Lecture 1: March 11

Lecturer: Uri Zwick *Scribe: Avi Shoshan*

1.1 Topics to be covered in the course

The course will (tentatively) cover the following topics:

- "Classical" approximation algorithms
  1. Metric TSP
  2. Set Cover
  3. Vertex Cover
  4. Knapsack
  5. Bin Packing
  6. Routing
- Constraint satisfaction problems (CSPs)
  1. The class MAX SNP
  2. The PCP theorem (without proof) and its immediate consequences
  3. Simple approximation algorithms for CSPs
  4. A linear programming based algorithm for MAX SAT
- Semidefinite programming based approximation algorithms
  1. MAX CUT and MAX 2SAT
  2. MAX 3SAT and other problems
  3. Coloring 3-colorable graphs
- Multicuts and multicommodity flow.
- The primal-dual method
- Hardness of Approximation

1.2 "Classical" approximation algorithms

1.2.1 Basic Definitions

Definitions

Given an instance I of an optimization problem Π, we let $\mathrm{sol}(I)$ denote the set of all possible solutions. We assume that $\mathrm{sol}(I) \neq \phi$ for every instance I. Each possible solution $x \in \mathrm{sol}(I)$ has a cost $\text{cost}(x)$ associated with it. The optimization problem is to find $x \in \text{sol}(I)$ for which $\text{cost}(x)$ is maximized (or minimized). We let $\text{OPT}(I)$ denote the cost of the optimal solution.

An *approximation algorithm* for a problem Π is an algorithm A with a polynomial running time that for each input instance I returns a solution $x \in Sol(I)$. We let $A(I)$ denote of the cost of the solution obtained by running algorithm A on the instance I. Hopefully, $A(I)$ is "not far" from $\text{OPT}(I)$.

An approximation algorithm A for Π has an *absolute performance guarantee* of c if for every instance I of Π we have $|\text{OPT}(I) - A(I)| \leq c$.

An approximation algorithm A for Π has a *relative performance guarantee* of α if for every instance I of Π we have $A(I)/\text{OPT}(I) \geq \alpha$, for a maximization problem, or $A(I)/\text{OPT}(I) \leq \alpha$, for a minimization problem. (For maximization problems we must have $\alpha \leq 1$. For minimization problems we must have $\alpha \geq 1$.) We occasionally use the term *performance ratio* instead of relative performance guarantee.

A maximization problem Π has a *PTAS - Polynomial Time Approximation Scheme* if for every $\epsilon \geq 0$ there is an approximation algorithm for Π with a relative performance guarantee of $1 - \epsilon$. For every fixed $\epsilon > 0$, the running time of the corresponding approximation algorithm should be polynomial in n. If the running time is polynomial in n and in $1/\epsilon$ we get a *FPTAS - a Fully Polynomial Time Approximation Scheme*. A similar definition applies to minimization problems.

Examples

1. **Edge Coloring of graphs:** Given a graph $G = (V, E)$ find a coloring of the edges using a minimal number of colors such that no two edges with the same color are adjacent on the same vertex. The minimal number of colors needed to color the edges of G in this way is called the edge coloring number of G and it is denoted by $\chi'(G)$.

   The edge coloring problem is NP-hard but we can easily obtain an approximation algorithm for it with an absoloute performance guarantee of 1.

   Theorem 1.1 (Vizing) *For every graph $G = (V, E)$ we have $\Delta \leq \chi'(G) \leq \Delta + 1$ where Δ is the maximal degree of G.*

   A coloring of the edges of G using at most $\Delta + 1$ colors can be found in polynomial time. Thus we get the absolute guarantee with $c = 1$.

2. **Vertex Coloring of planar graphs:** Given a planar graph $G = (V, E)$, find a coloring of the vertices using a minimal number of colors such that no two vertices connected by an edge are assigned the same color.

   The 4-color theorem states that every planar graph is 4-colorable. Furthemore, such a coloring can be found in polynomial time. On the other hand, deciding whether a planar graph is 3-colorable is NP-complete.

   Since we can easily decide in linear time whether a graph is 2-colorable we can easily obtain an approximation algorithm with an absolute performance guarantee of 1. Check whether G is 1-colorable or 2-colorable. If so, find an appropriate coloring and output it. Otherwise, find a coloring of G using 4 colors and output it.

3. **Finding a spanning tree with minimal maximum degree:** Given a connected graph $G = (V, E)$, find a spanning forest of G whose maximal degree is the smallest possible. This problem is NP-hard since G has a spanning forest with maximal degree 2 if and only if G contains a hamiltonian path. There is an approximation algorithm for the problem with absolute performance gurantee of 1. The algorithm was obtained by Fürer and Raghavachari [FR94] (see also also [Rag96]). We will not describe it here.

4. **The Traveling Salesman Problem (TSP):** Given a complete graph $G = (V, E)$ with a nonnegative weight function $w : E \to R^+$ on its edges. Find a Hamiltonian cycle with minimal total weight. This problem is NP-hard, even if all the weights are finite. Furthermore, no relative performance guarantee, and certainly no absolute performance gurantee may be obtained for the problem, unless $P \neq NP$. The reason is that given an approximation algorithm that achieves with a performance ratio of $\alpha < \infty$ for the problem we can solve the Hamiltonian cycle problem as follows. Given a graph $G = (V, E)$, give all the edges of E a weight of 1. Add edges between all pairs of vertices not connected by edges in G and give these edges a weight of αn. The graph $G = (V, E)$ has a Hamiltonian cycle if and only if the cost of the optimal traveling salesman is exactly n. Furthermore, if G does not have a Hamiltonian cycle then the cost of the optimal hamiltonian cycle is more than αn. Given an algorithm with a performance ration of α we will be able to distinguish between these two cases.

5. **Metric TSP:** This is a special case of the TSP problem considered above. Here we assume that the edge weights satisfy the triangle inequality. We will see next that we can obtain a relative performance guarantee of $3/2$ for this problem.

1.2.2 Approximation algorithms for metric TSP

Definition *Metric TSP: Given a complete undirected graph $G = (V, E)$ with a weight (or length) function $w : E \to R^+$ that satisfies the triangle inequality, find a Hamiltonian cycle of minimum weight. A weight function $w : E \to R^+$ satisfies the triangle inequality if for every three vertices $i, j, k \in V$ we have $w(i, j) \leq w(i, k) + w(k, j)$. A Hamiltonian cycle is a cycle that passes though each vertex exactly once. The weight of a cycle is the sum of the weights of the edges on it.*

It is not difficult to see that the metric TSP problem is still NP-hard. We first describe a simple 2-approximation algorithm for metric TSP, i.e., an approximation algorithm that with a relative performance guarantee of 2. We then show to improve the algorithm and obtain a 3/2-approximation algorithm for the problem (Christofides [Chr76]). This is currently the best approximation algorithm for the problem.

A 2-approximation algorithm for metric TSP

The algorithm is :

1. Find a minimum spanning tree T of G.

2. Duplicate each edge in T and find an Euler tour.

3. Convert the Euler circuit into a Hamiltonian cycle using "shortcuts": the vertices of the graph appear in the Hamiltonian cycle in their order of first appearance along the Euler tour, starting at an arbitrary point of the tour.

It is easy to see that all these operations can be performed in polynomial, in fact almost linear, time.

Theorem 1.2 *The algorithm given above has a relative performace guarantee of* 2.

Proof: Let MST(I) be the weight of a minimum spanning tree T of G. Let TSP(I) be the weight of the optimal Hamiltonian cycle. It is clear that $\mathrm{MST}(I) \le \mathrm{TSP}(I)$ as a spanning tree of G may be obtained by removing any edge from any Hamiltonian cycle.

The weight of the Euler tour constructed is clearly $2\mathrm{MST}(I)$, as each original edge of the minimum spanning tree T is used twice. Each edge on the Hamiltonian cycle constructed replaces a path on the Euler tour. The triangle inequality implies that the length of this edge is not larger than the length of the path it replaces. Thus, the total of the Hamiltonian path constructed is at most $2\mathrm{MST}(I)$, which is at most $2\mathrm{TSP}(I)$, as required. ■

Exercise: Show that the above analysis is tight. In other words, there are instances in which the weight of the Hamiltonian path produced is about twice the optimal.

A 3/2-approximation algorithm for metric TSP

In the 2-approximation algorithm described above we *duplicated* each edge of the spanning tree T in order to make it Eulerian. There is, however, a cheaper way of converting the spanning tree T into an Eulerian graph. Instead of duplicating all the edges, we can find a *matching* of the odd degree vertices of T. The edges of the matching need not be edges of the tree T. This idea was used by Christofides [Chr76] who suggested the following algorithm:

1. Find a minimum spanning tree T of G.
2. Find a minimum weight matching that matches all vertices of odd degree in T. (There must be an even number of odd degree vertices in T.)
3. By adding the egdes of the matching to T we get an Eulerian graph (all the degrees are even). Find an Euler tour of this graph.
4. Convert the Euler circuit into a Hamiltonian cycle using "shortcuts", as in the previous algorithm.

The problem of finding a minimum weight matching can be solved in polynomial time (see Lawler [Law76]).

Theorem 1.3 *The algorithm of Christofides gives a relative performance guarantee of* 3/2.

Proof: As in the previous proof, we get here $A(I) \le \mathrm{MST}(I) + \mathrm{MM}(I)$, where MM($I$) is the weight of the minimum weight matching of the odd degree vertices of T. The only thing left is to show is that $\mathrm{MM}(I) \le \mathrm{TSP}(I)/2$.

Consider a optimal Hamiltonian cycle of G. Let A be the set of odd degree vertices in T. By shortcutting, we can obtain a simple cycle that passes through all the vertices of A and whose weight is at most $\mathrm{TSP}(I)$. This cycle can be broken into two perfect matchings of A. Thus, the weight of the minimum weight matching is at most $\mathrm{TSP}(I)$, as required.

■

Exercise: Show that there are instances on which the performance ratio achieved by Christofides' algorithm approaches $3/2$.

As mentioned, $3/2$ is currently the best performance guarantee available for metric TSP. A ratio of $6/5$ was obtained by Papadimitriou and Yannakakis [PY93] for the special case in which all weights are either 1 or 2.

1.2.3 Set Cover

Definition *Set Cover: Given a finite set X of size n and m subsets $S_1, S_2, \ldots, S_m$ of X such that $\cup_{i=1}^m S_i = X$, find a set $C \subseteq \{1, 2, \ldots, m\}$ of minimal size such that $\cup_{i \in C} S_i = X$.*

This problem is a well known NP-complete problem, the vertex cover problem, to be discussed next, is a special case of it.

A very natural heuristic for solving the set cover problem is the *greedy* heuristic: choose the sets of the cover, i.e., the sets whose indices belong to C, one by one, always choosing the set that covers the largest number of uncovered elements, until all the elements are covered. More precisely, if C is the set of indices of the sets already chosen, the next set to be added to the cover is a set S_j such that $|(X - (\cup_{i \in C} S_i)) \cap S_j|$ is as large as possible.

We would now show that the greedy algorithm has a performance guarantee of about $\ln n$, or more precisely of $H(\Delta) = \sum_{i=1}^{\Delta} \frac{1}{i}$, where Δ is the size of the largest set among $S_1, S_2, \ldots, S_m$. The numbers $H(n) = \sum_{i=1}^{n} \frac{1}{n}$ are called *harmonic numbers* and they satisfy $\ln n \leq H(n) \leq \ln n + 1$. Note that the performance guarantee this time is not a constant but rather a function of n, or of Δ.

Theorem 1.4 (Chvátal [Chv79], Lovasz [Lov75]) *The greedy algorithm for set cover achieves a performance ratio of $H(\Delta)$, where Δ is the size of the largest set among $S_1, S_2, \ldots, S_m$.*

Proof: To prove the theorem we use a clever accounting scheme that transfers the cost of each set added to the cover to the elements covered for the first time by this new set.

Let C^* be an optimal set cover. Let C be the cover produced by the greedy algorithm. Let $T_1, T_2, \ldots, T_{|C|}$ be the subsets chosen by the greedy algorithm. If T_i is the first subset to cover an element x, we assign x with a cost of $c_x = 1/(|T_i - (T_1 \cup T_2 \cup \ldots \cup T_{i-1})|)$. In other words, the cost of the set T_i, which is one, is evenly split among the elements that are covered for the first time by T_i.

We now need the following lemma:

Lemma 1.5 *For every set S among $S_1, S_2, \ldots, S_m$ we have $\sum_{x \in S} c_x \leq H(|S|)$.*

Proof: Let u_i be the number of items in S not covered by $T_1 \cup T_2 \cup \ldots \cup T_i$. That is $u_i = |S - (T_1 \cup T_2 \cup \ldots \cup T_i)|$. Note that $u_0 = |S|$. Let k be the first index for which $u_k = 0$, that is the first k for which $S \subseteq T_1 \cup T_2 \cup \ldots \cup T_k$. As the set T_i is chosen greedily, we have $|T_i - (T_1 \cup T_2 \cup \ldots \cup T_{i-1})| \geq |S - (T_1 \cup T_2 \cup \ldots \cup T_{i-1})| = u_{i-1}$.

The number of elements of S covered for the first time by T_i is $u_{i-1} - u_i$. Each such element carries a cost of $1/|T_i - (T_1 \cup T_2 \cup \ldots \cup T_{i-1})| \leq 1/u_{i-1}$. Thus,

$$\sum_{x \in S} c_x \leq \sum_{i=1}^{k} \frac{u_{i-1} - u_i}{u_{i-1}} .$$

Now, if $a \leq b$ then $H(b) - H(a) = \sum_{i=a+1}^{b} \frac{1}{i} \geq \frac{b-a}{b}$. (We define $H(0) = 0$.) As $u_i \leq u_{i-1}$, we get that $(u_{i-1} - u_i)/u_{i-1} \leq H(u_{i-1}) - H(u_i)$. Thus,

$$\sum_{x \in S} c_x \leq \sum_{i=1}^{k} \frac{u_{i-1} - u_i}{u_{i-1}} \leq \sum_{i=1}^{k} (H(u_{i-1}) - H(u_i)) = H(u_0) - H(u_k) = H(|S|) .$$

■

We can now complete the proof of the theorem:

$$|C| = \sum_{x \in X} c_x \leq \sum_{i \in C^*} \sum_{x \in S_i} c_x \leq |C^*| \cdot H(\Delta) ,$$

as required. ■

We will see in the next lecture that this analysis is tight. Feige [Fei96] showed recently that no polynomial time approximation algorithm may achieve a performence ratio smaller than $\ln n$ for the set cover problem, unless P=DTIME($n^{O(\log \log n)}$).

Bibliography

[Chr76] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[Chv79] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[Fei96] U. Feige. A threshold of $\ln n$ for approximating set cover (preliminary version). In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing, Philadelphia, Pennsylvania*, pages 314–318, 1996.

[FR94] M. Fürer and B. Raghavachari. Approximating the minimum-degree Steiner tree to within one of optimal. *Journal of Algorithms*, 17:409–423, 1994.

[Law76] E.L. Lawler. *Combinatorial optimization: networks and matroids.* Holt, Rinehart and Winston, 1976.

[Lov75] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[PY93] C.H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18:1–12, 1993.

[Rag96] B. Raghavachari. Algorithms for finding low degree structures. In D.S. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, chapter 7, pages 266–293. PWS Publishing Company, 1996.