CS 491I Approximation algorithms
Lecture notes
March 15th & 20th ,2001
Dejan Desovski

1. Max-Cut

Problem Statement: Given graph $G = <V, E>$, $|V| = n$, $|E| = m$. Partition the vertices of G in two sets S and T, such that, the number of edges with one vertex in S and the other in T is maximal.

Randomized Algorithm

1. For every vertex in V throw a coin
   a) If it is head put the vertex in S
   b) Otherwise put the vertex in T

Analysis of the Algorithm
If we are given some partition of G in two sets S and T obtained by the previous algorithm, we are interested to find what is the expected number of edges crossing the cut.

The probability that a specified edge (a - b) is crossing the cut is equal to:

$\wp(a - b \text{ crossing cut}) = \wp(a \in S, b \in T) + \wp(b \in S, a \in T) = ½ \cdot ½ + ½ \cdot ½ = ½$

- the first vertex of the edge is in S and the second in T, or the first vertex is in T and the second in S.

we now assign an indicator variable:

$$x_i = \begin{cases} 1, \text{ the edge } i \text{ crosses the cut} \\ 0, \text{ otherwise} \end{cases}$$

The expected number of edges crossing the cut is:

$$E\left[\sum_{i=1}^{m} x_i\right] = \sum_{i=1}^{m} E[x_i] = \frac{m}{2}$$

Thus, this randomized algorithm is a 2-approximation algorithm because the optimal max-cut cannot exceed m.

Local Improvement

We will try to use a local improvement strategy in order to optimize the solution. The algorithm is as follows:

1. Start with some solution, in this case randomized partition of V in two sets S and T
2. Take each vertex in S and check whether if we move it to T the number of edges that cross the cut is going to increase.
   If the cut improves move the vertex to T.

Analysis of the algorithm

It is obvious that the algorithm produces some cut.
Assume that the vertices of ***G*** are partitioned in four sets V_1, V_2, V_3, V_4. And let the optimal partition be: $S = V_1 \cup V_3, T = V_2 \cup V_4$, whereas our algorithm produces the partition: $S = V_1 \cup V_2, T = V_3 \cup V_4$ (figure 1).

Let e_{ij} $\mathbf{1 \le i, j \le 4}$ be the number of edges from $\boldsymbol{V_i}$ to $\boldsymbol{V_j}$.

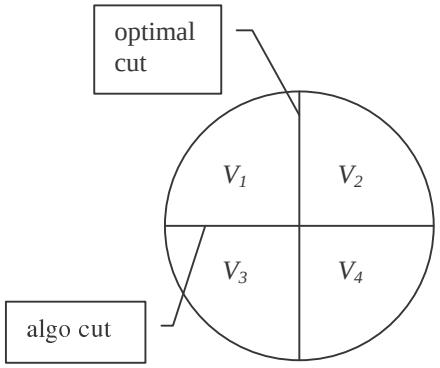

Figure 1

Because of the local optimization property of our algorithm, the following statement holds:
$e_{11} + e_{12} \le e_{13} + e_{14}$

Otherwise, if this is not satisfied, we can move some vertices in $\boldsymbol{V_1}$ to $\boldsymbol{V_3}$ and further increase the number of edges crossing the cut returned by the algorithm, which is a contradiction.
By discarding the first number of the previous inequality we get:
$e_{12} \le e_{13} + e_{14}$
Symmetrically, using the same argument about the local improvement property of the algorithm, we obtain the following inequalities:
$e_{12} \le e_{23} + e_{24}$
$e_{34} \le e_{14} + e_{24}$
$e_{34} \le e_{23} + e_{13}$

If we add the first and the second of these four inequalities we get:
$e_{12} + e_{34} \le e_{13} + e_{14} + e_{23} + e_{24}$

The following inequality is true because the right side is always greater than the left (two positive numbers are added):
$e_{14} + e_{23} \le e_{14} + e_{23} + e_{13} + e_{24}$

If we add them together we get:
$e_{12} + e_{34} + e_{14} + e_{23} \le 2(e_{13} + e_{14} + e_{23} + e_{24})$

Which states:

OPT ≤ 2 · algo

2. Max Independent set

Problem Statement: Given graph ***G*** = <***V***, ***E***>, | ***V*** | = ***n***, | ***E*** | = ***m***, and natural number ***k***. Find subset of vertices ***V'*** ⊆ ***V***, |***V'***| = ***k***, and **(** $\forall i, j \in V', e(i, j) \notin G'$ **)**, where ***G'*** is the graph induced by ***V'***.

Lemma: If a graph ***G*** has minimum vertex cover ***V'*** then its max independent set is ***V – V'***.
Proof: It is obvious that if ***V'*** is a vertex cover, then ***V – V'*** is independent set, otherwise if there is an edge between any two vertices in ***V – V'*** then ***V'*** cannot be a vertex cover because this edge is not covered. And also, by minimizing the vertex cover we maximize the independent set.

We will argue the approximation of the following algorithm:

1. Pick a vertex with minimum degree, let it be *x*
2. Put *x* in the Independent Set
3. Delete all edges incident on x and its neighbors
4. Repeat until there are no more edges

This algorithm can be very bad at certain problem instances, and thus does not have bound with respect to the optimal solution. For example (see figure 2) – this algorithm will return Independent Set with size 2, instead of the optimal *n*. Instead comparing with the optimal solution we will derive a bound of this algorithm which bounds it with the density of the graph. We would expect if the graph is sparse, then the Independent Set should be larger, and also if the graph is dense then the independent set should be smaller.
We will need the following theorem.

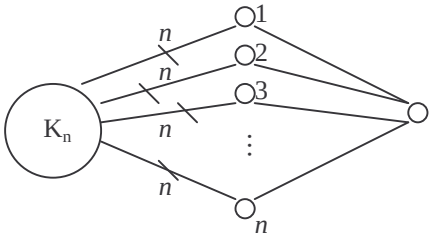

Figure 2

Theorem (**Cauchy - Schwarz Inequality**):

$$\left(\sum_{k=1}^{n} a_k b_k\right)^2 \le \left(\sum_{k=1}^{n} a_k^2\right)\left(\sum_{k=1}^{n} b_k^2\right)$$

Proof:

We can state the theorem in the following way: $\left|\vec{u}\cdot\vec{v}\right|^2 \le \left\|\vec{u}\right\|^2 \cdot \left\|\vec{v}\right\|^2$.

The following always holds: $\left(\vec{v}-t\vec{u}\right)\left(\vec{v}-t\vec{u}\right)\ge 0$, if we multiply we get: $\left\|\vec{v}\right\|^2 + t^2\left\|\vec{u}\right\|^2 - 2t\vec{u}\vec{v} \ge 0$.

We can choose $t = \dfrac{\vec{u}\cdot\vec{v}}{\left\|\vec{u}\right\|^2}$, thus we get: $\left\|\vec{v}\right\|^2 - \dfrac{\left|\vec{u}\cdot\vec{v}\right|^2}{\left\|\vec{u}\right\|^2} \ge 0$, and the theorem follows.

Analysis of the algorithm:

The density of the graph is defined as: $\delta = \dfrac{m}{n}$.

In ***i*^th^** iteration of the algorithm let d_i be the degree of the vertex that is picked (the vertex with the minimum degree). We are going to delete $d_i + 1$ vertices, each of them having degree of at least d_i. The total number of the edges that we are going to delete in this step is $\ge \dfrac{d_i(d_i+1)}{2}$. The division with 2 is because there are some edges counted twice in the product $d_i(d_i + 1)$.
If the algorithm finishes in ***q*** steps, this will be the size of the produced independent set: $|S| = q$.

The total number of deleted edges in q steps should be less then the total number of edges in the graph.

$$\sum_{i=1}^{q} \frac{d_i(d_i+1)}{2} \le m = \delta n$$

The total number of deleted vertices should be equal to n.

$$\sum_{i=1}^{q} (d_i+1) = n$$

We have:

$$\sum_{i=1}^{q} (d_i+1)^2 \le n + 2\delta n = n(2\delta+1)$$

Using Cauchy-Schwarz (with $a_k = d_k + 1$, and $b_k = 1$) for the lower bound we obtain:

$$\frac{n^2}{q} = \frac{1}{q}\left(\sum_{i=1}^{q}(d_i+1)\right)^2 \le \frac{1}{q}\left(\sum_{i=1}^{q}(d_i+1)^2\right)\left(\sum_{i=1}^{q}1^2\right) = n(2\delta+1)$$

Thus, the size of the obtained independent set is: $q \ge \frac{n}{2\delta+1}$.

3. Set Cover

Problem Statement: Given a ground set $U = \{u_1, \ldots, u_m\}$ and a collection of subsets $S = \{S_1, \ldots, S_n\}$, each of the subsets S_i has a cost associated with it w_i. The objective is to find a collection of sets $S' \subseteq S$, that will cover U (maybe some of the elements in U will be covered more than once) with minimum cost.

Proposed Approximation Algorithm:

1. $C = \emptyset, \overline{C} = U - C$
2. while $(C \neq U)$ do
   a) calculate $\alpha_{S_i} = \frac{w_i}{|S_i \cap \overline{C}|}$
   b) choose set S' whith minimum α
   c) $C = C \cup S', \overline{C} = U - C$
   d) for each element e' in S' covered for the first time
      $price(e') = \alpha_{S'}$

All of the elements of U will be covered by this algorithm, lets order them by how they are covered: $e_1, e_2, \ldots, e_m$.

The cost of our algorithm is $\sum_{i=1}^{m} price(e_i)$. Because the cost of the cover is distributed by the algorithm to every element, by assigning the cost of the set that covers the specific element for the first time, to the element.

We focus on the moment when we are picking a set in order to cover e_k.
The number of elements that are still not covered is:

$$\left|\overline{C}\right| \geq n-k+1$$

There is at least one set in each iteration with $\alpha \leq \frac{OPT}{\left|\overline{C}\right|} \leq \frac{OPT}{n-k+1}$.

Otherwise, even the optimal algorithm must create the cover by choosing one by one of the sets. If in each iteration $\alpha > \frac{OPT}{\left|\overline{C}\right|}$ then the optimal algorithm will produce a cover with $\sum_{i=1}^{m} price(e_i) > OPT$, which is a contradiction.

$$price(e_k) \leq \frac{OPT}{n-k+1}$$

$$\text{algo cost} = \sum_{k=1}^{m} price(e_k) \leq \sum_{k=1}^{m} \frac{OPT}{n-k+1} = OPT\left(\sum_{k=1}^{m} \frac{1}{n-k+1}\right) \leq \ln(n) \cdot OPT$$

Rounding Algorithm

Lets consider the IP for the Set Cover:

$$x_i = \begin{cases} 1, \text{if } S_i \text{ is picked} \\ 0, \text{otherwise} \end{cases}$$

$$\min \sum_{i=1}^{n} x_i w_i$$

$$\forall u \in U : \sum_{S_i : u \in S_i} x_i \geq 1$$

$$x_i \in \{0,1\}$$

We can solve the relaxed Integer Program in order to compute f – approximation of this problem, where f is the minimum of the frequencies of the elements in U.

We relax this IP to LP by setting $0 \leq x_i$.

The algorithm is as follows:

1. Solve the LP
2. For each S_i, pick S_i if $x_i \geq \frac{1}{f}$

Consider an arbitrary element e. Since e is in at least f sets, at least one of them will be picked and thus the obtained solution is definitely a cover because each of the elements will be covered. Each of the x_i which represent the sets that are picked is multiplied by at most f so the cost is:

$$\text{algo cost} \le f \cdot (\text{LP - relax}) \le f \cdot OPT$$

Randomized Rounding

Let $x_i = p_i, S_i \in S$. Viewing p_i as probabilities, we will pick each set S_i with probability p_i.

1. Solve the LP
2. For each S_i, pick S_i with probability x_i

The expected cost is:

$$\sum_{S_i \in S} p_i w_i = \sum_{S_i \in S} x_i w_i = LP - relax \le OPT$$

But this may not be regular cover.

Consider an arbitrary element ***u***. We like to compute the probability that ***u*** is covered. Let us say that ***u*** belongs to at most ***k*** sets.

$$x_1 + x_2 + \ldots + x_k \ge 1$$

Value that minimizes this inequality is $x_i = \frac{1}{k}, 1 \le i \le k$.

The probability that ***u*** is covered by at least one set is[1]:

$$1 - \left(1 - \frac{1}{k}\right)^k \ge 1 - \frac{1}{e}$$

Where ***e*** represents the basis of the natural logarithm, $e = 2.71...$

The probability that ***u*** is not covered is $\frac{1}{e}$, which is constant.

In order to get a complete set cover we run the algorithm $c \log n$ times and take union.

The probability that ***u*** is not covered is:

$$\left(\frac{1}{e}\right)^{c \log n} \le \frac{1}{2n}$$

$$\frac{1}{n^c} \le \frac{1}{2n}, \text{for some } c$$

For ***n > 1***, ***c = 2*** is sufficient for this inequality to hold.

The probability that some element is not covered $\le n \frac{1}{2n} \le \frac{1}{2}$.

The expected number of times that we need to repeat this algorithm in order to get a cover is **2.** Because this process represents Bernouli trials with probability for success ***p*** = ½, and the expected number of trials in Bernouli process till success is 1 / ***p***.

Thus the cost of the algorithm is: $2c \log n \cdot LP - relaxed \le 2c \log n \cdot OPT$.

---

[1] Using the following calculus theorem: $\lim_{\alpha \to 0} (1 + \alpha)^{1/\alpha} = e$