,3

CMPUT 675: Approximation Algorithms **Fall 2013**

Lecture 2,3 (Sept 10 and 12, 2013): TSP, Set Cover, Introduction to LP

Lecturer: Mohammad R. Salavatipour *Scribe: Tim Yee, Bing Xu and older notes*

2.1 Traveling Salesman Problem (TSP)

This is a very well-known NP-hard problem. There are at least three books written on this problem.

Definition 1 Traveling Salesman Problem (TSP): *Given a complete graph $G(V, E)$ on n vertices with edge cost $c : E \longrightarrow Q^+$, find a minimum cost cycle visiting every vertex exactly once, i.e. a minimum cost Hamiltonian cycle.*

Finding a Hamiltonian cycle in a graph is NP-hard. Using this fact, we show that TSP cannot have an approximation algorithm in the general case.

Theorem 1 *For any polynomially computable function $f(\cdot)$, TSP does not have an $f(n)$-approximation algorithm unless P=NP.*

Proof. Let G be the instance of Hamiltonian cycle problem and construct G' on the same vertex set in the following way:

- If $e \in G$, then the cost of e in G' is 1.
- If $e \notin G$, the cost of e in G' is $f(n) \cdot (n + 1)$, where n is the number of vertices in G.

If G has a Hamiltonian cycle then the TSP tour in G' has cost n and an $f(n)$-approximation returns a solution of cost at most $f(n) \cdot n$). If G does not have a Hamiltonian cycle then every TSP tour in G' must use at least one of those heavy edges and therefore has cost larger than $f(n) \cdot (n + 1)$. Thus, if we have an algorithm A for TSP with factor $f(n)$, we can decide whether G has a Hamiltonian cycle, which is NP-hard. ■

2.1.1 Approximation of metric TSP

So let's focus on the metric instances of TSP. A distance function $d : X \times X \to \mathbb{R}^+$ defined over X is a metric if:

- $d(u, u) = 0$ for all $u \in X$.
- $d(u, v) = d(v, u)$ for all $u, v \in X$
- $d(u, v) \leq d(u, w) + d(w, v)$ for all $u, v, w \in X$.

The third condition is called triangle inequality. Since TSP is not approximable in general we focus on weighted graphs where the cost function is a metric. This means, the input graph is a complete graph and the edge weights satisfy triangle inequality. For example, one can take the metric completion of the input graph which is the graph whose edge weights are the shortest path distances of the original graph. It is easy to see that the shortest path distances define a metric.

From now on, when we talk about TSP we will be assuming metric graphs. This assumption implies a complete graph obeying the triangle inequality. The first algorithm we present is a simple 2-approximation that uses minimum spanning tree. Note that if T is a MST then $c(T) \leq OPT_{TSP}$ since deleting any edge from a TSP tour results in a tree.

2.1.1.1 Algorithm 1

1. Find a MST T in the input graph
2. Duplicate all edges and call this new graph T'. (Note: Cost(T') = 2*Cost(T))
3. Find an Eulerian walk (a path that uses all edges in a graph). Let's call this walk W.
4. Find a path P by following W, but skipping previously visited vertices by taking the direct route to the next unvisited vertex along W. The resulting path P is our approximation of TSP.

Note: $Cost(T) \leq OPT_{TSP}$ and $Cost(P) \leq Cost(T')$ by the metric property. So we can see that we have a 2-approximation of TSP.

2.1.1.2 Algorithm 2

We next see how we can improve the approximation ratio of the previous algorithm. The factor 2 loss in the approximation ratio came from doubling the edges of the MST found. The reason we doubled the edges was to find an Eulerian graph (i.e. a graph in which all degrees are even). The improvement comes by adding fewer edges to T. Let O be the set of odd degree nodes of T. Note that $|O|$ is even. We find a minimum cost matching over O, call it M and add this to T. It is easy to see that $T + M$ is now an Eulerian graph as all the degrees are even. We will argue that the cost of the matching M added will be at most $OPT_{TSP}/2$ and this will imply a 3/2-approximation.

1. Find T a MST on the graph; let O be the set of odd degree nodes of T.
2. Find a minimum cost matching M over O.
3. Create a new graph $M + T$. All degrees in this graph are even, so we can find an Eulerian walk on it.
4. Repeat steps 3 & 4 from Algorithm 1.

Therefore, we only need to show that $cost(M) \leq OPT_{TSP}/2$, since the cost of Euler tour found in Step 3 is exactly $cost(T) + cost(M)$. The following lemma completes the proof of this algorithm.

Lemma 1 *Let $V' \subseteq V$ s.t $|V'|$ is even and let M be a minimum cost perfect matching on V'. Then the $cost(M) \leq OPT_{TSP}/2$.*

Proof. Consider any optimal TSP tour τ of G and let τ' be the tour obtained from τ by shortcutting on the vertices of $V - V'$, i.e. skip the vertices of $V - V'$. So τ' is a tour on V' only and $cost(\tau') \leq cost(\tau)$ because we have a metric instance. Now, since $|V'|$ is even, τ' can be decomposed to two perfect matchings by choosing the even edges or the odd edges on the tour. Since the cost of a minimum perfect matching on V' is smaller that each of these: $cost(M) \leq \frac{1}{2}cost(\tau') \leq OPT_{TSP}/2$. ■

From the lemma, we can obtain the guarantee ratio for the algorithm to be $\frac{3}{2}$.

This algorithm, called Christofides, is the best known approximation algorithm for TSP for the past 36 years.

Major open problem: Obtain a better approximation algorithm for metric TSP or prove that there is no such algorithm, under some reasonable complexity assumption.

2.2 Set Cover Problem

Now we turn our attention to the Set Cover problem, which is (perhaps) the most central problem in the study of approximation algorithms. There are different algorithms for this problem. In this course we will see at least 4 different approximation algorithms for this using different methods.

Set Cover:

- Input:
  - A set of n elements $U = \{e_1, \ldots, e_n\}$, called the Universe.
  - A set $S = \{S_1, \ldots, S_m\}$ of m subsets of U such that each $e \in U$ is in some $S_i \in S$
  - A cost function $c : S \to \mathbb{Q}^+$
- Goal: Find a minimum cost subset S' of S such that each $e \in U$ is in some $S_i \in S'$.

Note that vertex cover is a special case of set cover where U is the set of all edges and each vertex v is a subset in S which contains all edges incident to v. In this case, each element is in exactly two subsets in S. We present a greedy approximation algorithm for Set Cover. This is probably the most natural greedy algorithm for this problem. The idea is, at each iteration pick a set where the ratio of the cost of the set divided by the number of new elements it covers is minimized. This general idea of "covering" elements iteratively by finding good partial solutions has been used in many other problems. The analysis of set-cover (we present here) can typically be extended to those other covering algorithms that behave similarly.

Definition 2 *Given a subset C of U, define the cost effectiveness of set $S_i \in S$ as $\frac{c(S_i)}{|S_i - C|}$. If $S_i \subseteq C$ then say the cost effectiveness is $+\infty$.*

Algorithm SC1

$C \leftarrow \emptyset$
$S' \leftarrow \emptyset$
while $C \neq U$ **do**
  select $S_i \in S$ with minimum cost effectiveness $\alpha = \frac{c(S_i)}{|S_i - C|}$ with respect to C
  for each $e \in S_i$, define $price(e)$ as α
  $S' \leftarrow S' \cup \{S_i\}$
  $C \leftarrow C \cup S_i$
return S'

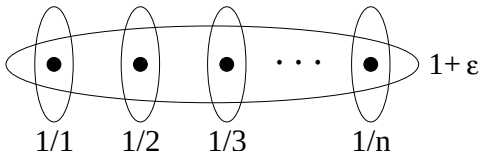

Figure 2.1: A tight example for $SC1$.

Obviously, all elements are eventually covered by S' since the algorithm terminates only when $C = U$. Note that the final cost of set S' is $\sum_{e \in U} price(e)$ since, for each $S_i \in S'$, the cost of S_i is distributed among all elements in S_i that were covered for the first time when S_i was picked.

Lemma 2 *Algorithm $SC1$ is an $\ln n$-approximation algorithm; more precisely it has ratio at most H_n, where H_n is the n'th harmonic number.*

Proof. Let $T_{OPT} \subseteq S$ be a set cover with minimum cost OPT. Order the elements of U by the time they were covered by algorithm SC1 (breaking ties arbitrarily) as $e_1, e_2, \ldots, e_n$.

Consider the time just before e_k is covered. The remaining at least $n - k + 1$ elements can be covered at a price of no more than OPT by adding the currently unselected sets of T_{OPT} to S'. In other words, each element can be covered at a price of no more than $\frac{OPT}{n-k+1}$ on average.

We claim that there must be a set with cost effectiveness at most $\frac{OPT}{n-k+1}$. If this were not true, then the cost of covering the remaining uncovered elements would be strictly greater than $(n-k+1) \cdot \frac{OPT}{n-k+1} = OPT$ which contradicts the fact that the remaining elements can be covered at a cost of at most OPT by selecting T_{OPT}. Thus, $price(e_k) \leq \frac{OPT}{n-k+1}$ which yields

$$\sum_{k=1}^{n} price(e) \leq \sum_{k=1}^{n} \frac{OPT}{n-k+1} = OPT \cdot \sum_{k=1}^{n} \frac{1}{k} = OPT \cdot H_n$$

where H_n is the $n'th$ harmonic number. By comparison with $\int \frac{dx}{x}$ we see that $\ln n \leq H_n \leq \ln n + 1$. Therefore $SC1$ is an $O(\log n)$ approximation algorithm. ■

Through similar analysis, we can show that $SC1$ is an $O(\log k)$-approximation where $k = \max |S_i|$. Note that this proves the ratio of $O(\log \Delta)$ for the greedy vertex cover algorithm where Δ is the size of maximum degree of nodes. The analysis of $SC1$ is also tight. For any $\epsilon > 0$ being a small constant, consider the following instance of set cover (illustrated in Figure 2.2):

- $U = \{e_1, \ldots, e_n\}$
- $S = \{S_0, S_1, \ldots, S_n\}$
- $c(S_0) = 1 + \epsilon$ and $c(S_i) = \frac{1}{i}$ for all $1 \leq i \leq n$.

The optimum solution is S_0 with a cost of $1 + \epsilon$ while $SC1$ returns the solution $\{S_1, \ldots, S_n\}$ with a cost of $H_n = OPT \cdot \frac{H_n}{1+\epsilon}$. Since this holds for any small constant $\epsilon > 0$, the analysis is tight (even up to the constant).

Interestingly, the above algorithm is essentially the best possible for set cover.

Theorem 2 (Lund and Yannakakis (92), Feige (96), Raz and Safra (97), Sudan (97))

- *Unless* P = NP*, there is no $c \ln n$ approximation algorithm for set cover for some constant $0 < c < 1$.*
- *Unless* NP $\subseteq$ DTIME$(n^{O(\log \log n)})$*, there is no $(1 - \epsilon) \ln n$ approximation for set cover for all $\epsilon > 0$.*

2.3 Linear Programming

Definition 3 Linear Programming *(or LP) is the problem of optimizing a linear function of variables subject to some linear constraints.*

An LP with n variables $x_1, \ldots, x_n$ takes the form

$$\begin{aligned} \text{minimize :} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{subject to :} \quad & \sum_{i=1}^{n} a_{ji} x_i \quad \geq \quad b_j, \quad \forall 1 \leq j \leq m \\ & x_i \quad \geq \quad 0, \quad \forall 1 \leq i \leq n \end{aligned}$$

for some constants $c_i, a_{ij}, b_j, 1 \leq i \leq n, 1 \leq j \leq m$. Here, the first expression is called the *objective function*. In matrix form, an LP looks like

$$\begin{aligned} \text{minimize :} \quad & x \cdot c \\ \text{subject to :} \quad & x \cdot A \quad \geq \quad b, \quad \forall 1 \leq j \leq m \\ & x \quad \geq \quad 0, \quad \forall 1 \leq i \leq n \end{aligned}$$

where x is an n-dimensional vector of variables. In such a case, A is called the *constraint matrix*. An assignment of values to the variables in x is said to be *feasible* if all the constraints are satisfied.

Consider the following LP with two variables and four constraints.

$$\begin{aligned} \text{minimize :} \quad & x_2 \\ \text{subject to :} \quad & 3x_1 - x_2 \quad \geq \quad 0 \\ & x_1 + x + 1 \quad \geq \quad 6 \\ & -x_1 + 2x_2 \quad \geq \quad 0 \\ & x_1 \quad \geq \quad 2 \\ & x_1, x_2 \quad \geq \quad 0 \end{aligned}$$

In this example, the assignment $x_1 = 2$ and $x_2 = 4$ is feasible and has objective function value 4. See figure 2.3 for a visualization of the LP as well as how the following terminology applies to the example.

Each LP falls into exactly one of the following three categories:

- There is a feasible $x \in \mathbb{R}^n$ such that for all feasible $x' \in \mathbb{R}^n$, the objective function value under assignment x' is at least the objective function value under x. We say the LP has a *finite optimum*.
- Every $x \in \mathbb{R}^n$ is not feasible. In this case, we say the LP is *infeasible*.
- For every $z \in \mathbb{R}$, there exists a feasible $x \in \mathbb{R}^n$ such that the objective function value under assignment z exceeds the objective function value under assignment x. Here, the LP is said to be *unbounded*.

Each constraint defines a half-space of $\mathbb{R}^n$. The feasible region is the intersection of all half-spaces defined by the constraints of the LP. Consider an LP with a non-empty feasible region. If this feasible region is bounded then the LP has a finite optimum which also implies that the feasible region is infinite if the LP is unbounded. Finally, if the LP is infeasible, then the feasible region is empty.

Every solution to an LP is a vector in $\mathbb{R}^n$. Every constraint corresponds to a half-space. The set of feasible solutions to an LP is*convex*. That is, for all feasible $x, y \in \mathbb{R}^n$ we have $\alpha x + (1 - \alpha)y$ being feasible for each

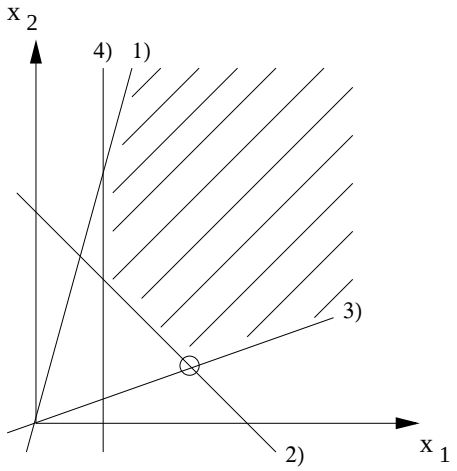

Figure 2.2: A visualization of the example LP. The four lines correspond to the four constraints and are numbered in the order they appear in the example. The feasible region (shaded above) is infinite while the LP has a finite optimum. The optimum value $x_2 = 2$ occurs at the basic feasible solution (circled above) defined by the second and third constraints.

$0 \leq \alpha \leq 1$. Intuitively, all points along the straight line between two feasible points are also feasible. A feasible region for an LP is called a *polyhedron*. If the LP is bounded, then the feasible region is called a *polytope*. A feasible point $x \in \mathbb{R}^n$ is called a *basic feasible solution* or a *vertex* of the polytope if some n constraints are satisfied with equality under assignment x. Geometrically, a basic feasible solution is found where the boundaries of n distinct half-spaces intersect. By definition, a vertex solution is not a convex combination of two or more distinct feasible solution. It can be shown that if an LP has a finite optimum, then some optimum occurs at a basic feasible solution.

A linear program can be solved in polynomial time using, for example, the ellipsoid method or an interior point method. First we note that the difficulty of linear programming is not in finding an optimum solution, it is rather finding a feasible solution. In other words, it is an easy exercise to show that having access to a procedure that finds a feasible solution one can optimize over the feasible region. So the main task is to find a feasible soltuion. The ellipsoid method works in the following way. We start with a large ellipsoid that is containing the convex set P (feasible region); the question of how to start with such a ellipsoid is something we skip at this point. Then we check if the center of this ellipsoid is inside P or not. If it is then we have found a feasible solution and we are done. Otherwise, the algorithm finds a linear inequality that is satisfied by all the points in the feasible region and is not satisfied by the center point. Then it tries to find a smaller ellipsoid that contains all the region of the original ellipsoid that is to one side of that linear constraint (separating the center point from P). This is guaranteed to contain P and is guaranteed to have a smaller volume by a factor. One significant feature of this method is that one can solve even LP's with exponentially manin constraints as long as there is a polynomial time separation oracle. An *oracle* for an LP is a method that decides if a proposed point $x \in \mathbb{R}^n$ is feasible and, if not, produces a violated constraint. The ellipsoid method, while not nearly as practical as interior point methods, has the advantage of solving an LP in polynomial time if there is a polynomial time oracle for the problem even if the number of constraints is exponential. We will use this property in the design of approximation algorithms.

2.4 Set Cover by deterministic rounding

Let's formulate the Set Cover problem as an integer program and find the LP relaxation. Summarizing the definition of Set Cover we have:

- Input
  - $U = \{e_1, \ldots, e_n\}$ is the *universe* of elements.
  - $S = \{S_1, \ldots, S_m\}$ is a collection of subsets of U with costs $c(S_1), \ldots, c(S_m)$.
- Goal: to find the minimum cost set $S' \subseteq S$ such that the union of all elements of S' give U.

$$\begin{aligned} \text{minimize:} \quad & \sum_{i=1}^{n} c(S_i) \cdot x_i & & & & \\ \text{s.t} \quad & \sum_{e \in S_j} x_j & \geq & 1, \quad \forall\ e_i \in U & (1) \\ & x_i & \in & \{0,1\} & (2) \end{aligned}$$

If we relax constraint (2) to $0 \leq x_i \leq 1$ then we obtain the LP relaxation to the IP. Note that in any IP/LP, the solution to the LP is a lower bound for the IP. Note that, we can remove the condition $x_i \leq 1$ as in any optimal solution there cannot be a value larger than 1. The reason is that in any feasible solution we can round all the $x_i > 1$ to 1 and still get a feasible solution with a lower cost.

A common technique in design of approximation algorithms is to first formulate the problem as an IP/LP, then solve the LP relaxation, and then round this fractional solution to an integer one while keeping a bound on the total value of the objective function. There are different ways to round a fractional solution to an integer one. Here we discuss a deterministic rounding for Set Cover.

We define the frequency of an element $e \in U$ to be the number of the sets of S that contain e and let f be the largest frequency among all elements. For instance, for the case of vertex cover (where the elements are edges and the sets are the vertices) the frequency of each element is 2, so $f = 2$. Below we describe an f-approximation rounding algorithm for set cover.

SC2: Deterministic Rounding for Set Cover

- Let x^* be an optimal solution to the LP relaxation of set-cover.
- For each set i define $\tilde{x}_i = 1$ if $x_i^* \geq 1/f$ and 0 otherwise.
- return the set of sets with $\tilde{x}_i = 1$.

First we claim that this algorithm returns a feasible solution. The reason is that for each element e, constraint (1) implies that among all the at most f sets containing e, at least one set has x^* value at least $1/f$. Therefore, for at least one of those sets $\tilde{x}$ value is 1. To see that this is an f-approximation it is sufficient to observe that for each set S_i picked in our solution, the fractional solution is paying at least $\frac{1}{f} \cdot c(S_i)$ since $x_i^* \geq 1/f$.

The above algorithm when applied to vertex cover implies a 2-approximation even for the case of weighted vertex cover in which every node $v \in V$ has a weight (or cost) c_v and the goal is to find a feasible solution with minimum total cost (the cardinality vertex cover is the special case where $c_v = 1$ for all nodes).

2.5 Integrality Gap

For an instance I of a minimization problem, let $OPT_f(I)$ be the cost of the optimal LP solution. Then, integrality gap of this instance is given by $OPT(I)/OPT_f(I)$. The integrality gap of the problem Π is $\max_I OPT(I)/OPT_f(I)$, i.e. the largest ratio between the fractional and integral solution over all possible instances of Π. Note that in any LP rounding algorithm, we use the LP solution as a lower bound for the optimal solution. Therefore, any such algorithm cannot have a performance ratio better than the integrality gap. In general, it is difficult to design algorithms with ratio better than the integrality gap. Therefore, large integrality gaps typically are indications of difficulty of a problem. Consider for example the Vertex Cover problem, and let I be the complete graph K_n. Clearly by assigning a value of $\frac{1}{2}$ to each vertex we get a fractional solution of value $\frac{n}{2}$ whereas any integer solution must contain at least $n-1$ nodes (or else one edge is not covered). Thus the integrality gap of V.C. is at least $2-\frac{2}{n}$.

Exercise. Prove that the integrality gap for Set Cover problem is $\Omega(\log n)$.

2.5.1 Weighted Vertex Cover.

In this section we show that the standard LP for the weighted version of V.C has some other nice features. Recall the following LP relaxation of the vertex cover problem:

$$\begin{aligned} \text{minimize}: \quad & \sum_{i=1}^{n} c_v \cdot x_v \\ \text{subject to}: \quad & x_u + x_v \geq 1, \quad \forall\ uv \in E \\ & x_v \geq 0 \end{aligned}$$

Definition 4 (Half-integer) *A variable is half-integer, if it is an integer factor of* $\{0, \frac{1}{2}, 1\}$.

Our main goal is to prove the following lemma:

Lemma 3 *Any basic solution to the above LP for the Vertex Cover is half-integer*

It's easy to see that by rounding each $x_i \geq \frac{1}{2}$ to 1 we get a 2-approximation algorithm for V.C.

Proof. In order to prove this lemma, we take advantage of the fact that all non-basic feasible solutions can be written as a convex combination of two other feasible solutions. However, basic solutions cannot be written in such a way. Let x be any basic solution to the LP. Assume x is not half-integer. Define:

$$V^{>} = \left\{ v \;\middle|\; \frac{1}{2} < x_v < 1 \right\}$$
$$V^{<} = \left\{ v \;\middle|\; 0 < x_v < \frac{1}{2} \right\}$$

Assuming that x is not half-integer, we must have $V^{>} \cup V^{<} \neq \emptyset$. We would like to show that there exist feasible solutions y_v and z_v such that x can be written as a convex combination of them, which is a contradiction. Define y_v and z_v as follows:

$$y_v = \begin{cases} x_v + \epsilon & x \in V^{>} \\ x_v - \epsilon & x \in V^{<} \\ x_v & \text{Otherwise} \end{cases} \quad \text{and} \quad z_v = \begin{cases} x_v - \epsilon & x \in V^{>} \\ x_v + \epsilon & x \in V^{<} \\ x_v & \text{Otherwise} \end{cases}$$

Observation 1. If $V^> \cup V^< \neq \emptyset$, then $y \neq x$ and $z \neq x$.

Observation 2. $x = \frac{1}{2}(y + z)$; i.e. x is written as a combination of y and z.

We can easily make sure that $0 \leq y_v, z_v \leq 1$ by choosing ϵ sufficiently small. In order to prove y and z are both feasible solutions, we consider two cases:

1. For every edge uv with $x_u + x_v > 1$, we can easily choose ϵ so small that we have $(x_u - \epsilon) + (x_v - \epsilon) \geq 1$. In this case, both $y_u + y_v \geq 1$ and $z_v + z_u \geq 1$ since y_u and z_u are both $\geq x_u - \epsilon$ (similarly for y_v and z_v)
2. Now, suppose $x_u + x_v = 1$. In this case one of x_u and x_v belongs to $V^>$ and the other belongs to $V^<$. Assume $x_u \in V^>$ and $x_v \in V^<$. Therefore $(\underbrace{x_u + \epsilon}_{y_u} - \epsilon) + (\underbrace{x_v - \epsilon}_{y_v} + \epsilon) = 1 \Rightarrow y_u + y_v = 1$ and $(\underbrace{x_u - \epsilon}_{z_u} + \epsilon) + (\underbrace{x_v + \epsilon}_{z_v} - \epsilon) = 1 \Rightarrow z_u + z_v = 1$

Therefore, in z, y are both feasible and x (which is a basic solution) is a convex combination of two feasible solution, a contradiction.

■