Randomized Algorithms
Lecture 6

Kousha Etessami

Max-Cut

Recall: for an undirected graph $G = (V, E)$, a *cut* is a partition of V into two non-empty sets, $(S, V \setminus S)$. The *capacity* of the cut $(S, V \setminus S)$ is the total number $|C_S|$ of edges that cross the cut, where

$$C_S = \{\{u, v\} \in E \mid u \in S, \ \& \ v \in V \setminus S\}$$

A *maximum cut* in G is a cut with maximum capacity.

The **Max-Cut problem**: given G, find a maximum cut.

This is a classic NP-hard problem.
Indeed, it is **NP-complete** to decide, given G and k, whether the size of the max-cut is $\geq k$. ([Karp,1972])

So we believe there is no polynomial-time algorithm to compute this exactly.

Question: How about approximating a max-cut? Can we do so efficiently? For example, can we get within factor $\frac{1}{2}$ of a max cut in polynomial time?

We will next show, using a trivial randomized algorithm, that every graph $G = (V, E)$ has a cut of size *at least* $|E|/2$, which can be found efficiently.

Max-Cut: random partitioning

Consider the following trivial **random partition** algorithm:

Algorithm RANDOMCUT$(G = (V, E))$

$S \leftarrow \emptyset$

for every $v \in V$ **do**

  Flip an (independent) fair coin, X_v .

  if (X_v = "Heads") **then**

    $S \leftarrow S \cup \{v\}$

return S, $V \setminus S$

Question: What is the *expected size* of C_S?

Max-Cut: analyzing random partitioning

Theorem (6.3)

Let $(S, V \setminus S)$ be the output of RandomCut*. Then* $E[|C_S|] = \frac{|E|}{2}$.

Proof.

Consider the random S created by RandomCut.
For each edge $e = \{u, v\} \in E$, let I_e be the indicator variable of whether e is in C_S or not. There are 4 possibilities for each $e = \{u, v\} \in E$:

$$(a)\ u, v \in S; \quad (b)\ u \in S, v \notin S; \quad (c)\ u \notin S, v \in S; \quad (d)\ u, v \notin S.$$

Note: in exactly 2 out of the 4 cases (namely, (b) an (c)) we have $e \in C_S$.
Thus, $\mathrm{E}[I_e] = \Pr[e \in C_S] = \frac{1}{2}$.
Note that $|C_S| = \sum_{e \in E} I_e$. Hence, summing over all $e \in E$, and by linearity of expectation,

$$\mathrm{E}[|C_S|] = \sum_{e \in E} \mathrm{E}[I_e] = \frac{|E|}{2}.$$

□

Max-Cut: analyzing the random partition algorithm

Corollary

For any *graph* $G = (V, E)$, *there exists* some *cut* $(S, V \setminus S)$ *such that* $|C_S| \geq |E|/2$.

Proof.

Basic but useful observation: If the *expected* size of C_S is $|E|/2$, then there *certainly* must exist at least one cut of at least that size. □

The probabilistic method

- The proof that every graph has a cut of cardinality $\geq |E|/2$ is a very very simple example of the probabilistic method.

- With the probabilistic method, we use randomness and the laws of probability/expectation to prove that a certain combinatorial object must exist.

The probabilistic method

The (basic) probabilistic method:

- Draw a random object from a set of candidate objects Ω;
- Prove that the probability that the random object satisfies a certain property is strictly positive;
- Therefore, an object satisfying that property must exist!

This is a non-constructive method of proving the existence of combinatorial objects, pioneered by Paul Erdős.

Although this approach uses probability, the result (that some object with the property exists) is a definite fact, not a probabilistic statement.

Although it only tells us that some object satisfying some desired property exist, in many cases we can also find / construct the object efficiently.

More on the Probabilistic Method later in the course.

De-randomization

- We did not analyse the probability that RandomCut gives a good (high cardinality) cut, and are not going to do that.
- Instead, we will in fact *de-randomize* the algorithm using conditional expectation, to obtain a deterministic algorithm that always produces a cut with capacity at least $\frac{|E|}{2}$.

De-randomization

We derandomize via “conditional expectation”.

We are interested in the value of $|C_S|$, and the (conditional) expected value of this quantity will change throughout the algorithm, as vertices get added to S or $\overline{S}$.

Our randomized algorithm considered the vertices in fixed order. Let $X_1, \ldots,$ X_n be the indicator random variables ($X_i = 1$ means that v_i is added to S, whereas $X_i = 0$ means v_i is added to $\overline{S}$).

Our deterministic derandomized algorithm will construct a specific cut (defined inductively by assigning values $x_1, \ldots, x_n$ to $X_1, \ldots, X_n$) of size $\geq \frac{|E|}{2}$, by making decisions for the vertices sequentially one-by-one. At each step we will ensure we choose x_{k+1} so that

$$\mathrm{E}[|C_S| \mid X_1 = x_1, \ldots, X_{k+1} = x_{k+1}] \geq \mathrm{E}[|C_S| \mid X_1 = x_1, \ldots, X_k = x_k].$$

Derandomization cont'd.

Suppose $V = \{v_1, \ldots, v_n\}$, and suppose we have considered vertices v_1, ..., v_k sequentially so far, and we have taken decisions x_1, ..., x_k for these vertices.

Suppose (the induction hypothesis) we know that

$$\mathrm{E}[|C_S| \mid X_1 = x_1, \ldots, X_k = x_k] \geq \mathrm{E}[|C_S|].$$

Think about the (random) process of adding v_{k+1}. There are two choices for x_{k+1}, of equal probability. Hence,

$$\begin{aligned} \mathrm{E}[|C_S| \mid X_1 = x_1, \ldots, X_k = x_k] \quad = \quad & \frac{\mathrm{E}[|C_S| \mid X_1 = x_1, \ldots, X_k = x_k, X_{k+1} = 1]}{2} \\ & + \frac{\mathrm{E}[|C_S| \mid X_1 = x_1, \ldots, X_k = x_k, X_{k+1} = 0]}{2}. \end{aligned}$$

Hence, one of the two conditional expectations on the right hand side must be $\geq \mathrm{E}[|C_S| \mid X_1 = x_1, \ldots, X_k = x_k]$, which (by induction) is $\geq \mathrm{E}[|C_S|] = \frac{|E|}{2}$.

Derandomization cont'd.

In our de-randomized algorithm, how do we decide the value of X_{k+1}?

For $i \in \{0, 1\}$, we want to compute the conditional expectations

$$Z_i := \mathrm{E}[|C_S| \mid X_1 = x_1, \ldots, X_{k+1} = i].$$

Recall the linearity of conditional expectation, and $|C_S| = \sum_{e \in E} I_e$. Hence, we just need to compute

$$Z_{i,e} := \mathrm{E}[I_e \mid X_1 = x_1, \ldots, X_{k+1} = i]$$

for each $e \in E$, and then note that $Z_i = \sum_{e \in E} Z_{i,e}$.

Derandomization cont'd.

$$Z_{i,e} = \mathrm{E}[I_e \mid X_1 = x_1, \ldots, X_{k+1} = i]$$

There are three possibilities of the two endpoints of e:

- Both have been determined — the conditional expectation is 0 or 1;
- One of them is determined — the conditional expectation is 1/2;
- None of them is determined — the conditional expectation is 1/2.

Moreover, these values are easy to compute.
Thus we can compute $Z_{i,e}$ for each e, and sum them up to compute the desired Z_i.

Then we compare Z_0 and Z_1, and choose the larger of the two.

Derandomization cont'd.

To decide the value of X_{k+1}, all we care actually is whether or not $Z_1 - Z_0 \geq 0$. In particular, we will let $x_{k+1} := 1$ (respectively $x_{k+1} := 0$) precisely when $Z_1 - Z_0 \geq 0$ (respectively, $Z_1 - Z_0 < 0$). Note that

$$Z_1 - Z_0 = \sum_{e \in E} Z_{1,e} - Z_{0,e}.$$

Back to the possibilities for e:

- If neither endpoints of e is v_{k+1}, $Z_{1,e} = Z_{0,e}$;
- If one endpoint of e is v_{k+1} and the other end is not determined, then $Z_{1,e} = Z_{0,e} = \frac{1}{2}$.
- If one endpoint of e is v_{k+1} and the other end is determined, then $Z_{1,e} \neq Z_{0,e}$.

Thus we only need to care about the last case.

Derandomization cont'd.

For $i = 0, 1$, let $A_i := \{v_j \mid j \in [k],\ X_j = i, \{v_j, v_{k+1}\} \in E\}$.
Namely A_1 (respectively A_0) is the set of neighbours of v_{k+1} from among the already determined nodes $\{v_1, \dots, v_k\}$ that are in S (respectively, in $\overline{S}$).

Each vertex in A_1 contributes 1 to Z_0,

and each vertex in A_0 contributes 1 to Z_1.

Thus, $Z_1 - Z_0 = |A_0| - |A_1|$.

So, what is the de-randomized algorithm?

Algorithm: starting from the first vertex to the last, assign the current vertex to S or to $\overline{S}$ so as to maximize the current cut value.

So, what is the de-randomized algorithm?

Algorithm: starting from the first vertex to the last, assign the current vertex to S or to $\overline{S}$ so as to maximize the current cut value.

This amounts to just the following simple "**greedy**" algorithm: consider the vertices in some specific order $v_1, \ldots, v_n$. Add v_1 to S (arbitrarily). Then, successively, add the next vertex v_i to the side which has fewer of its neighbors (resulting in the larger addition to the size of the cut), breaking ties arbitrarily.

So, what is the de-randomized algorithm?

Algorithm: starting from the first vertex to the last, assign the current vertex to S or to $\overline{S}$ so as to maximize the current cut value.

This amounts to just the following simple "**greedy**" algorithm: consider the vertices in some specific order $v_1, \ldots, v_n$. Add v_1 to S (arbitrarily). Then, successively, add the next vertex v_i to the side which has fewer of its neighbors (resulting in the larger addition to the size of the cut), breaking ties arbitrarily.

The greedy algorithm is guaranteed to yield a cut at least as large and the expected size of a random cut, which is $\frac{|E|}{2}$.

Hence, we can efficiently (and *deterministically*) $\frac{1}{2}$-approximate a MaxCut.

Question: Can we do better than factor $\frac{1}{2}$?

Max-Cut: the Goemans-Williamson algorithm

In fact, there is a (randomized) polynomial-time algorithm for Max-Cut with a better than $\frac{1}{2}$-approximation ratio.

In a breakthrough result, Goemans and Williamson (1995) gave a Max-Cut algorithm with approximation ratio ≈ 0.87856.

Improving upon this ratio would disprove a major conjecture in computational complexity: either the $\mathbf{P} \neq \mathbf{NP}$ conjecture, or the "Unique Games Conjecture" ([Khot'02]) which asserts NP-hardness of a certain problem.

Max-Cut: the Goemans-Williamson algorithm

In fact, there is a (randomized) polynomial-time algorithm for Max-Cut with a better than $\frac{1}{2}$-approximation ratio.

In a breakthrough result, Goemans and Williamson (1995) gave a Max-Cut algorithm with approximation ratio ≈ 0.87856.

Improving upon this ratio would disprove a major conjecture in computational complexity: either the $\mathbf{P} \neq \mathbf{NP}$ conjecture, or the "Unique Games Conjecture" ([Khot'02]) which asserts NP-hardness of a certain problem.

Max-Cut is equivalent to the following quadratic integer program:

$$\text{MaxCut}(G) = \max : \frac{1}{2} \sum_{\{u,v\} \in E} 1 - x_u x_v$$

$$\text{Subject to:} \quad x_v \in \{+1, -1\}, \quad \forall v \in V.$$

Max-Cut: the Goemans-Williamson algorithm

In fact, there is a (randomized) polynomial-time algorithm for Max-Cut with a better than $\frac{1}{2}$-approximation ratio.

In a breakthrough result, Goemans and Williamson (1995) gave a Max-Cut algorithm with approximation ratio ≈ 0.87856.

Improving upon this ratio would disprove a major conjecture in computational complexity: either the $\mathbf{P} \neq \mathbf{NP}$ conjecture, or the "Unique Games Conjecture" ([Khot'02]) which asserts NP-hardness of a certain problem.

Max-Cut is equivalent to the following quadratic integer program:

$$\mathrm{MaxCut}(G) \;=\; \max : \; \frac{1}{2} \sum_{\{u,v\} \in E} 1 - x_u x_v$$

$$\text{Subject to:} \quad x_v \in \{+1, -1\}, \quad \forall v \in V.$$

We cannot solve this efficiently. Instead, we can solve the following Semidefinite Programming relaxation:

$$\mathrm{MaxCut}^R(G) \;=\; \max : \; \frac{1}{2} \sum_{\{u,v\} \in E} 1 - \langle \vec{x}_u, \vec{x}_v \rangle$$

$$\text{Subject to:} \quad \vec{x}_v \in \mathbb{R}^n \;\; \& \;\; \|\vec{x}_v\|_2 = 1, \quad \forall v \in V.$$

Goemans-Williamson algorithm

For $G = (V, E)$, with $|V| = n$, solving

$$\mathrm{MaxCut}^{R}(G) \;=\; \max : \; \frac{1}{2} \sum_{\{u,v\} \in E} 1 - \langle \vec{x}_u, \vec{x}_v \rangle$$

$$\text{Subject to:} \quad \vec{x}_v \in \mathbb{R}^n \;\; \& \;\; \|\vec{x}_v\|_2 = 1, \quad \forall v \in V.$$

gives us a collection of n vectors, $\vec{x}_v$, $v \in V$, on the unit sphere in $\mathbb{R}^n$.

Question: How do we extract an approximately optimal $\{+1, -1\}$ solution x to $\mathrm{MaxCut}(G)$, from a solution to $\mathrm{MaxCut}^{R}(G)$?

Goemans-Williamson algorithm

For $G = (V, E)$, with $|V| = n$, solving

$$\mathrm{MaxCut}^R(G) \;=\; \max : \;\frac{1}{2} \sum_{\{u,v\} \in E} 1 - \langle \vec{x}_u, \vec{x}_v \rangle$$

$$\text{Subject to:}\quad \vec{x}_v \in \mathbb{R}^n \;\;\&\;\; \|\vec{x}_v\|_2 = 1, \quad \forall v \in V.$$

gives us a collection of n vectors, $\vec{x}_v$, $v \in V$, on the unit sphere in $\mathbb{R}^n$.

Question: How do we extract an approximately optimal $\{+1, -1\}$ solution x to $\mathrm{MaxCut}(G)$, from a solution to $\mathrm{MaxCut}^R(G)$?

Answer: We do a kind of "randomized rounding" of the vector solution. Specifically, choose a "random vector", $\vec{r} \in \mathbb{R}^n$, and set $x_v := +1$ if $\langle \vec{x}_v, \vec{r} \rangle \geq 0$, and otherwise, set $x_v := -1$.

Equivalently: r describes a random hyperplane, H, through the origin that cuts the unit sphere in half, defining a partition of the vertices (via the partition defined by H of the unit vectors associated with vertices).

It can be shown that the approximation ratio of this algorithm is at least:

$$\frac{2}{\pi} \cdot \min_{0 \leq \theta \leq \pi} \frac{\theta}{(1 - \cos\theta)} \approx 0.87856\,.$$

References and reading

The $\frac{1}{2}$-approximation algorithm for MaxCut is covered in Sections 6.2, 6.3 of the book.

The book does not cover the Goemans-Williamson algorithm, and we will not expect you to know the Goemans-Williamson algorithm for the exam. (The G-W algorithm has also been derandomized, using a more involved application of the method of conditional expectations ([Mahajan-Ramesh,1995]).)

If you want to learn more about these and other approximation algorithms for NP-hard problems, two very nice books on the subject are:

V. Vazirani, *Approximation Algorithms*, Springer, 2001.

D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*, Cambridge University Press, 2011.

We will return to the probabilistic method, and derandomization, later in the lectures.

We will cover Chernoff Bounds next. It's a good idea to start reading the early sections of Chapter 4.