Approximation Algorithms for Max Cut

Angeliki Chalki
Network Algorithms and Complexity, 2015

NTUA - MPLA

June 19, 2015

Definition

- **Max Cut Definition**: Given an undirected graph G=(V, E), find a partition of V into two subsets A, B so as to maximize the number of edges having one endpoint in A and the other in B.
- **Weighted Max Cut Definition**: Given an undirected graph G=(V, E) and a positive weght w_e for each edge, find a partition of V into two subsets A, B so as to maximize the combined weight of the edges having one endpoint in A and the other in B.

NP- Hardness

Max- Cut:

- is **NP- Hard** (reduction from 3-NAESAT).
- is the same as finding **maximum bipartite subgraph of G**.
- can be thought of a variant of the **2-coloring problem in which we try to maximize the number of edges consisting of two different colors**.
- is **APX-hard** [Papadimitriou, Yannakakis, 1991]. There is no PTAS for Max Cut unless P=NP.

Max Cut with unit weights

The algorithm iteratively updates the cut.

1. Initialize the cut arbitrarily.
2. For any vertex v which has less than $\frac{1}{2}$ of its edges crossing the cut, we move the vertex to the other side of the cut.
3. If no such vertex exists then stop else go to step 2.

Each iteration involves examining at most $|V|$ vertices before moving one, hence $O(|V^2|)$ time.
The cut value is increased by at least 1 after each iteration. The maximum possible cut value is $|E|$, hence there are **at most** $|\mathbf{E}|$ **iterations**.

Overall time complexity: $O(|V^2| \cdot |E|)$.

What is the approximation ratio of this algorithm?

At least half of the edges of each vertex contributes to the solution.
$\forall \upsilon \in A : \sum_{u\in B} w_{(\upsilon,u)} \geq \frac{1}{2} \cdot \sum_{(\upsilon,u)\in E} w_{(\upsilon,u)}$
$\forall u \in B : \sum_{\upsilon\in A} w_{(u,\upsilon)} \geq \frac{1}{2} \cdot \sum_{(u,\upsilon)\in E} w_{(u,\upsilon)}$

Summing over all vertices:
$2 \cdot \sum_{\upsilon\in A, u\in B} w_{(u,\upsilon)} \geq \sum_{e\in E} w_e \geq OPT$

- What is a tight example for this algorithm?
- A better ratio cannot be achieved using $\sum_{e\in E} w_e$ as an upper bound for the maxcut. In complete graphs the max cut is half the size of the upper bound.

The previous algorithm for Weighted Max Cut

For the general case of weighted graphs, the time complexity becomes $O(|V^2| \cdot \sum_{e \in E} w_e)$ which is not always polynomial in the size of the input.

Note. There exist modifications to the algorithm that give a strongly polynomial running time, with an additional ϵ term in the approximation coefficient.

Randomized Approximation Algorithm for Max Cut

There is a simple randomized algorithm for Max Cut:

> Assign each vertex at random to A or to B with equal probability, such that the random decisions for the different vertices are mutually independent.

The expected weight of the solution is:

$$\mathbb{E}\left(\sum_{e \in E(A,B)} w_e\right) = \sum_{e \in E} w_e \cdot Pr(e \in E(A,B)) =$$
$$= \frac{1}{2}\sum_{e \in E} w_e \geq \frac{1}{2} OPT.$$

In the case of Max Cut with unit weights the expected number of cut edges is $\frac{|E|}{2}$.

Las Vegas algorithm for Max Cut with unit weights

In the case of unit weights ($w_e = 1 \ \forall e \in E$), we can obtain a Las Vegas algorithm repeating the previous procedure.

Let, $p = Pr\left(\sum_{e \in E(A,B)} w_e \geqslant \frac{|E|}{2} \right)$. Then,

$$\frac{|E|}{2} = \mathbb{E}\left(\sum_{e \in E(A,B)} w_e \right) =$$

$$\sum_{i \leqslant \frac{|E|}{2} - 1} i \cdot Pr\left(\sum_{e \in E(A,B)} w_e = i \right) + $$

$$+ \sum_{i \geqslant \frac{|E|}{2}} i \cdot Pr\left(\sum_{e \in E(A,B)} w_e = i \right)$$

$\leqslant (1 - p) \cdot \left(\frac{|E|}{2} - 1 \right) + p \cdot |E|$, which implies that

$p \geqslant \frac{1}{\frac{|E|}{2} + 1}$ and the expected number of steps before finding a cut of value at least $\frac{|E|}{2}$ is $\frac{|E|}{2} + 1$.

Derandomization using conditional expectations

Instead of random choices, **we evaluate both alternatives according to the conditional expectation** of the objective function if we fix the decisions until now.
We need three sets A, B, C. Initially C=V and A=B=∅.
At an intermediate state, the expected weight $w(A, B, C)$ of the random cut produced by this procedure is:

$$w(A, B, C) = \sum_{e \in E(A,B)} w_e + \frac{1}{2} \sum_{e \in E(A,C)} w_e + \frac{1}{2} \sum_{e \in E(B,C)} w_e + \frac{1}{2} \sum_{e \in E(C,C)} w_e.$$

Derandomization using conditional expectations

Initialize $A = B = \emptyset$, $C = V$.
for all $v \in V$ **do**
  Compute $w(A + v, B, C - v)$ and $w(A, B + v, C - v)$.
  if $w(A + v, B, C - v) > w(A, B + v, C - v)$ **then**
    $A = A + v$
  else
    $B = B + v$
  end if
  $C = C - v$
end for
return A,B

Derandomization using conditional expectations

The analysis of the algorithm is based on the observation that:

Initially $w(A, B, C) = \frac{1}{2} w_e$.

For every A,B,C and $\upsilon \in V$:
$max\{w(A + \upsilon, B, C - \upsilon), w(A, B + \upsilon, C - \upsilon)\} \geq w(A, B, C)$.

The algorithm computes a partition (A,B) such that the weight of the cut is at least $\frac{1}{2} w_e$.

Obtaining a greedy algorithm

The algorithm can be simplified!
Observe that:

$$w(A+\upsilon, B, C-\upsilon) - w(A, B+\upsilon, C-\upsilon) = \sum_{e\in E(\upsilon,B)} w_e - \sum_{e\in E(\upsilon,A)} w_e.$$

The following algorithm is a 2-approximation algorithm running in linear time!

Greedy Algorithm

Initialize $A = B = \emptyset$.
for all $v \in V$ **do**
  if $\sum_{e \in E(v,B)} w_e - \sum_{e \in E(v,A)} w_e > 0$ **then**
    $A = A + v$
  else
    $B = B + v$
  end if
end for
return A,B

This algorithm first appeared in the 1967 paper "On bipartite subgraphs of graphs" of Erdős.

Greedy Algorithm

The property $\sum_{e \in E(A,B)} w_e \geq \sum_{e \in E(A,A)} w_e + \sum_{e \in E(B,B)} w_e$ is a loop invariant of the algorithm.

It is easy to see that after any loop more edges (weight of edges) are added to the cut than added inside the set A or B.

But,
$\sum_{e \in E(A,B)} w_e + \left(\sum_{e \in E(A,A)} w_e + \sum_{e \in E(B,B)} w_e \right) = \sum_{e \in E} w_e$,
hence $\sum_{e \in E(A,B)} w_e \geq \frac{1}{2} \cdot \sum_{e \in E} w_e$.

PCP Theorem and Inapproximability

A decision problem L belongs to $PCP_{c(n),s(n)}[r(n), q(n)]$, if there is a randomised oracle Turing Machine V (verifier) that, on input x and oracle access to a string w (the proof or witness), satisfies the following properties:

Completeness: If $x \in L$ then for some w, $V^w(x)$ accepts with probability at least c(n).

Soundness: If $x \notin L$ then for every w, $V^w(x)$ accepts with probability at most s(n).

The randomness complexity r(n) of the verifier is the maximum number of random bits that V uses over all x of length n.

The query complexity q(n) of the verifier is the maximum number of queries that V makes to w over all x of length n.

PCP Theorem and Inapproximability

In "Some Optimal Inapproximability results", Håstad [2001], proved inapproximability results based on the following theorem.

PCP Theorem

For every $\epsilon > 0$, $NP = PCP_{1-\epsilon,1/2+\epsilon}[O(logn), 3]$.
Furthermore, the verifier behaves as follows: it uses its randomness to pick three entries i, j, k in the witness w and a bit b, and it accepts if and only if $w_i \oplus w_j \oplus w_k = b$.

Gap introducing reduction from SAT to MaxE3Lin2

For every problem in NP, for example SAT, and for every $\epsilon > 0$ there is a reduction that given a 3CNF formula ϕ constructs a system of linear equations with three variables per equation and:

- If ϕ is satisfiable, there is an assignment to the variables that satisfies a $1 - \epsilon$ fraction of the equations
- If ϕ is not satisfiable, there is no assignment that satisfies more than a $\frac{1}{2} + \epsilon$ fraction of equations.

Obtaining Inapproximability results

Let a reduction f from L_1 to some optimization problem L_2 satisfying the following:

- If $x \in L_1$, then $OPT(f(x)) > k_1$
- If $x \notin L_1$, then $OPT(f(x)) \leqslant k_2$

Then a $\frac{k_1}{k_2}$-approximation algorithm for L_2 can be used to decide L_1.

How? We have that $OPT \leqslant \frac{k_1}{k_2} \cdot SOL$, or $SOL \geqslant \frac{k_2}{k_1} \cdot OPT$. Using this in the case of $x \in L_1$, $SOL \geqslant \frac{k_2}{k_1} \cdot OPT > \frac{k_2}{k_1} \cdot k_1 = k_2$.

Gap preserving reduction from MaxE3Lin2 to Max Cut

In "Gadgets, Approximation, and Linear Programming" [Trevisan et al., 2000] was given a gap preserving reduction from MaxE3Lin2 to Max CUT.

Max Cut can be thought of as a boolean constraint satisfaction problem and from every equation of MaxE3LIN2 instance "cut constraints" are constructed. When the construction is completed a gap between the "yes" and the "no" instances of SAT has been achieved.

PCP Theorem

If there is an r-approximation algorithm for Max CUT, where $r < \frac{17}{16}$, then $\mathrm{P} = \mathrm{NP}$.

Other results

- In 1995, Michel Goemans and David Williamson discovered an algorithm with approximation factor 1.14, based on semidefinite programming.
- If the unique games conjecture is true, this is the best possible approximation ratio for maximum cut[Khot, 2004].