| | |
|---|---|
| **CS787: Advanced Algorithms** | **Scribe:** Taylor Kemp |
| **Lecture 5:** Approximation Algorithms:Introduction | **Date:** Feb 7, 2019 |

**Outlines for the Following Lecture**

- General Framework for Optimization
- Approximation Algorithms on Vertex Cover
- Analysis of Approximation Algorithms
- Approximation Algorithms for Travelling Salesman Problem

## 5.1 General Framework for Optimization

A problem is defined by a tuple $(\mathcal{F}(I), Obj)$ where $\mathcal{F}(I)$ is the feasible set of instance $I$ and $Obj$ is some objective function.

### 5.1.1 Approximation Algorithms for Vertex Cover

**Recall** : A vertex cover of a graph $G = (E, V)$ is a subset $s \subset V$ s.t. $s$ contains all edges in E.

For vertex cover we can assign the following values to fit it into our optimization framework

| Instance | G(V, E) |
|---|---|
| Output | subsets s of V |
| Obj | size of vertex cover |
| $\mathcal{F}(G)$ | $\{s : \ s\ covers\ E\}$ |

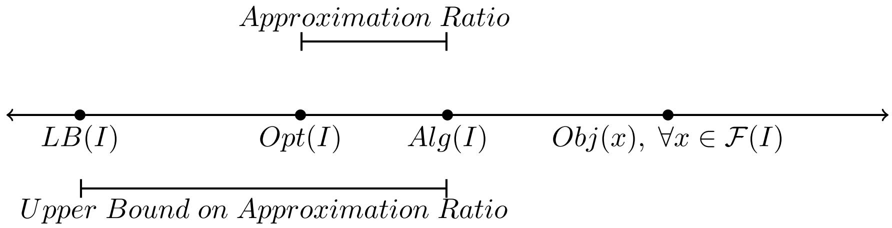
For minimization problem we define the following:

$$\begin{aligned} Opt(I) &= \min_{x \in \mathcal{F}(I)} Obj(x) \\ Opt(I) &= \arg\min_{x \in \mathcal{F}(I)} Obj(x) \\ Alg(I) &: \ \text{return } x \in \mathcal{F}(I) \\ Alg(I) &= Obj(\hat{Alg}(I)) \end{aligned}$$

Notice that we abused notation to let $Opt$ and $Alg$ both stands for the objective value and the feasible argument that achieves the corresponding objective vlaue.

## 5.2 Analysis of Approximate Algorithms

We can motivate the following analysis through the following pictorial representation of our metrics we have currently defined.




**Definition 5.2.1** *For a minimization problem, we define the approximation factor $\alpha$ of an algorithm as*

$$\alpha = \max_I \left(\frac{Alg(I)}{Opt(I)}\right) \geq 1$$

*Similarly, for a maximizarion problem, we define the approximation factor $\alpha$ as*

$$\alpha = \max_I \left(\frac{Opt(I)}{Alg(I)}\right) \geq 1$$

**Definition 5.2.2** *Lower Bound is any function LB s.t.*
*1. LB is easy to compute*
*2. $LB(I) \leq Opt(I),\ \forall\ I$*
*3. hopefully, $\frac{Opt(I)}{LB(I)}$ is small, $\forall I$*

In order to analyze the effectiveness of our algorithm as well as the lower bound we have determined, we define the following five metrics.

- What approximation factor $\alpha$ does the algorithm achieve?
- If $\exists\ I\ s.t.\ \frac{Alg(I)}{Opt(I)} = \alpha$ we say the analysis is tight.
- If $\exists\ I\ s.t.\ \frac{Alg(I)}{LB(I)} = \alpha$ we say the analysis for LB is tight.
- Does there exist an instance $I\ s.t.\ \frac{Opt(I)}{LB(I)} = \alpha$?
- Is there an efficient algorithm with approximation factor lower than $\alpha$? This can indicate the hardness of an approximation result.

**Example:** Let us consider the algorithm that approximates the optimal vertex cover through the maximal matching of a graph. We will then analyze it across the five metrics from above.

- From previous size of vertex recover returned is $2 \cdot |M| \leq 2 \cdot Opt(G) \implies \alpha = 2$

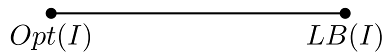



Figure 5.2.1: Instance where analysis is tight

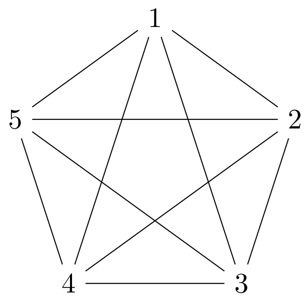



Figure 5.2.2: Instance where $\frac{Opt(I)}{LB(I)}$ is tight

- See 5.2 to see an example where the optimal is a vertex cover of size one and our algorithm returns a solution of size 2 as the Maximal matching has two endpoints. The optimal solution is one node hence our analysis is tight.
- Also tight from the example for number two.
- If we consider the case of a fully connected graph the minimum size vertex cover is $n - 1$ where n is the number of nodes in the graph. The Maximal matching however will be one half of the number of nodes. 5.2
- Vertex Cover is NP-Hard to approximate within $2 - \epsilon$ where $\epsilon$ is a constant

■

## 5.3 Approximation Algorithms for Traveling Salesman Problem

The traveling salesman problem is one where a set of $V$ cities are provided as well as paths connecting each city. The goal of the salesman is to make a tour and visit every city at least once in as little travel time as possible. We denote $\mathcal{F}$ as all tours that visit every vertex in the graph $G(V, E)$, where G is complete and has non-negative weights on its edges $e \in E$. The tour begins at a node $v \in V$ and our Objective function Obj is defined as the sum of all weights used on the path for the tour, $Obj(tour) = length$. It is also worth noting that as we can visit a city multiple times, the edge weights satisfy the triangle inequality, defined below. We define the following potential Lower Bound functions LB for traveling salesman problem.

- $\sum_v \min_{(u,v)\in E} W_{u,v}$ the sum of the minimum path away from each node.

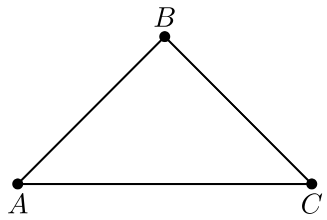



Figure 5.3.3: The above graph satisfies triangle inequality if $d(A, C) \leq d(A, B) + d(B, C)$

- Diameter of the graph G.
- Minimum Spanning Tree. This works as every tour visits every node at least once, so every tour contains a spanning tree. As such we can define the lower bound function LB as the smallest such spanning tree.
- Min cost of the Maximal matching

---

**Algorithm 1** Christofieds Algorithm Version 1

1: G=(V, E)
2: Find minimum spanning tree T of G
3: Construct tour that visits every edge of T twice

---

**Definition 5.3.1** *A graph G is Eulerian if every node has even degree.*

**Lemma 5.3.2** *If the graph G is Eulerian, then we may construct a tour that visits each edge exactly once, hence it visits each node at least once, and has cost equal to the sum of the cost of all edges in the graph.*

---

**Algorithm 2** Christofieds Algorithm Version 2

1: G=(V, E)
2: Find minimum spanning tree T of G
3: Let S be all odd degree vertices in T
4: Find minimum cost perfect matching over S $\rightarrow$ M
5: $T \cup M$ is Eulerian, construct a tour over $T \cup M$

---

Notice that in step 4 of Algorithm 2, we can always find such a perfect matching because there are even number of nodes in S. For the above 2 algorithm, we have that

$$\text{Cost of Algorithm 1's solution} = 2wt(T) \leq Opt$$
$$\text{Cost of Algorithm 2's solution} = wt(T) + wt(M) \leq 1.5 \cdot Opt$$