algorithms for the TSP
2. Randomization & Derandomization for MAXSAT
3. Outlook on Further Topics
    - Discrete Optimization
    - Online Optimization
    - Algorithmic Game Theory

Outline

1. Approximation Algorithms for the TSP
2. Randomization & Derandomization for MAXSAT
3. Outlook on Further Topics
   - Discrete Optimization
   - Online Optimization
   - Algorithmic Game Theory

The TSP is Really Hard

Symmetric TSP: Given undirected, complete graph $G = (V, E)$, nonnegative integer edge lengths c_e, $e \in E$, find a Hamiltonian cycle (a tour visiting each vertex) of minimum length
(asymmetric TSP: directed graph, so $c_{ij} \neq c_{ji}$ is possible)

Theorem

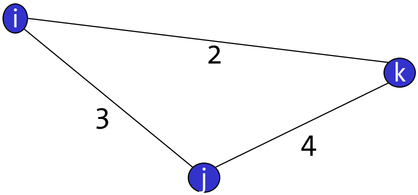
For any constant $\alpha > 1$, there cannot exist an α-approximation algorithm for the (symmetric) TSP, unless $\mathcal{P}$=$\mathcal{NP}$.

Proof: Exercise.

Metric and Euclidean TSP

1. Metric TSP: The distance function c on the edges is required to be a metric. That is, the $\triangle$-inequality holds

$$c_{ik} \leq c_{ij} + c_{jk}$$



2. Euclidean TSP: The nodes are points in $\mathbb{R}^2$ and the metric is given by Euclidean distances

$$c_{ij} = c_{ji} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

The Symmetric TSP: Overview of Results

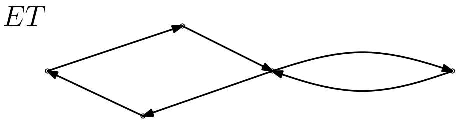
Theorems on TSP

1. General TSP: No α-approximation algorithm unless $\mathcal{P}=\mathcal{NP}$
2. Metric TSP:
   - There is a simple 2-approximation algorithm (Double-Tree Algorithm)
   - There is a simple 3/2-approximation algorithm (Christofides Tree-Matching Algorithm 1976)
3. Euclidean TSP: $\exists$ PTAS, i.e. for any given $\varepsilon > 0$, there is a $(1+\varepsilon)$-approximation algorithm (Arora, Mitchell 1996)

Facts on Euler Tours

Definition

An Euler tour is a closed walk in a graph or multigraph (also parallel edges allowed) that traverses each edge exactly once.

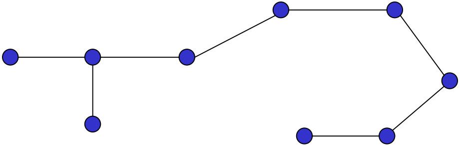


Theorem (Euler 1741) $\longrightarrow$ Bridges of Königsberg

An Euler tour exists if and only if each node has even degree. Moreover, it can be found in $O(n + m)$ time.

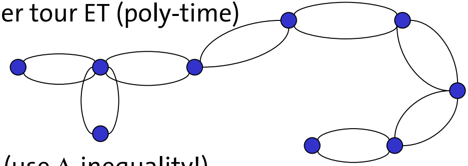
2-approximation: The Double-Tree Algorithm

(1) Compute MST (min. spanning tree)



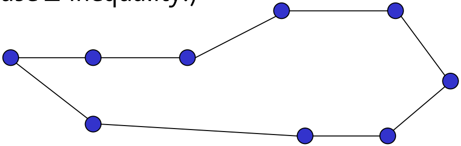
MST $\leq$ TSP

(2) 2 MST = 1 Euler tour ET (poly-time)



ET $\leq$ 2TSP

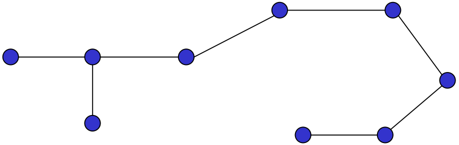
(3) Shortcutting (use Δ-inequality!)



Tour $\leq$ 2TSP

3/2-approximation: Tree-Matching Algorithm

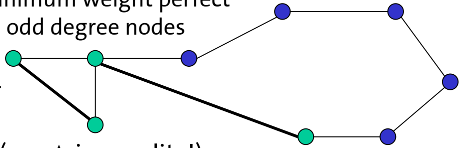
(1) Compute MST (min. spanning tree)



$\text{MST} \leq \text{TSP}$

(2) (a) Compute minimum weight perfect Matching M on odd degree nodes

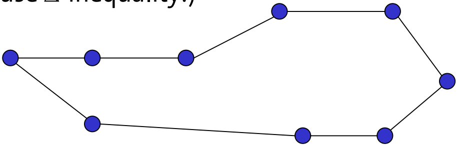
(b) Compute ET



$M \leq \tfrac{1}{2}\,\text{TSP}$

$\text{ET} \leq 3/2\ \text{TSP}$

(3) Shortcutting (use Δ-inequality!)



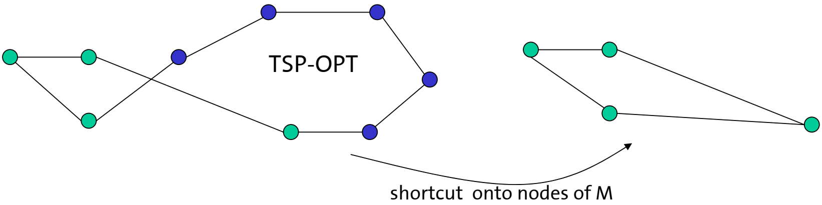
$\text{Tour} \leq 3/2\text{TSP}$

Proofs

- **Claim 1: Eulertour always exists**
  Proof: in both cases there are no odd-degree nodes in the (multi)graph in which we compute an Euler tour
- **Claim 2: Shortcutting works, even in linear time**
  Proof: Walk along the Euler tour, as soon as a node is seen for the second time, store last visited node i, continue along Euler tour, as soon as next unvisited node j is seen, introduce shortcut $\{i, j\}$. This is linear time, $O(\, n\,)$.
- **Claim 3: A perfect matching exists on odd-degree nodes, computable in poly-time**
  Proof: Recall that we assume (w.l.o.g.) a complete graph. And, we must have an even number of odd-degree nodes in the MST, as $2|E| = \sum_v d(v)$, for any graph. We can use Edmonds Matching Algorithm to compute it.

Claim 4: Cost of Matching $c(M) \leq 1/2$ Cost of TSP OPT

Proof:



This result of shortcutting the TSP-OPT tour onto only nodes of M contains exactly two matchings, say M_1 and M_2

So $c(M_1) + c(M_2) \leq$ TSP-OPT

But $c(M) \leq c(M_1), c(M_2)$, as M is min-cost matching, so

$$c(M) \leq \frac{1}{2}\Big(c(M_1) + c(M_2)\Big) \leq \frac{1}{2}\text{TSP-OPT} \qquad \square$$

Outline

1. Approximation Algorithms for the TSP
2. Randomization & Derandomization for MAXSAT
3. Outlook on Further Topics
   - Discrete Optimization
   - Online Optimization
   - Algorithmic Game Theory

'Good' Solutions for Satisfiability

Given a SAT formula in conjunctive normal form,
$F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, on n boolean variables $x_1, \ldots, x_n$

How many of the m clauses are satisfiable at least?

Theorem

There exists a truth assignment fulfilling at least $\frac{1}{2}$ of the clauses

Proof:

1. Let $x_j = \mathit{true}$ with probability $\frac{1}{2}$, for each x_j independently
2. Show $E[\text{number fulfilled clauses}] \geq \frac{1}{2}m$
3. So $\exists\ x \in \{\mathit{true}, \mathit{false}\}^n$ that fulfills $\geq \frac{1}{2}m$ clauses
   (otherwise expectation can't be that large) $\square$

Proof of the Claim

We show even more:

If each clause has at least k literals (variables or their negation), then the expected number of fulfilled clauses is at least

$$\left(1-\left(\frac{1}{2}\right)^k\right)m$$

(as any clause must have at least 1 literal, the claim follows, and e.g. for 3-SAT, that is at least $\frac{7}{8} = 87.5\%$ of the clauses)

Proof:
A clause with $\ell \geq k$ literals is false with probability $\frac{1}{2}^{\ell} \leq \frac{1}{2}^{k}$

Hence, $E[\#\text{true clauses}] = \sum_{i=1}^{m} P(C_i = true) \geq \sum_{i=1}^{m}(1-\frac{1}{2}^{k}) = (1-\frac{1}{2}^{k})m$ $\square$

Randomized Algorithm for Max-SAT

Max-SAT

Given formula F, find a truth assignment x maximizing $\#$ of fulfilled clauses

Max-SAT is (strongly) $\mathcal{NP}$-hard (SAT: $\exists\ x$ fulfilling $\geq m$ clauses?)

What we have:

Randomized Algorithm

- For $(j = 1, \ldots, n)$
  - Let $x_j = \mathit{true}$ with probability $\frac{1}{2}$, $x_j = \mathit{false}$ otherwise

- if randomization $\in O(1)$ time, this is a linear time algorithm
- produces a solution x that is reasonably good in expectation ($\#$ fulfilled clauses $\geq \frac{1}{2}m \geq \frac{1}{2}OPT$, as $OPT \leq m$)

But can we also find such x in poly-time?

Derandomization by Conditional Expectations

Let $X :=$ # of true clauses by algorithm ($X =$ random variable)

$$E[X] = \frac{1}{2}\underbrace{E[X|x_1 = \textit{true}]}_{(1)} + \frac{1}{2}\underbrace{E[X|x_1 = \textit{false}]}_{(2)}$$

Note that (1) and (2) can be computed easily (in time $O(\, nm \,)$), as $E[X] = \sum_i P(C_i = \textit{true})$, for example: $C_i = (x_1 \vee x_2 \vee \bar{x}_7)$

$P(C_i = \textit{true}|x_1 = \textit{true}) = 1$
$P(C_i = \textit{true}|x_1 = \textit{false}) = 1 - P(C_i = \textit{false}|x_1 = \textit{false}) = \frac{3}{4}$

If $(1) \geq (2)$, then $(1) \geq E[X]$, fix $\underline{x_1 = \textit{true}}$ (else, fix $\underline{x_1 = \textit{false}}$)

Assuming $(1) \geq (2)$, next step would be to fix x_2 by the larger of $E[X|x_1 = \textit{true}, x_2 = \textit{true}]$ and $E[X|x_1 = \textit{true}, x_2 = \textit{false}]$, etc.

Keeping x_1 and x_2 fixed, do the same with x_3, etc.... thus get fixed x fulfilling at least $E[X] \geq \frac{1}{2}m$ clauses, in $O(\, n^2 m \,)$ time.

Computing Conditional Expectations: Example

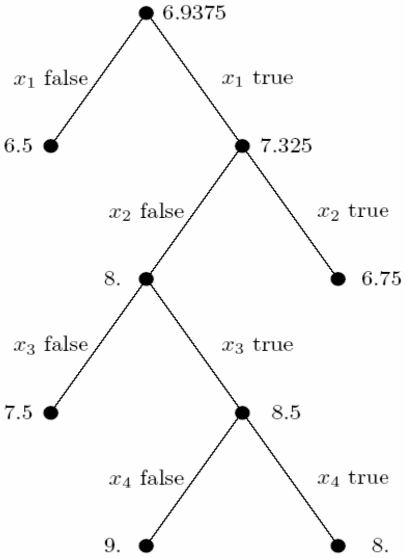
Clauses	nothing fixed	$x_1$false	$x_1$true	$x_1$true $x_2$false	$x_1$true $x_2$true
(x_1)	0.5	0.	1.	1.	1.
$(\overline{x_2})$	0.5	0.5	0.5	1.	0.
(x_3)	0.5	0.5	0.5	0.5	0.5
$(\overline{x_4})$	0.5	0.5	0.5	0.5	0.5
(x_1, x_2)	0.75	0.5	1.	1.	1.
$(\overline{x_3}, \overline{x_4})$	0.75	0.75	0.75	0.75	0.75
$(\overline{x_1}, x_3)$	0.75	1.	0.5	0.5	0.5
$(x_1, \overline{x_2}, x_3)$	0.875	0.75	1.	1.	1.
$(\overline{x_1}, \overline{x_2}, x_4)$	0.875	1.	0.75	1.	0.5
$(\overline{x_1}, x_2, \overline{x_3}, x_4)$	0.9375	1.	0.875	0.75	1.
Expected value	6.9375	6.5	7.325	8.	6.75

...

Derandomization by Conditional Expectations: Example

Clauses	nothing fixed	$x_1$false	$x_1$true	$x_1$true $x_2$false	$x_1$true $x_2$true
(x_1)	0.5	0.	1.	1.	1.
$(\overline{x_2})$	0.5	0.5	0.5	1.	0.
(x_3)	0.5	0.5	0.5	0.5	0.5
$(\overline{x_4})$	0.5	0.5	0.5	0.5	0.5
(x_1, x_2)	0.75	0.5	1.	1.	1.
$(\overline{x_3}, \overline{x_4})$	0.75	0.75	0.75	0.75	0.75
$(\overline{x_1}, x_3)$	0.75	1.	0.5	0.5	0.5
$(x_1, \overline{x_2}, x_3)$	0.875	0.75	1.	1.	1.
$(\overline{x_1}, \overline{x_2}, x_4)$	0.875	1.	0.75	1.	0.5
$(\overline{x_1}, x_2, \overline{x_3}, x_4)$	0.9375	1.	0.875	0.75	1.
Expected value	6.9375	6.5	7.325	8.	6.75

$\ldots$



$1/2$-approximation for MaxCut

MaxCut

Given undirected graph $G = (V, E)$, find a subset $W \subseteq V$ of the nodes of G such that $\delta(W) = \delta(V \setminus W)$ is maximal.

MaxCut is (strongly) $\mathcal{NP}$-complete.

Theorem

There exists a randomized $1/2$-approximation algorithm, which can (easily) be derandomized to yield a $1/2$-approximation algorithm.

Outline

1. Approximation Algorithms for the TSP
2. Randomization & Derandomization for MAXSAT
3. Outlook on Further Topics
   - Discrete Optimization
   - Online Optimization
   - Algorithmic Game Theory

Approximation Algorithms

- LP-based Algorithms with Clever Rounding Schemes
  (for example, Shmoys & Tardos 1993)

- 0.878-Approximation for $\textsc{MaxCut}$ using Semidefinite Programming Relaxation
  (Goemans & Williamson 1994)

- The PTAS for Euclidean TSP
  (Arora 1996)

- The PCP-Theorem
  (alternative characterization of $\mathcal{NP}$)

Integer Linear Programming

- Separation & Optimization are equivalent
  (Grötschel, Lovasz, Schrijver 1981)

- Column Generation Algorithms
  (Dual of adding cuts - namely adding variables)

- Dantzig & Wolfe Decomposition
  (Problem reformulation - then column generation)

Online Optimization

An example, the **Ski Rental problem**: go skiing for n days,

should I rent for \$1 per day (with sunshine)

of buy a pair of skis right away for \$11?

Competitive Analysis

$$\text{Online Algorithm} \;\leq\; \alpha\; \text{Offline Optimum}$$

Buying a pair of skis only after having spent 10\$ for rent, we pay never more than twice the optimum. (2-competitive algorithm)

And, no algorithm can be better than 3/2-competitive (no matter if $\mathcal{P}{=}\mathcal{NP}$ or not).

Algorithmic Game Theory (AGT)

- Assume I have a (poly-time) algorithm that routes all daily traffic on Dutch highways, avoiding congestion
- Great, but nobody will listen: Drivers behave selfishly, only in their own interest

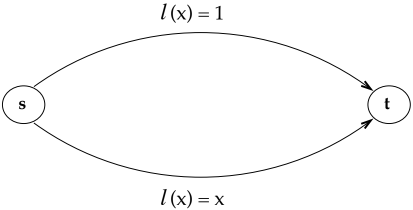
Price of Anarchy is α if

$$\text{Selfish Equilibrium } = \alpha \text{ System Optimum} \qquad (\alpha \geq 1)$$

Mechanism Design: Define incentives (e.g., taxation), such that

$$\text{Selfish Equilibrium } \approx \text{ System Optimum}$$

Example: Price of Anarchy

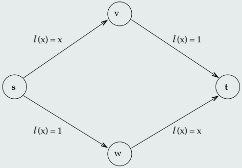


Sending one (splittable) unit of flow

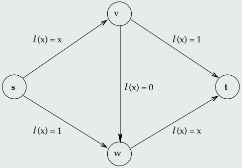
- System optimum: Total latency $= 1/2 + 1/4 = 3/4$
- Nash equilibrium: Total latency $= 1$
- $\Rightarrow$ Price of Anarchy PoA $\geq 4/3$
- Roughgarden/Tardos (2002) show

$$\text{PoA} \;\leq 4/3 \;\forall \text{ networks } \forall \text{ linear functions } \ell$$

Example, cont.: Do they need 42nd street?



(a) Before



(b) After

- Before: Nash $=$ OPT, total latency $= 3/2$
- After: OPT $= 3/2$ (still), but Nash total latency $= 2$

New York Times, December 25, 1990

`What if they closed 42nd street?` by Gina Kolata

Example: Private Information & Mechanism Design

An example

- Single machine, jobs $j \in \{1, \ldots, n\}$ = agents
- Processing times p_j public knowledge
- Weights w_j private information to job j (job j's type)

- Interpretation: w_j = job j's individual cost for waiting

Task

- Schedule jobs, but reimburse for disutility of waiting
- Problem: We do not know w_j's and jobs may lie...

Theorem. If (and only if) $S_j \downarrow$ with $w_j \uparrow$, payments can be defined such that all jobs will tell their true w_j (in equilibrium)

Example: Complexity of Nash

Nash (1951)

A (mixed) Nash equilibrium always exists. Proof uses Brouwers fixed point theorem. Consequence: If we can find Brouwer fixed points (efficiently), we can find Nash equilibria (efficiently).

Question: $\exists$ efficient algorithm to find a Nash equilibrium?

Daskalakis, Goldberg, Papadimitriou (2005)

If we can compute Nash equilibrium (efficiently), we can find Brouwer Fixed points (efficiently).

Consequence: Computing Nash Equilibria is (PPAD) hard.

Thanks for coming

Please fill in the questionnaires, now