SAPIENZA
UNIVERSITÀ DI ROMA

"Sapienza" University of Rome
Faculty of Information Engineering,
Informatics and Statistics
Department of Computer Science

Advanced Algorithms

Lecture notes integrated with the book TODO

Author
Alessio Bandiera

February 27, 2025

Contents

Information and Contacts

Personal notes and summaries collected as part of the *Advanced Algorithms* course offered by the degree in Computer Science of the University of Rome "La Sapienza".

Further information and notes can be found at the following link:
https://github.com/aflaag-notes. Anyone can feel free to report inaccuracies, improvements or requests through the Issue system provided by GitHub itself or by contacting the author privately:

- Email: **alessio.bandiera02@gmail.com**
- LinkedIn: **Alessio Bandiera**

The notes are constantly being updated, so please check if the changes have already been made in the most recent version.

Suggested prerequisites:

- Progettazione degli Algoritmi

Licence:

1 TODO

1.1 TODO

1.1.1 The Max Cut problem

The **Max Cut problem** — in the unweighted case — is a classic combinatorial optimization problem in the branch of graph theory, in which we seek to partition the vertices of an undirected graph into two disjoint subsets while maximizing the number of edges that have endpoints in both subsets. More formally, we will define a **cut** of a graph as follows.

Definition 1.1: Cut of a graph

Given an undirected graph $G = (V, E)$, and a subset of its vertices $S \subseteq V$, the **cut** induced by S on G is defined as follows

$$\text{cut}(S) := \{e \in E \mid |S \cap e| = 1\}$$

Note that in the definition above we are defining the cut of a graph through the intersection between a set of vertices S and edges in E; this is because, in the undirected case, we will consider the edges of a graph $G = (V, E)$ as sets of 2 elements

$$E = \{\{u, v\} \mid u, v \in V\}$$

Therefore, given a set of vertices S, the cut induced by S is simply the set of edges that have only one endpoint in S (implying that the other one will be in $V - S$).

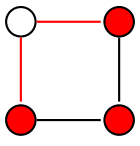

Figure 1.1: Given the set of red vertices S, the green edges represent $\text{cut}(S)$.

With this definition, we can introduce the **Max Cut** problem, which is defined as follows.

Definition 1.2: Max Cut problem

Given an undirected graph $G = (V, E)$, determine the set $S \subseteq V$ that maximizes $|\text{cut}(S)|$.

Although this problem is known to be APX-Hard [1], approximation algorithms and heuristic methods like greedy algorithms and local search are commonly used to find near-optimal solutions.

For now, we present the following **randomized algorithm**, which provides a straightforward $\frac{1}{2}$-approximation for the Max Cut problem. This algorithm runs in polynomial time and achieves the approximation guarantee with high probability.

Algorithm 1.1: Random Cut

Given an undirected graph $G = (V, E)$, the algorithm returns a cut of the graph.

```
1: function RandomCut(G)
2:     S := ∅
3:     for v ∈ V do
4:         Let c_v be the outcome of the flip of an independent fair coin
5:         if c_v == heads then
6:             S = S ∪ {v}
7:         end if
8:     end for
9:     return S
10: end function
```

Note that this algorithm does not care about the structure of the graph in input, since the output is completely determined by the coin flips performed in the `for` loop. Now we will prove that this algorithm is a correct **expected $\frac{1}{2}$-approximation** of the Max Cut problem.

Theorem 1.1: Expected approximation ratio of RandomCut

Let $G = (V, E)$ be a graph, and let S^* be an optimal solution to the Max Cut problem on G. Then, given $S = \text{RandomCut}(G)$, it holds that

$$\mathbb{E}[|\text{cut}(S)|] \geq \frac{|\text{cut}(S^*)|}{2}$$

Proof. By definition, note that

$$\forall e \in E \quad e \in \text{cut}(S) \iff |S \cap e| = 1$$

Consider an edge $e = \{v, w\} \in E$; then, by definition

$$\{v, w\} \in \text{cut}(S) \iff (v \in S \wedge w \notin S) \vee (v \notin S \wedge w \in S)$$

and let ξ_1 and ξ_2 be these last two events respectively. Then

$$\Pr[\xi_1] = \Pr[c_v = \text{heads} \wedge c_w = \text{tails}]$$

by definition of the algorithm, and by independence of the flips of the fair coins we have that

$$\Pr[\xi_1] = \Pr[c_v = \text{heads}] \cdot \Pr[c_w = \text{tails}] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

Analogously, we can show that

$$\Pr[\xi_2] = \frac{1}{4}$$

This implies that

$$\Pr[e \in \text{cut}(S)] = \Pr[\xi_1 \vee \xi_2] = \Pr[\xi_1] + \Pr[\xi_2] - \Pr[\xi_1 \wedge \xi_2] = \frac{1}{4} + \frac{1}{4} - 0 = \frac{1}{2}$$

Hence, we have that

$$\mathbb{E}[|\text{cut}(S)|] = \sum_{e \in E} 1 \cdot \Pr[e \in \text{cut}(S)] = \frac{|E|}{2} \geq \frac{|\text{cut}(S^*)|}{2}$$

where the last inequality directly follows from the definition of cut of a graph. □

As previously mentioned, this algorithm has an **expected approximation ratio** of $\frac{1}{2}$, which implies that it may return very bad solutions in some cases, depending on the outcomes of the coin flips. However, thanks to the following algorithm, we can actually transform the **guarantee of expectations** into a **guarantee of high probability**.

Algorithm 1.2: t-times Random Cut

Given an undirected graph $G = (V, E)$ and an integer $t > 0$, the algorithm returns a cut of the graph.

1: **function** t-timesRandomCut(G, t)
2: **for** $i \in [t]$ **do**
3: $S_i :=$ RandomCut(G)
4: **end for**
5: **return** $S \in \arg\max_{i \in [t]} |\text{cut}(S_i)|$
6: **end function**

The algorithm above simply runs the RandomCut algorithm t times, and returns the set S_i that maximizes the cut, among all the various $S_1, \ldots, S_t$. The following theorem will show that a *reasonable number* of runs of the RandomCut algorithm suffices in order to almost certainly obtain a $\approx \frac{1}{2}$-approximation of any optimal solution.

Theorem 1.2

Let $G = (V, E)$ be a graph, and let S^* be an optimal solution to the Max Cut problem on G. Then, given $S = t\text{-timesRandomCut}(G, t)$, it holds that

$$\Pr\left[|\text{cut}(S)| > \frac{1-\varepsilon}{2}|\text{cut}(S^*)|\right] > 1 - \delta$$

where $t = \frac{2}{\varepsilon}\ln\frac{1}{\delta}$ and $0 < \varepsilon, \delta < 1$.

Proof. For each $i \in [t]$, let $C_i := |\text{cut}(S_i)|$ for each S_i defined by the algorithm, and let $N_i := |E| - C_i$. Let $0 < \varepsilon < 1$; since N_i is a non-negative random variable, by Markov's inequality we have that

$$\Pr[N_i \geq (1+\varepsilon)\mathbb{E}[N_i]] \leq \frac{1}{1+\varepsilon} = 1 - \frac{\varepsilon}{1+\varepsilon} \leq 1 - \frac{\varepsilon}{2}$$

In particular, this inequality can be rewritten as follows:

$$\begin{aligned} 1 - \frac{\varepsilon}{2} &\geq \Pr[N_i \geq (1+\varepsilon)\mathbb{E}[N_i]] \\ &= \Pr[|E| - C_i \geq (1+\varepsilon)(|E| - \mathbb{E}[C_i])] \\ &= \Pr[-\varepsilon|E| \geq C_i - (1+\varepsilon)\mathbb{E}[C_i]] \end{aligned}$$

As shown in the proof of Theorem 1.1, we know that $\mathbb{E}[C_i] = \frac{|E|}{2}$, therefore

$$\begin{aligned} 1 - \frac{\varepsilon}{2} &\geq \Pr[-\varepsilon|E| \geq C_i - (1+\varepsilon)\mathbb{E}[C_i]] \\ &= \Pr\left[-\varepsilon|E| \geq C_i - \frac{1+\varepsilon}{2}|E|\right] \\ &= \Pr\left[-\varepsilon\frac{|E|}{2} \geq C_i - \frac{|E|}{2}\right] \\ &= \Pr\left[\frac{1-\varepsilon}{2}|E| \geq C_i\right] \\ &= \Pr\left[(1-\varepsilon)\mathbb{E}[C_i] \geq C_i\right] \end{aligned}$$

Note that the event in the last probability, namely

$$|\text{cut}(S_i)| \leq (1-\varepsilon)\mathbb{E}[|\text{cut}(S_i)|]$$

corresponds to a "bad" solution, i.e. one whose cardinality is at most $(1-\varepsilon)$-th of the expected value.

By definition of the algorithm, each of the t runs of the RandomCut algorithm is independent from the others, therefore the probability of *all* the solutions $S_1, \ldots, S_t$ being "bad" is bounded by

$$\Pr[\forall i \in [t] \quad C_i \leq (1-\varepsilon)\mathbb{E}[C_i]] = \prod_{i=1}^{t} \Pr[C_i \leq (1-\varepsilon)\mathbb{E}[C_i]] \leq \left(1 - \frac{\varepsilon}{2}\right)^t$$

Using the fact that

$$\forall x \in \mathbb{R} \quad 1 - x \leq e^{-x} \implies 1 - \frac{\varepsilon}{2} \leq e^{-\frac{\varepsilon}{2}}$$

we have that

$$\Pr[\forall i \in [t] \quad C_i \leq (1-\varepsilon)\mathbb{E}[C_i]] \leq \left(1 - \frac{\varepsilon}{2}\right)^t \leq e^{-\frac{\varepsilon}{2} \cdot t} = e^{-\ln \frac{1}{\delta}} = \delta$$

Therefore, the probability that at least one among $S_1, \ldots, S_t$ is a "good" solution is bounded by

$$\Pr[\exists i \in [t] \quad C_i > (1-\varepsilon)\mathbb{E}[C_i]] = 1 - \Pr[\forall i \in [t] \quad C_i \leq (1-\varepsilon)\mathbb{E}[C_i]] \geq 1 - \delta$$

placeholder □

last part

Note that this result is *very powerful*: for instance, if $\varepsilon = \delta = 0.1$, we get that

$$\Pr[|\text{cut}(S)| > 0.45 \cdot |\text{cut}(S^*)|] \geq 0.9$$

and $t \approx 46$, meaning that we just need to run the RandomCut algorithm approximately 46 times in order to get a solution that is better than a 0.45-approximation with 90% probability.

Bibliography

[1] Sanjeev Arora et al. "Proof verification and the hardness of approximation problems". In: *Journal of the ACM* 45.3 (May 1998), 501–555. ISSN: 1557-735X. DOI: 10.1145/278298.278306. URL: http://dx.doi.org/10.1145/278298.278306.