Approximation algorithms

Lecture 1 — Montevideo

Maurice QUEYRANNE — August 3, 2009

Background.

- graph theory, combinatorial optimization
- computational complexity : NP-hard problems
- linear programming : LP duality

Overview: why approximation algorithms?

- algorithms that "approximately" solve optimization problems
- many useful optimization problems are <u>difficult</u> to solve exactly
  - often a finite number of possible solutions
  - NP-hard : at least as hard as an NP-complete problem

    recall : an NP-complete problem is a <u>decision problem</u> (YES/NO output)

"at least as hard as" : if there exists a polynomial time algorithm for an NP-hard problem, then there exist a polytime algorithm for all NP-complete decision problems

If $NP \neq P$, then there does not exist a <u>polytime</u> (1) algorithm that solves <u>all instances</u> (2) of an NP-hard optimization problem to <u>exact optimality</u> (3)

How to live with an NP-hard optimization problem?

① develop an exact algorithm, which may <u>not</u> run in polytime for all instances

- branch and bound algorithms - mostly empirical

  example: existing Mixed Integer Programming software

② restrict attention to subsets of instances ("special cases") that can be solved exactly in polytime. - mostly theoretical

③ develop algorithms that find, in polytime for all instances, solutions that are "close to the optimum"

- the objective value of the solution is close to the optimum value
  - empirically: metaheuristics, evaluated on samples of "typical instances"
  - theoretically: mostly the worst-case approximation ratio

Formally, given a discrete optimization problem (P) $\min\{f_I(x) : x \in X_I\}$ where X_I is a finite set, and $f_I : X_I \to \mathbb{R}_+$ for every instance I and a real number $\alpha > 0$

an algorithm A is an <u>α-approximation</u> for (P) if it produces a solution $x^A(I)$ for every instance I, such that

$$f(x^A(I)) \leq \alpha \, OPT(I)$$

where $OPT(I) = \min\{f_I(x) : x \in X_I\}$ for instance I

<u>Remark</u>: an α-approximation for a <u>maximization</u> problem produces a solution with $\alpha f(x^A(I)) \geq OPT(I)$

Therefore $\alpha \geq 1$

$\alpha = 1$ iff the algorithm solves the problem exactly

A dual bound $\beta(I)$ for a minimization problem (P) is any number number such that $\beta(I) \leq OPT(I)$

- most approaches, exact or approximate, to optimization problems are based on finding good dual bounds
  - for example, if algorithm A guarantees that
  $$f_I(x^A(I)) \leq \alpha \, \beta(I) \quad \text{for all instances } I$$
  then A is an α-approximation.

Remark: finding a good dual bound is often itself an optimization problem, a maximization problem $\max\{\beta(I) : \beta \in B(I)\}$, called a dual problem

The Vertex Cover Problem:

an instance is a graph $G = (V(G), E(G)) = (V, E)$

where V is a given finite set of vertices (nodes)

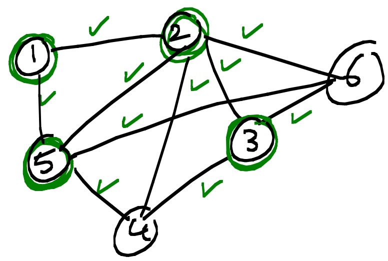
E — — — — — — edges

an edge $e = \{i, j\}$ is a pair of two vertices

(i and j are connected by edge e, are adjacent

i and j are incident to edge e)

a vertex cover (a cover of the edge set with vertices) is a set $C \subseteq V$ of vertices such that every edge $e \in E$ contains at least one vertex in C ("is covered by at least one of its endpoints in C")



$C = \{1, 2, 3, 5\}$ is a vertex cover

The **Vertex Cover Problem** is to find a vertex cover of least cardinality

The Vertex Cover problem is NP-hard

A **greedy algorithm** for Vertex Cover:

start with $C := \emptyset$

while C is not a Vertex Cover (while there exists an edge $e = \{i, j\} \in E$ with $i \notin C$ and $j \notin C$)

choose a vertex v that is incident to the largest number of uncovered edges

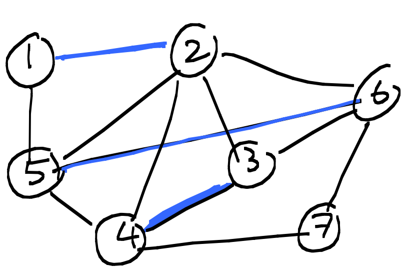
$C := C \cup \{v\}$

return $C^G := C$

- the greedy algorithm returns a Vertex Cover
- how good is C^G ?

Another algorithm (based on a dual problem)

a set $M \subseteq E$ of edges in a graph $G = (V, E)$ is a **matching** if no two edges in M have a common vertex



$M = \{ \{1,2\}, \{5,6\} \}$ is a matching in G

a matching M is **maximal** if there no other matching M' in G that strictly contains it (i.e., we cannot add any edge to M)

$M' = \{ \{1,2\}, \{5,6\}, \{4,3\} \}$ is a **maximal** matching in G.

A simple (polytime) algorithm to find a maximal matching

$M := \emptyset$

while there exists an edge $\{i,j\} \in E$ such that no edge in M is incident to i or to j

$M := M \cup \{\{i,j\}\}$

Maximal-Matching Based Vertex Cover Algorithm: (MMB-VCA)

- construct a maximal matching M in the given graph G
- let $C^M = \bigcup_{\{i,j\} \in M} \{i,j\}$ the set of all endpoints of all edges in M
- return C^M

Example: $C^M = \{1, 2, 5, 6, 3, 4\}$

- the algorithm returns a vertex cover
- How good is C^M?

Theorem: the Maximal-Matching Based Vertex Cover Algorithm returns a Vertex Cover C^M such that $|C^M| \leq 2\,OPT$

proof: let M be a matching then $|M| \leq OPT$
(because any vertex cover must contain at least $|M|$ vertices, one for each edge in M)
then $C^M = 2|M| \leq 2\,OPT$ QED

Remark: The dual problem is Maximum (Cardinality) Matching problem:
given a graph G find a matching M with largest $|M|$
we have the Weak Duality Property:
$\forall$ matching M, $\forall$ vertex cover C in G $\quad |M| \leq |C|$

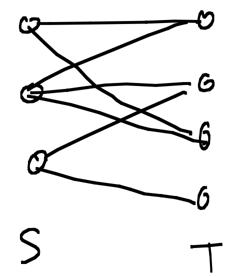
Therefore $\max_{M \text{ matching}} |M| \le \min_{C \text{ vertex cover}} |C|$

the difference $\min|C| - \max|M|$ is the **duality gap**

Remark: **König-Egerváry Theorem:**

This duality gap is zero if and only if the graph G is bipartite

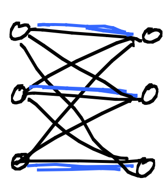
(a **bipartite graph** is such that its vertex set is partitioned into two subsets $V = S \uplus T$ such that every edge $e \in E$ has exactly one endpoint in S and one endpoint in T



The Maximum Matching problem is solvable in poly time

However, in the MMB-VCA, in the worst case, there is no advantage to using a Maximum Cardinality Matching instead of a Maximal one

Tight instances: complete bipartite graph $K_{n,n}$



$K_{3,3}$

a Maximum matching has n edges

so $|C^M| = 2n$

choosing one side, say S, gives a vertex cover C with $|C| = n$

<u>**Inapproximability results:**</u>

There does not exist a polytime algorithm with approximation ratio smaller than

$10\sqrt{5} - 21 \approx 1.36..$ if $P \neq NP$ [Dinur & Safra, STOC 2002]

2 if the "Unique Games Conjecture" is true [Khot & Regev, CCC 2003]

Tentative course schedule

- Today: Metric Steiner tree and Metric TSP
- Linear Programming based approximations: multicut, integer multicommodity flows, facility location
- Other approaches: local search: k-median, submodular maximization
  approximation schemes: Knapsack, Euclidean TSP
  randomization and de-randomization: Max SAT
  semidefinite relaxation: Max Cut, 3-coloring
- Limits to approximation: inapproximability results: Max-Cut, vertex cover

The main paradigm in the study of approximation algorithms

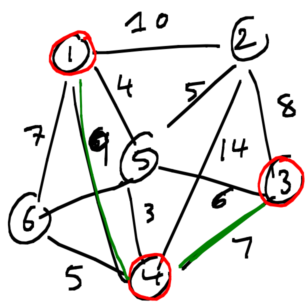
- worst case
- poly time algorithm (but little attempts to reduce running times)
- focus on approximation ratio (focus on finding smallest possible approximation ratio)

"What is the best possible approximation factor one can obtain in poly time for a given problem?"

Steiner tree problem in a graph (a network design problem)

given a graph $G = (V, E)$
edge lengths $\ell(e) \geq 0$
subset $R \subseteq V$ of required vertices

Find a subset $T \subseteq E$ which connects all vertices in R, possibly through some additional vertices, called Steiner vertices, of minimum total length (for every $u, v \in R$ there exists a path from u to v consisting of edges in T)



$T = \{\{1,4\}, \{4,3\}\}$ is a Steiner tree (for $R = \{1,3,4\}$) with total length 16

$T' = \{\{1,5\}, \{4,5\}, \{4,3\}\}$ is another Steiner tree (with Steiner node 5) and total length 14

The Steiner Tree problem in a graph is NP-hard.

Recall: a <u>tree</u> is a connected graph with no cycles

<u>Observation</u>: if $T \subseteq E$ connecting all vertices in R is not a tree, then we can delete some edges, until it becomes a tree still connecting all vertices in R, without increasing its total length

Therefore there exists an optimum solution to the Steiner Tree problem, which is a tree

Let $d(i,j)$ = shortest distance from i to j in graph G (with edge lengths ℓ) – can be found by Dijkstra's shortest path algorithm (because lengths $\ell \geq 0$)

<u>Observation</u>: If a Steiner tree T uses an edge $\{i,j\}$ with length $\ell(i,j) > d(i,j)$ we can improve T by replacing the edge $\{i,j\}$ with a shortest i-j-path

Therefore we can solve a Steiner Tree instance I by considering the complete graph $K = (V, E(K))$ on the same vertex set and with edge lengths $d(i,j)$

The shortest distance satisfy the triangle inequality (Δ-inequality)

$$d(i,j) + d(j,k) \geq d(i,k) \quad \forall i,j,k$$

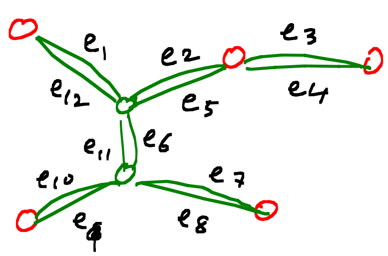
The Metric Steiner Tree Problem is the Steiner Tree problem on complete graph with edge lengths satisfying the Δ-inequality

Consider the complete subgraph induced by R, with lengths $d(i,j)$ we can find a shortest spanning tree T^S (by Kruskal's algorithm, or Prim's algorithm)

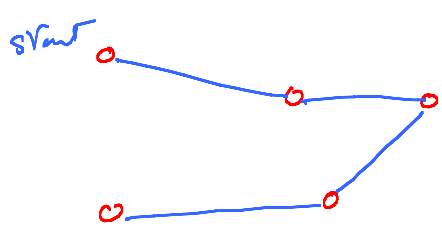
- does not use any Steiner nodes
- how good is it?

Theorem: The Shortest Spanning Tree algorithm is a 2-approximation for the Metric Steiner Tree Problem

proof: let T^* denote an optimum Steiner Tree



T^*



The tree T'
(see next page)

Duplicate every edge in T^* gives T^{**} with total length 2 OPT

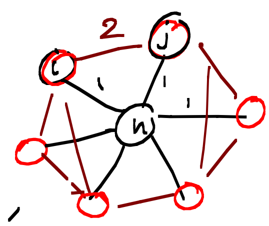
T^{**} is connected and every vertex has even degree, i.e., a Eulerian graph: there exists a Euler cycle, i.e., with a sequence $e_1, \ldots, e_m$ of all edges in E, that forms a cycle C (where each e_i has one vertex in common with e_{i+1} and the other with e_{i-1}, letting $e_0 = e_m$ and $e_{m+1} = e_1$, circularly)

Start at an arbitrary node, follow a Eulerian cycle C of T^{**} taking shortcut to the next unvisited required vertex, until all required vertices are visited

This produces a (Hamiltonian) path on the required vertices, hence a tree T', and: $\text{lgth}(T^S) \leq \text{lgth}(T') \underset{\Delta\text{-ineq}}{\leq} \text{lgth}(T^{**}) = 2\,\text{lgth}(T^*)$ QED

Asymptotically tight instance: complete graph K_n on n nodes

- node n is the "center": $\ell(n,i) = 1$ for all $i \neq n$
  $\ell(i,j) = 2$ for all $i \neq n \neq j$



satisfies Δ-inequality

- required nodes $R = \{1 \dots n-1\}$
  shortest Steiner Tree T^* is the star with center n, $\text{lgth}(T^*) = n-1$

  Shortest Spanning Tree on R: any tree on R, with $\text{lgth} = 2(n-2) > (2-\varepsilon)\,OPT$ for n large enough, for any $\varepsilon > 0$.

Traveling Salesman Problem (TSP)

given a graph $G=(V,E)$ and edge lengths $\ell(e) \geq 0 \ \forall e \in E$
find a Hamiltonian cycle (a cycle that visits every vertex in V exactly once) with least total length

TSP is NP-hard

Actually, the decision problem "Does G contain a Hamiltonian cycle?" is NP-complete

Theorem: If $P \neq NP$ there does not exist a (polynomially computable) function $\alpha(n) \geq 1$ (where n is the number of vertices) and a polytime algorithm with performance ratio smaller than $\alpha(n)$ for the TSP

proof: by contradiction, assume there exists such $\alpha(n)$ and polytime algorithm

Consider any graph G. Define a TSP instance with the same vertex set and in the complete graph, with edge lengths

$$\ell(i,j) = \begin{cases} 1 & \text{if } i,j \in E(G) \\ \alpha(n)\, n & \text{otherwise} \end{cases} \qquad \text{where } n = |V(G)|$$

If G contains a Hamiltonian cycle then $OPT = n$

else $OPT \geq \alpha(n)\, n + 1 > \alpha(n)\, OPT$

Apply the $\alpha(n)$-approximation algorithm to this TSP instance:

- if the length of the solution is $\leq \alpha(n)$ then it must be a Hamiltonian cycle in G
- else, by the approximation guarantee, $OPT \geq \frac{\text{length(solution)}}{\alpha(n)} > n$ therefore G does not contain a Hamiltonian cycle

This would give a polytime algorithm for deciding the Hamiltonian Cycle problem, which would imply $P = NP$, a contradiction. QED