Tutorial 7 - Approximation algorithms

Gidon Rosalki

2024-12-19

Definition 0.1. *Let there be an optimisation problem*

$$\max_{x \in X} \{f(x)\}$$

Linear programming: X - polyhedron, $f(x) = C^T x$ - linear function.

For example - MST: X - the collection of all the spreading trees of the graph, $f(x)$ - the weight of the spanning tree.

Definition 0.2 (C-approximation)**.** *We will say that an algorithm is a C-approximation of a **maximisation** problem for $C \geq 1$. We will write that the input is X, and the optimal solution is X^*. So the algorithm is a C-approximation if $f(x) \geq \frac{1}{C} f(X^*)$*

Similarly, for a minimisation problem: $f(x) \leq c \cdot f(X^*)$

1 Max-Cut

Input: An undirected graph $G = (V, E)$.
Definition: a **cut** in the graph is a splitting of V into 2 sets, A and B, such that $A \cap B = \emptyset \wedge A \cup B = V$.
Size: The cut is of size the number of edges where one of their nodes are in A, and the other is in B.
Output: A cut of maximum size.

1.0.1 Algorithm 2-approximation

Symbols: $\Gamma(v) \subseteq V$ is the set of neighbours of the nodes v.

1. **Preprocessing:** We will number the nodes in some way $V = (v_1, \ldots, v_n)$
2. **Initialisation:** We will initialise $B = \emptyset, A = V$
3. **Iteration:** We will pass over the nodes in the order $1, \ldots, n$, if $v_i \in A \wedge |\Gamma(v_i) \cap A| > |\Gamma(v_i) \cap B|$ then we will move v_i to B. Similarly, if $v_i \in B \wedge |\Gamma(v_i) \cap B| > |\Gamma(v_i) \cap A|$ then we will move v_i to A
4. **End:** We will repeat the iterative step until there are no more nodes moved, and then return.

1.0.2 Runtime

Theorem 1. *Every time we move a node from one set to the other in the iteration step, we increase the size of the cut by at least 1.*

Proof . Let us say that A, B is the cut before we make a change, and A', B' afterwards. We will assume that $v \in A$. So $A' = A \setminus \{v\} \wedge B' = B \cup \{v\}$. □

1. $O(n),\ n = |V|,\ m = |E|$
2. $O(n)$
3. $O(m + n)$
4. $O(m)$

So our total runtime is $O(m(m + n))$.

1.0.3 Correctness

The solution is correct since we start with a correct solution, and every step make a change which does not impact the correctness of the solution.

Reminder: $\sum_{v \in V} |\Gamma(v)| = 2m$. We shall write $|\Gamma(v)| = d(v)$.

Theorem 2 (2-approximation)**.** *We will write that t is the* ***size*** *of the cut that is returned by the algorithm. We will write t^* to be the optimal cut size. So $t \geq \frac{1}{2} t^*$*

Proof . Let there be A, B, the cut that the algorithm returned.

$$\begin{aligned} t &= \sum_{v \in A} |\Gamma(v) \cap B| \\ &= \sum_{v \in B} |\Gamma(v) \cap A| \\ &= \frac{1}{2} \left(\sum_{v \in A} |\Gamma(v) \cap B| + \sum_{v \in B} |\Gamma(v) \cap A| \right) \\ &= * \end{aligned}$$

Note that from the node $v \in A$, $|\Gamma(v) \cap B| \geq \frac{1}{2} |\Gamma(v)|$ since $d(v) = |\Gamma(v)| = |\Gamma(v) \cap A| + |\Gamma(v) \cap B|$, and according to the algorithm implementation $|\Gamma(v) \cap B| \geq |\Gamma(v) \cap A|$. Therefore:

$$\begin{aligned} * &\geq \frac{1}{2} \left(\sum_{v \in A} \frac{1}{2} d(v) + \sum_{v \in B} \frac{1}{2} d(v) \right) \\ &= \frac{1}{4} \sum_{v \in V} d(v) \\ &= \frac{2m}{4} \\ &= \frac{1}{2} m \\ &\geq \frac{1}{2} t^* \end{aligned}$$

Since the size of the cut cannot rise above the number of edges in the graph. □

The best known approximation: Goemans - Williamson, where $\frac{1}{C} = 0.87\ldots$, and finding a better approximation is NP-hard

2 Travelling salesman problem - TSP

Input: A complete undirected graph $G = (V, E)$, and a weight function $\sigma : E \to \mathbb{R}$.

Definition 2.1 (Hamiltonian cycle)**.** *A cycle that passes through all the nodes of the graph at most once.*

Output: A Hamiltonian cycle of minimum weight.
Assumption: The triangle inequality holds: $\forall v, w, y : \sigma(v, u) \leq \sigma(v, w) + \sigma(w, u)$.

2.0.1 Algorithm 2-approximation

1. We find an MST - T.
2. We will choose some node v_1
3. We will run DFS on the tree from v_1, and number the nodes according to the first time we find them in DFS.
4. We will return the cycle $H = (v_1, \ldots, v_n, v_1)$

2.0.2 Runtime

Symbols: $n = |V|,\ m = |E|$

1. $O(m \log(m))$
2. $O(1)$
3. $O(n)$
4. $O(1)$

So in total, $O(m \log(m))$

Theorem 3 (2-approximation)**.** *Let W be the weird of the Hamiltonian cycle that the algorithm returns, and W^* the optimal weight. So $W \leq 2 \cdot W^*$*

Proof . Note that if we remove an edge from the Hamiltonian cycle, then we get a spanning tree.
Let us write H^* to be some optimal cycle, and T^* to be the spanning tree that we get by removing some edge from H^*. We shall note that

$$\sigma(T) \leq \sigma(T^*) \leq \sigma(H^*)$$

We want to show that $\frac{1}{2}\sigma(H) \leq \sigma(T)$. Therefore if $\sigma(H) \leq 2\sigma(T)$, we have finished. We shall write p to be the **complete walk** of DFS, which is to say we are adding the node v to the list p, both the first time we visit it, and also after we have finished to visit all the of its sub trees. We shall note that $\sigma(p) = 2\sigma(T)$. Therefore it is sufficient to show that $\sigma(H) \leq \sigma(p)$.

$$\begin{aligned} \sigma(H) &\leq \sigma(p) \\ &= 2\sigma(T) \\ &=\leq 2 \cdot \sigma(T^*) \\ &\leq 2 \cdot \sigma(H^*) \end{aligned}$$

Indeed, H is created from p by removing nodes. For every node, we will leave its first appearance, and remove all others. At every remove, we remove two edges $\{u, v\}, \{v, w\}$ and add one edge $\{u, w\}$. From the triangle inequality, the weight of the graph did not increase. □