CS 598CSC: Approximation Algorithms **Lecture date: January 21, 2011**
Instructor: Chandra Chekuri **Scribe: Sungjin Im (2009)**

Notes edited by instructor in 2011.

In the previous lecture, we had a quick overview of several basic aspects of approximation algorithms. We also addressed approximation (both offline and online) algorithms for the Steiner Tree Problem. In this lecture, we explore another important problem – the Traveling Salesperson Problem (TSP).

1 The Traveling Salesperson Problem (TSP)

1.1 TSP in Undirected Graphs

In the Traveling Salesperson Problem (TSP), we are given an undirected graph $G = (V, E)$ and cost $c(e) > 0$ for each edge $e \in E$. Our goal is to find a Hamiltonian cycle with minimum cost. A cycle is said to be Hamiltonian if it visits every vertex in V exactly once.

TSP is known to be NP-Hard. Moreover, we cannot hope to find a good approximation algorithm for it unless $P = NP$. This is because if one can give a good approximation solution to TSP in polynomial time, then we can exactly solve the NP-Complete Hamiltonian cycle problem (HAM) in polynomial time, which is not possible unless $P = NP$. Recall that HAM is a decision problem: given a graph $G = (V, E)$, does G have a Hamiltonian cycle?

Theorem 1 ([3]) *Let $\alpha : \mathbb{N} \to \mathbb{N}$ be a polynomial-time computable function. Unless $P = NP$ there is no polynomial-time algorithm that on every instance I of TSP outputs a solution of cost at most $\alpha(|I|) \cdot \mathrm{OPT}(I)$.*

Proof: For the sake of contradiction, suppose we have an approximation algorithm $\mathcal{A}$ for TSP with an approximation ratio $\alpha(|I|)$. We show a contradiction by showing that using $\mathcal{A}$, we can exactly solve HAM in polynomial time. Let $G = (V, E)$ be the given instance of HAM. We create a new graph $H = (V, E')$ with cost $c(e)$ for each $e \in E'$ such that $c(e) = 1$ if $e \in E$, otherwise $c(e) = B$, where $B = n\alpha(n) + 2$ and $n = |V|$. Note that this is a polynomial-time reduction since α is a polynomial-time computable function.

We observe that if G has a Hamiltonian cycle, $\mathrm{OPT} = n$, otherwise $\mathrm{OPT} \geq n - 1 + B \geq n\alpha(n) + 1$. (Here, OPT denotes the cost of an optimal TSP solution in H.) Note that there is a "gap" between when G has a Hamiltonian cycle and when it does not. Thus, if $\mathcal{A}$ has an approximation ratio of $\alpha(n)$, we can tell whether G has a Hamiltonian cycle or not: Simply run $\mathcal{A}$ on the graph H; if $\mathcal{A}$ returns a TSP tour in H of cost at most $\alpha(n)n$ output that G has a Hamilton cycle, otherwise output that G has no Hamilton cycle. We leave it as an exercise to formally verify that this would solve HAM in polynomial time. $\square$

Since we cannot even approximate the general TSP problem, we consider more tractable variants.

- *Metric-TSP*: In Metric-TSP, the instance is a complete graph $G = (V, E)$ with cost $c(e)$ on $e \in E$, where c satisfies the triangle inequality, i.e. $c(uw) \leq c(uv) + c(vw)$ for any $u, v, w \in V$.

- *TSP-R*: TSP with repetitions of vertices allowed. The input is a graph $G = (V, E)$ with non-negative edge costs as in TSP. Now we seek a minimum-cost *walk* that visits each vertex at least once and returns to the starting vertex.

Exercise: Show that an α-approximation for Metric-TSP implies an α-approximation for TSP-R and vice-versa.

We focus on Metric-TSP for the rest of this section. We first consider a natural greedy approach, the Nearest Neighbor Heuristic (NNH).

```
NEAREST NEIGHBOR HEURISTIC($G(V, E), c : E \to \mathcal{R}^+$):
Start at an arbitrary vertex $s$,
While (there are unvisited vertices)
  From the current vertex $u$, go to the nearest unvisited vertex $v$.
Return to $s$.
```

Exercise:

1. Prove that NNH is an $O(\log n)$-approximation algorithm. (**Hint:** Think back to the proof of the $2H_{|S|}$-approximation for the Greedy Steiner Tree Algorithm.)
2. NNH is not an $O(1)$-approximation algorithm; can you find an example to show this? In fact one can show a lower bound of $\Omega(\log n)$ on the approximation-ratio achieved by NNH.

There are constant-factor approximation algorithms for TSP; we now consider an MST-based algorithm. See Fig 1.

```
TSP-MST($G(V, E), c : E \to \mathcal{R}^+$):
Compute an MST $T$ of $G$.
Obtain an Eulerian graph $H = 2T$ by doubling edges of $T$
An Eulerian tour of $2T$ gives a tour in $G$.
Obtain a Hamiltonian cycle by shortcutting the tour.
```

Theorem 2 *MST heuristic(TSP-MST) is a 2-approximation algorithm.*

Proof: We have $c(T) = \sum_{e \in E(T)} c(e) \leq \text{OPT}$, since we can get a spanning tree in G by removing any edge from the optimal Hamiltonian cycle, and T is a MST. Thus $c(H) = 2c(T) \leq 2\text{OPT}$. Also shortcutting only decreases the cost. $\square$

We observe that the loss of a factor 2 in the approximation ratio is due to doubling edges; we did this in order to obtain an Eulerian tour. But any graph in which all vertices have even degree is Eulerian, so one can still get an Eulerian tour by adding edges only between odd degree vertices in T. Christofides Heuristic [2] exploits this and improves the approximation ratio from 2 to 3/2. See Fig 2 for a snapshot.

```
CHRISTOFIDES HEURISTIC($G(V, E), c : E \to \mathcal{R}^+$):
Compute an MST $T$ of $G$.
Let $S$ be the vertices of odd degree in $T$. (Note: $|S|$ is even)
Find a minimum cost matching $M$ on $S$ in $G$
Add $M$ to $T$ to obtain an Eulerian graph $H$.
Compute an Eulerian tour of $H$.
Obtain a Hamilton cycle by shortcutting the tour.
```

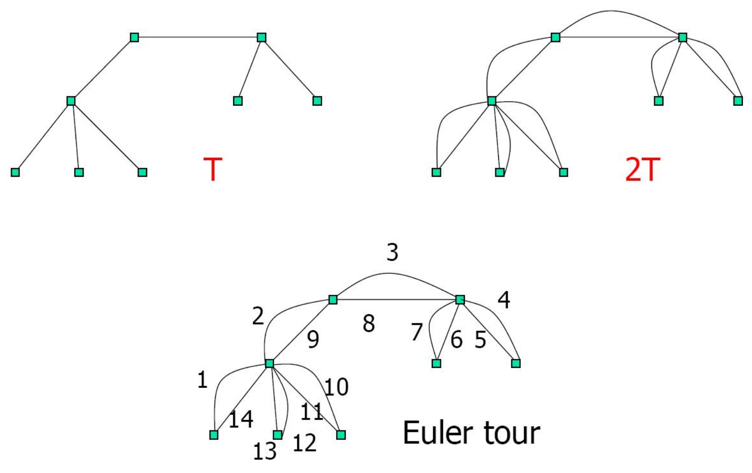

Figure 1: MST Based Heuristic

Theorem 3 *Christofides Heuristic is a 1.5-approximation algorithm.*

Proof: The main part of the proof is to show that $c(M) \leq .5\text{OPT}$. Suppose that $c(M) \leq .5\text{OPT}$. Then, since the solution of Christofides Heuristic is obtained by shortcutting the Eulerian tour on H, its cost is no more than $c(H) = c(T) + c(M) \leq 1.5\text{OPT}$. (Refer to the proof of Lemma 2 for the fact $c(T) \leq \text{OPT}$.) Therefore we focus on proving that $c(M) \leq .5\text{OPT}$.

Let F^* be an optimal tour in G of cost OPT; since we have a metric-instance we can assume without loss of generality that F^* is a Hamiltonian cycle. We obtain a Hamiltonian cycle F^*_S in the graph $G[S]$ by short-cutting the portions of F^* that touch the vertices $V \setminus S$. By the metric-condition, $c(F^*_S) \leq c(F^*) = \text{OPT}$. Let $S = \{v_1, v_2, \ldots, v_{|S|}\}$. Without loss of generality F^*_S visits the vertices of S in the order $v_1, v_2, \ldots, v_{|S|}$. Recall that $|S|$ is even. Let $M_1 = \{v_1v_2, v_3v_4, \ldots v_{|S|-1}v_{|S|}\}$ and $M_2 = \{v_2v_3, v_4v_5, \ldots v_{|S|}v_1\}$. Note that both M_1 and M_2 are matchings, and $c(M_1) + c(M_2) = c(F^*_S) \leq \text{OPT}$. We can assume without loss of generality that $c(M_1) \leq c(M_2)$. Then we have $c(M_1) \leq .5\text{OPT}$. Also we know that $c(M) \leq c(M_1)$, since M is a minimum cost matching on S in $G[S]$. Hence we have $c(M) \leq c(M_1) \leq .5\text{OPT}$, which completes the proof. □

Notes:

1. In practice, local search heuristics are widely used and they perform extremely well. A popular heuristic *2-Opt* is to swap pairs from xy, zw to xz, yw or xw, yz, if it improves the tour.

2. There have been no improvements to Metric-TSP since Christofides heuristic was discovered in 1976. It remains a major open problem to improve the approximation ratio of $\frac{3}{2}$ for Metric-TSP; it is conjectured that the Held-Karp LP relaxation [4] gives a ratio of $\frac{4}{3}$. Very recently a breakthrough has been announced by Oveis-Gharan, Saberi and Singh who claim a $3/2 - \delta$ approximation for some small but fixed $\delta > 0$ for the important special case where $c(e) = 1$ for each edge e.

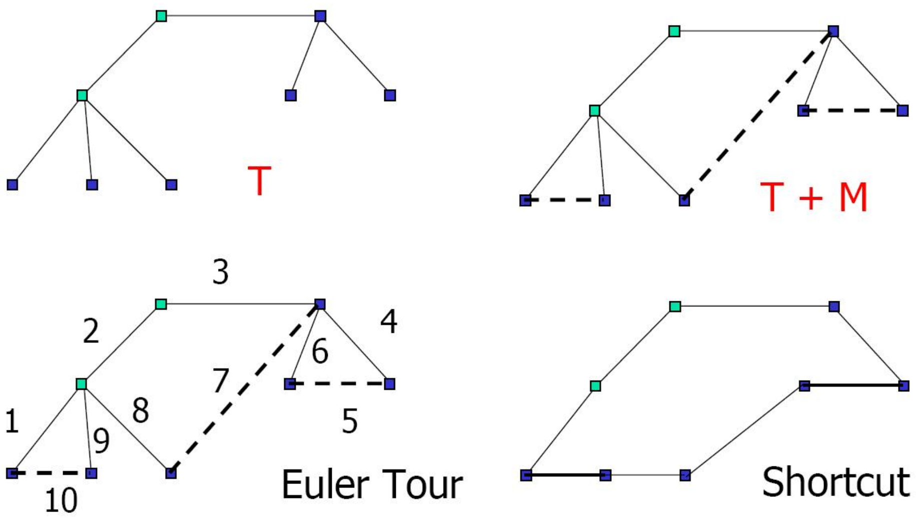

Figure 2: Christofides Heuristic

1.2 TSP in Directed Graphs

In this subsection, we consider TSP in directed graphs. As in undirected TSP, we need to relax the problem conditions to get any positive result. Again, allowing each vertex to be visited multiple times is equivalent to imposing the asymmetric triangle inequality $c(u, w) \leq c(u, v) + c(v, w)$ for all u, v, w. This is called the asymmetric TSP (ATSP) problem. We are given a directed graph $G = (V, A)$ with cost $c(a) > 0$ for each arc $a \in A$ and our goal is to find a closed walk visiting all vertices. Note that we are allowed to visit each vertex multiple times, as we are looking for a walk, not a cycle. For an example of a valid Hamiltonian walk, see Fig 3.

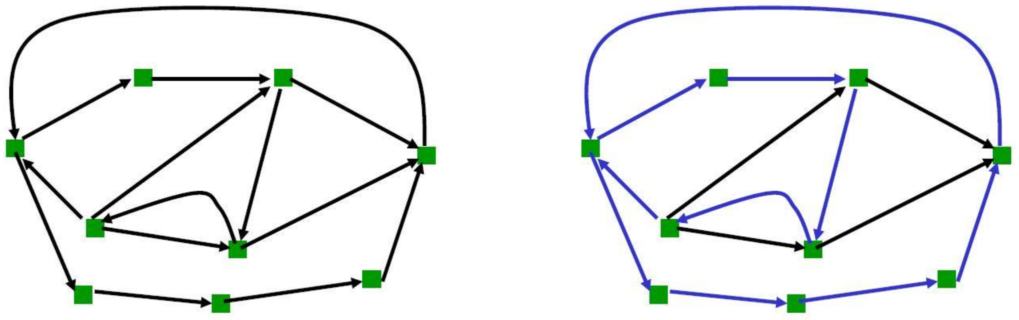
Figure 3: A directed graph and a valid Hamiltonian walk

The MST-based heuristic for the undirected case has no meaningful generalization to the directed setting This is because costs on edges are not symmetric. Hence, we need another approach. The *Cycle Shrinking Algorithm* repeatedly finds a min-cost cycle cover and shrinks cycles, combining the cycle covers found. Recall that a cycle cover is a collection of disjoint cycles covering all vertices. It is known that finding a minimum-cost cycle cover can be done in polynomial time (see

Homework 0). The Cycle Shrinking Algorithm achieves a $\log_2 n$ approximation ratio.

```
CYCLE SHRINKING ALGORITHM($G(V, A), c : A \to \mathcal{R}^+$):
Transform $G$ s.t. $G$ is complete and satisfies $c(u, v) + c(v, w) \geq c(u, w)$ for $\forall u, v, w$
If $|V| = 1$ output the trivial cycle consisting of the single node
Find a minimum cost cycle cover with cycles $C_1, \ldots, C_k$
From each $C_i$ pick an arbitrary proxy node $v_i$
Recursively solve problem on $G[\{v_1, \ldots, v_k\}]$ to obtain a solution $C$
$C' = C \cup C_1 \cup C_2 \ldots C_k$ is a Eulerian graph.
Shortcut $C'$ to obtain a cycle on $V$ and output $C'$.
```

For a snapshot of the Cycle Shrinking Algorithm, see Fig 4.

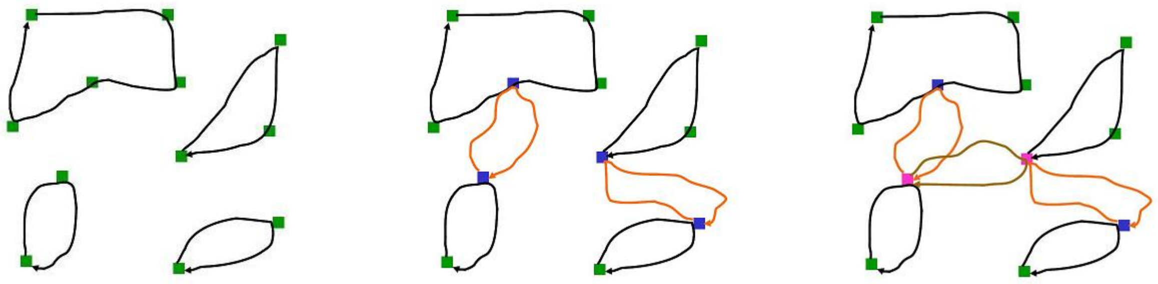

Figure 4: A snapshot of Cycle Shrinking Algorithm. To the left, a cycle cover $\mathcal{C}_1$. In the center, blue vertices indicate proxy nodes, and a cycle cover $\mathcal{C}_2$ is found on the proxy nodes. To the right, pink vertices are new proxy nodes, and a cycle cover $\mathcal{C}_3$ is found on the new proxy nodes.

Lemma 4 *Let the cost of edges in G satisfy the asymmetric triangle inequality. Then for any $S \subseteq V$, the cost of an optimal TSP tour in $G[S]$ is at most the cost of an optimal TSP tour in G.*

Proof: Since G satisfies the triangle inequality there is an optimal tour TSP tour in G that is a Hamiltonian cycle C. Given any $S \subseteq V$ the cycle C can be short-cut to produce another cycle C' that visits only S and whose cost is at most the cost of C. $\square$

Lemma 5 *The cost of a min-cost cycle-cover is at most the cost of an optimal TSP tour.*

Proof: An optimal TSP tour is a cycle cover. $\square$

Theorem 6 *The Cycle Shrinking Algorithm is a $\log_2 n$-approximation for ATSP.*

Proof: We prove the above by induction on n the number of nodes in G. It is easy to see that the algorithm finds an optimal solution if $n \leq 2$. The main observation is that the number of cycles in the cycle-cover is at most $\lfloor n/2 \rfloor$; this follows from the fact that each cycle in the cover has to have at least 2 nodes and they are disjoint. Thus $k \leq \lfloor n/2 \rfloor$. Let $\text{OPT}(S)$ denote the cost of an optimal solution in $G[S]$. From Lemma 4 we have that $\text{OPT}(S) \leq \text{OPT}(V) = \text{OPT}$ for all $S \subseteq V$. The algorithm recurses on the proxy nodes $S = \{v_1, \ldots, v_k\}$. Note that $|S| < n$, and by induction, the cost of the cycle C output by the recursive call is at most $(\log_2 |S|)\text{OPT}(S) \leq (\log_2 |S|)\text{OPT}$.

The algorithm outputs C' whose cost is at most the cost of C plus the cost of the cycle-cover computed in G. The cost of the cycle cover is at most OPT (Lemma 5). Hence the cost of C' is at most $(\log_2 |S|)\text{OPT} + \text{OPT} \leq (\log_2 n/2)\text{OPT} + \text{OPT} \leq (\log_2 n)\text{OPT}$; this finishes the inductive proof. $\square$

Notes:

1. The running time of the Cycle Shrinking Algorithm is $O(T(n))$ where $T(n)$ is the time to find a min-cost cycle cover (why?). In Homework 0 you have a problem that reduces this problem to that of finding a min-cost perfect matching in an *undirected* graph. This can be done in $O(n^3)$-time. One can improve the running time to $O(n^2)$ for by approximating the min-cost cycle-cover problem; one loses an additional constant factor.

2. It has remained an open problem for more than 25 years whether there exists a constant factor approximation for ATSP. Recently Asadpour *et al.* [1] have obtained an $O(\log n/\log\log n)$-approximation for ATSP using some very novel ideas and a well-known LP relaxation.

2 Some Definitions

2.1 NP Optimization Problems

In this section, we cover some formal definitions related to approximation algorithms. We start from the definition of optimization problems. A problem is simply an infinite collection of *instances*. Let Π be an optimization problem. Π can be either a minimization or maximixation problem. Instances I of Π are a subset of Σ^* where Σ is a finite encoding alphabet. For each instance I there is a set of feasible solutions $\mathcal{S}(I)$. We restrict our attention to real/rational-valued optimization problems; in these problems each feasible solution $S \in \mathcal{S}(I)$ has a value $val(S, I)$. For a minimization problem Π the goal is, given I, find $\text{OPT}(I) = \min_{S\in\mathcal{S}(I)} val(S, I)$.

Now let us formally define NP optimization (NPO) which is the class of optimization problems corresponding to NP.

Definition 7 Π *is in NPO if*

- *Given $x \in \Sigma^*$, there is a polynomial-time algorithm that decide if x is a valid instance of* Π. *That is, we can efficiently check if the input string is well-formed. This is a basic requirement that is often not spelled out.*
- *For each I, and $S \in \mathcal{S}(I)$, $|S| \leq poly(|I|)$. That is, the solution are of size polynomial in the input size.*
- *There exists a poly-time decision procedure that for each I and $S \in \Sigma^*$, decides if $S \in \mathcal{S}(I)$. This is the key property of NP; we should be able to* verify *solutions efficiently.*
- *$val(I, S)$ is a polynomial-time computable function.*

We observe that for a minimization NPO problem Π, there is a associated natural decision problem $L(\Pi) = \{(I, B) : \text{OPT}(I) \leq B\}$ which is the following: given instance I of Π and a number B, is the optimal value on I at most B? For maximization problem Π we reverse the inequality in the definition.

Lemma 8 $L(\Pi)$ *is in NP if* Π *is in NPO.*

2.2 Relative Approximation

When Π is a minimization problem, recall that we say an approximation algorithm $\mathcal{A}$ is said to have approximation ratio α iff

- $\mathcal{A}$ is a polynomial time algorithm
- for all instance I of Π, $\mathcal{A}$ produces a feasible solution $\mathcal{A}(I)$ s.t. $val(\mathcal{A}(I), I) \leq \alpha\, val\,(\mathrm{OPT}(I), I)$. (Note that $\alpha \geq 1$.)

Approximation algorithms for maximization problems are defined similarly. An approximation algorithm $\mathcal{A}$ is said to have approximation ratio α iff

- $\mathcal{A}$ is a polynomial time algorithm
- for all instance I of Π, $\mathcal{A}$ produces a feasible solution $\mathcal{A}(I)$ s.t. $val(\mathcal{A}(I), I) \geq \alpha\, val\,(\mathrm{OPT}(I), I)$. (Note that $\alpha \leq 1$.)

For maximization problems, it is also common to see use $1/\alpha$ (which must be ≥ 1) as approximation ratio.

2.3 Additive Approximation

Note that all the definitions above are about relative approximations; one could also define *additive* approximations. $\mathcal{A}$ is said to be an α-additive approximation algorithm, if for all I, $val(\mathcal{A}(I)) \leq \mathrm{OPT}(I)+\alpha$. Most NPO problems, however, do not allow any additive approximation ratio because $\mathrm{OPT}(I)$ has a scaling property.

To illustrate the scaling property, let us consider Metric-TSP. Given an instance I, let I_β denote the instance obtained by increasing all edge costs by a factor of β. It is easy to observe that for each $S \in \mathcal{S}(I) = \mathcal{S}(I_\beta)$, $val(S, I_\beta) = \beta val(S, I_\beta)$ and $\mathrm{OPT}(I_\beta) = \beta \mathrm{OPT}(I)$. Intuitively, scaling edge by a factor of β scales the value by the same factor β. Thus by choosing β sufficiently large, we can essentially make the additive approximation(or error) negligible.

Lemma 9 *Metric-TSP does not admit an α additive approximation algorithm for any polynomial-time computable α unless $P = NP$.*

Proof: For simplicity, suppose every edge has integer cost. For the sake of contradiction, suppose there exists an additive α approximation $\mathcal{A}$ for Metric-TSP. Given I, we run the algorithm on I_β and let S be the solution, where $\beta = 2\alpha$. We claim that S is the optimal solution for I. We have $val(S, I) = val(S, I_\beta)/\beta \leq \mathrm{OPT}(I_\beta)/\beta + \alpha/\beta = \mathrm{OPT}(I) + 1/2$, as $\mathcal{A}$ is α-additive approximation. Thus we conclude that $\mathrm{OPT}(I) = val(S, I)$, since $\mathrm{OPT}(I) \leq val(S, I)$, and $\mathrm{OPT}(I)$, $val(S, I)$ are integers. This is impossible unless $P = NP$. $\square$

Now let us consider two problems which allow additive approximations. In the Planar Graph Coloring, we are given a planar graph $G = (V, E)$. We are asked to color all vertices of the given graph G such that for any $vw \in E$, v and w have different colors. The goal is to minimize the number of different colors. It is known that to decide if a planar graph admits 3-coloring is NP-complete, while one can always color any planar graph G with using 4 colors. Further, one can efficiently check whether a graph is 2-colorable (that is, if it is bipartite). Thus, the following algorithm is a 1-additive approximation for Planar Graph Coloring: If the graph is bipartite, color it with 2 colors; otherwise, color with 4 colors.

As a second example, consider the Edge Coloring Problem, in which we are asked to color edges of a given graph G with the minimum number of different colors so that no two adjacent edges have different colors. By Vizing's theorem [6], we know that one can color edges with either $\Delta(G)$ or $\Delta(G)+1$ different colors, where $\Delta(G)$ is the maximum degree of G. Since $\Delta(G)$ is a trivial lower bound on the minimum number, we can say that the Edge Coloring Problem allows a 1-additive approximation. Note that it is known to be NP-complete to decide whether the exact minimum number is $\Delta(G)$ or $\Delta(G)+1$.

2.4 Hardness of Approximation

Now we move to hardness of approximation.

Definition 10 (Approximability Threshold) *Given a minimization optimization problem Π, it is said that Π has an approximation threshold $\alpha^*(\Pi)$, if for any $\epsilon > 0$, Π admits a $\alpha^*(\Pi)+\epsilon$ approximation but if it admits a $\alpha^*(\Pi)-\epsilon$ approximation then $P = NP$.*

If $\alpha^*(\Pi) = 1$, it implies that Π is solvable in polynomial time. Many NPO problems Π are known to have $\alpha^*(\Pi) > 1$ assuming that $P \neq NP$. We can say that approximation algorithms try to decrease the upper bound on $\alpha^*(\Pi)$, while hardness of approximation attempts to increase lower bounds on $\alpha^*(\Pi)$.

To prove hardness results on NPO problems in terms of approximation, there are largely two approaches; a direct way by reduction from NP-complete problems and an indirect way via gap reductions. Here let us take a quick look at an example using a reduction from an NP-complete problem.

In the (metric) k-center problem, we are given an undirected graph $G = (V, E)$ and an integer k. We are asked to choose a subset of k vertices from V called centers. The goal is to minimize the maximum distance to a center, i.e. $\min_{S \subseteq V, |S|=k} \max_{v \in V} \mathrm{dist}_G(v, S)$, where $\mathrm{dist}_G(v, S) = \min_{u \in S} \mathrm{dist}_G(u, v)$.

The k-center problem has approximation threshold 2, since there are a few 2-approximation algorithms for k-center and there is no $2-\epsilon$ approximation algorithm for any $\epsilon > 0$ unless $P = NP$. We can prove the inapproximability using a reduction from the decision version of Dominating Set: Given an undirected graph $G = (V, E)$ and an integer k, does G have a dominating set of size at most k? A set $S \subseteq V$ is said to be a dominating set in G if for all $v \in V$, $v \in S$ or v is adjacent to some u in S. Dominating Set is known to be NP-complete.

Theorem 11 ([5]) *Unless $P = NP$, there is no $2-\epsilon$ approximation for k-center for any fixed $\epsilon > 0$.*

Proof: Let I be an instance of Dominating Set Problem consisting of graph $G = (V, E)$ and integer k. We create an instance I' of k-center while keeping graph G and k the same. If I has a dominating set of size k then $\mathrm{OPT}(I') = 1$, since every vertex can be reachable from the Dominating Set by at most one hop. Otherwise, we claim that $\mathrm{OPT}(I') \geq 2$. This is because if $\mathrm{OPT}(I') < 2$, then every vertex must be within distance 1, which implies the k-center that witnesses $\mathrm{OPT}(I')$ is a dominating set of I. Therefore, the $(2-\epsilon)$ approximation for k-center can be used to solve the Dominating Set Problem. This is impossible, unless $P = NP$. $\square$

References

[1] A. Asadpour, M. Goemans, A. Madry, S. Oveis Gharan, and A. Saberi. An $O(\log n / \log\log n)$-approximation algorithm for the assymetric traveling salesman problem. *Proc. of ACM-SIAM SODA*, 2010.

[2] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. *Technical Report, Graduate School of Industrial Administration, CMU* 1976.

[3] S. Sahni and T.F. Gonzalez. P-Complete Approximation Problems. *J. of the ACM* 23: 555-565, 1976.

[4] M. Held and R. M. Karp. The traveling salesman problem and minimum spanning trees. *Operations Research* 18: 1138–1162, 1970.

[5] W. L. Hsu and G.L. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics.* 1:209-216, 1979.

[6] D. West. *Introductin to Graph Theory.* 2d ed. Prentice Hall. 277-278, 2001.