CS612 Algorithm Design and Analysis

Lecture 20. MAXCUT problem: random sampling, derandomization, and semi-definite programming [1]

Dongbo Bu

Institute of Computing Technology
Chinese Academy of Sciences, Beijing, China

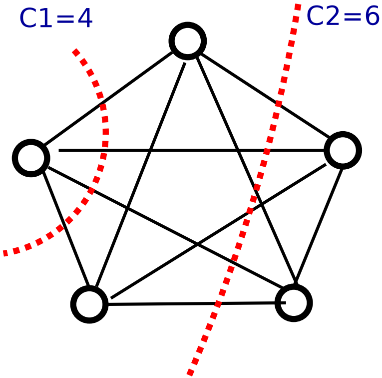
[1] The slides are made based on Approximation Algorithms for NP-Hard problems by D. S. Hochbaum, Computational Complexity by C. H. Papadimitriou, and a report by D. P. Williamson.

Outline I

- Introduction to MaxCut problem;
- NP-Hardness of MaxCut problem;
- Local search algorithm;
- Dumb-randomization algorithm and derandomization;
- "LP+RR" algorithm by Arora, et al;
- Semi-definite programming method;

MaxCut problem

INPUT: An undirected graph $G =< V, E >$.
OUTPUT: A cut of $V = A \cup B$, $A \cap B = \phi$, such that the number of edge crossing the cut is maximized.



Theorem

MaxCut *problem is NP-Hard.*

Proof:

(Reduction from NAESAT to MaxCut.)

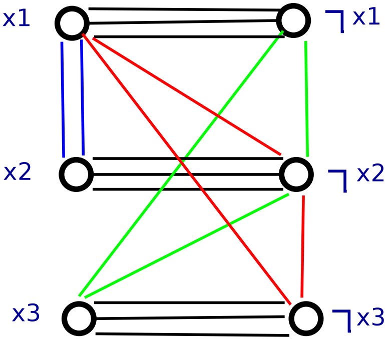
Gaudget: tri-angle. (max cut = 2)

- Nodes: G has $2n$ nodes, including x_i and $\neg x_i$ for each variable i;
- Edges:
  1. Connecting x_i and $\neg x_i$ with n_i edges, where n_i is the total number of occurence of x_i and $\neg x_i$.
  2. For each clause $x_i \vee x_j \vee x_k$, draw a tri-angle; for a clause $(x_1 \vee x_2)$, draw two parallel lines (x_1, x_2), (x_1, x_2).

e.g.:

$(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \Leftrightarrow$

$(x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



Claim: there is a cut with size $k \geq 5m$ in G iff the NAESAT instance is satisfiable.

- $\Rightarrow$:
  If G has a cut S of size $5m$ or more, w.l.o.g, we can assume x_i and $\neg x_i$ are in different side, which contributes $3m$ edges to the cut. The other $2m$ edges come from the tri-angles. Constructing an assignment to set all literals in S to be TRUE. All clauses are NAESAT under this assignment.
  $\Leftarrow$:
  Let S be the literals that are true. Then the cut $(S, V - S)$ has size $3m + 2m = 5m$.

A local search algo

see Lec14.ppt.

A dumb randomized algorithm

Algorithm

1. $A \leftarrow \phi$, $B \leftarrow \phi$,;
2. for i = 1 to n
3. if random($\frac{1}{2}$) = 1
4. $A = A \cup \{i\}$;
5. else
6. $B = B \cup \{i\}$;

Theorem (Sahni, Gonzalez '76)

DumbRandom is a $\frac{1}{2}$-approximation algorithm.

Proof.

- We define a random variable $x_{ij} = 1$ iff i and j are not in A simultanously, and $x_{ij} = 0$ otherwise.
- We define $W = \sum_{i<j} w_{ij} x_{ij}$. We have:

$$
\begin{aligned}
E(W) &= E(\sum_{i<j} w_{ij} x_{ij}) &(1)\\
&= \sum_{i<j} w_{ij} E(x_{ij}) &(2)\\
&= \frac{1}{2} \sum_{i<j} w_{ij} &(3)\\
&\geq \frac{1}{2} OPT &(4)
\end{aligned}
$$

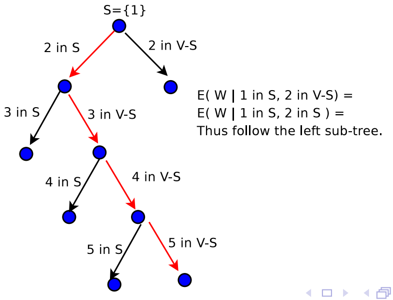
$\square$

Derandomization I

- Changing a randomized algorithm to a deterministic algorithm: Algorithmic derandomization techniques look at a particular randomized algorithm, and using the inherent properties of the problem, analyze the randomized algorithm better to come up with ways to remove randomness from that algorithm.
- Basic idea: conditional expectance. e.g. Since $E(W) = \frac{1}{2}E(W|v_1 \in A) + \frac{1}{2}E(W|v_1 \in B)$ We have $E(W) \leq \max\{E(W|v_1 \in A), E(W|v_1 \in B)\}$.
- Derandomization strategy: **put** v_{i+1} **into** A **if** $E(W|v_1, ..., v_i$ **are determined**$, v_{i+1} \in A) \geq E(W|v_1, ..., v_i$ **are determined**$, v_{i+1} \in B)$.

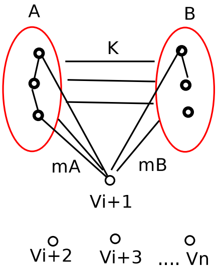
- Repeatedly applying this strategy, we have:

$$\begin{aligned}
\frac{1}{2}OPT &\leq E(W) \\
&\leq E(W|v_1 \text{ is determined according to the strategy}) \\
&\leq E(W|v_1, v_2 \text{ are determined according to the strategy}) \\
&\quad \ldots\ldots\ldots \\
&\leq E(W|v_1, v_2, ..., v_n \text{ are determined according to the stra}
\end{aligned}$$



Derandomization III

- Question: how to calculate $E(W|v_1, ..., v_i \text{ are determined}, v_{i+1} \in A)$?



I: the total number of edges insides A and B.

Derandomization IV

$E(W|v_1, ..., v_i$ are determined$, v_{i+1} \in A) =$
$k + m_B + \frac{1}{2}(m - k - m_A - m_B - l)$
$E(W|v_1, ..., v_i$ are determined$, v_{i+1} \in B) =$
$k + m_A + \frac{1}{2}(m - k - m_A - m_B - l)$
Thus, the strategy can be rewritten as:

- Derandomization strategy: **put** v_{i+1} **into** A **if** $m_A \leq m_B$.
  Algorithm:
  1. $A \leftarrow \{v_1\}$, $B \leftarrow \phi$,;
  2. for i = 2 to n
  3. put i into A or B to maximize the cut size;

Theorem

Let $x_1, x_2, ..., x_n$ be n independent $0/1$ random variabless (not necessarily from the same distribution). Let
$X = x_1 + x_2 + ... + x_n$*, and* $\mu = E[X]$*. For* $0 \leq \delta \leq 1$*,*
$\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{1}{3}\mu\delta^2}$ *and*
$\Pr[X \leq (1-\delta)\mu] \leq e^{-\frac{1}{2}\mu\delta^2}$*.*

Hoeffding bound

Theorem

Let $x_1, x_2, ..., x_n$ be n independent random variabless (not necessarily from the same distribution, and $x_i = 0$ or α_i, where $\alpha_i \leq 1$. Let $X = x_1 + x_2 + ... + x_n$, and $\mu = E[X]$. For $0 \leq \delta \leq 1$,

$\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{1}{3}\mu\delta^2}$ *and*

$\Pr[X \leq (1-\delta)\mu] \leq e^{-\frac{1}{2}\mu\delta^2}$.

- A quadratic programming model:

$$\begin{array}{ll} \max & \sum_{i=1}^{n} x_i \sum_{(i,j)\in E}(1-x_j) \\ s.t. & x_i \in \{0,1\} \end{array}$$

- Definition: let $ZN(x,i)$ denote the number of neighboors in $V-S$ of i under solution x.

$$\begin{array}{ll} \max & \sum_{i=1}^{n} x_i ZN(x,i) \\ s.t. & x_i \in \{0,1\} \end{array}$$

- Let x^* denote the an optimal solution.
- Suppose we have high-quality estimation Z_i of $ZN(x^*,i)$, i.e., $Z_i - \epsilon n \leq ZN(x^*,i) \leq Z_i + \epsilon n$.
- Then we can approximate the quadratic model through the following LP model:

LP+RR Algorithm for dense graph II

$$\begin{array}{ll} \max & \sum_{i=1}^{n} y_i Z_i \\ s.t. & \sum_{(i,j)\in E}(1-y_j) \leq Z_i + \epsilon n \\ & \sum_{(i,j)\in E}(1-y_j) \geq Z_i - \epsilon n \\ & y_i \in \{0,1\} \end{array}$$

Assumption: the graph is dense, i.e., $|E| = \alpha n^*$ and $w_{ij} = 1$.
Observations:

1. $OPT \geq \frac{\alpha}{2}n^2$. (Probability method proof: $E(W) \geq \frac{\alpha}{2}n^2 \Rightarrow OPT \geq \frac{\alpha}{2}n^2$.)
2. x^* is also a feasible solution of the LP model.
3. The objective function of x^* in the LP model is close to that in the quadratic model, (denoted as $OPT = \sum_i (ZN(x^*, i)x_i^*)$. Therefore, $Z_{LP} \geq (1 - \frac{2\epsilon}{\alpha})OPT$.

$$\begin{aligned}
\sum_{i=1}^{n} Z_i x_i^* &\geq \sum_i (ZN(x^*, i) - \epsilon n)x_i^* &(10)\\
&= \sum_i (ZN(x^*, i)x_i^* - \epsilon n \sum_i x_i^* &(11)\\
&\geq OPT - \epsilon n^2 \quad (\sum_i x_i^* \leq n) &(12)\\
&\geq (1 - \frac{2\epsilon}{\alpha})OPT &(13)
\end{aligned}$$

"LP+RR" method by Arora, Karger, and Karpinski, '95.

Algorithm

1. Get Z_i from genie;
2. Solve LP, get y^*;
3. For all node i in V,
4. if random$(y_i^*) = 1$
5. $x'_i = 1$; (Add i to S)
6. else
7. $x'_i = 0$; (Add i to $V - S$)

Claim: $\sum_i x'_i ZN(x', i) \geq (1 - \frac{5\epsilon}{\alpha})OPT$ with high probability.
Proof:
Fact 1: With high proability, $ZN(x', i)$ is close to $ZN(y^*, i)$

$$E(ZN(x', i)) = \sum_{(i,j)\in E} E(1 - x'_j) \quad (14)$$

$$= \sum_{(i,j)\in E} (1 - y^*_j) \quad (15)$$

$$= ZN(y^*, i) \quad (16)$$

Thus with high proability, $ZN(x', i)$ is close to $ZN(y^*, i)$. Speficically,

$$\begin{aligned}
\Pr[ZN(x', i) &\leq (1-\delta)ZN(y^*, i)] && (17)\\
&\leq e^{-\frac{1}{2}ZN(y^*,i)\delta^2}) && (18)\\
&\leq e^{c\ln n} \quad (\text{ setting } \delta^2 = \min\{1, \tfrac{2c\ln n}{ZN(y^*,i)}\}) && (19)\\
&= n^{-c} && (20)
\end{aligned}$$

Fact 2: With high proability, $\sum_i x_i' Z_i$ is close to $\sum_i Z_i y_i^*$. Since $E(\sum_i x_i' Z_i) = \sum_i y_i^* Z_i$, we apply the Hoeffding bound to get $\Pr[\sum_i x_i' \frac{Z_i}{Z_{max}} \leq (1-\delta)\sum_i y_i^* \frac{Z_i}{Z_max}] \leq n^{-c}$, where

Analysis III

$\delta = \min\{1, \frac{2c\ln n}{\sum_i y_i^* \frac{Z_i}{Z_{max}}}\}$, $Z_{max} = \max_i\{Z_i\}$ to ensure $\frac{Z_i}{Z_{max}} \leq 1$.

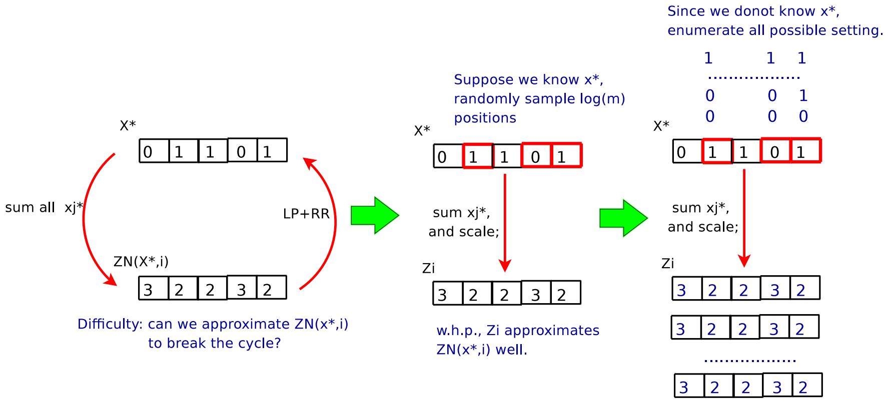
Thus with high probability, we have:

$$
\begin{aligned}
\sum_i x_i' Z_i &\geq (1-\delta)\sum_i Z_i y_i^* && (21)\\
&= (1-\min\{1, \frac{2c\ln n}{\sum_i y_i^* \frac{Z_i}{Z_{max}}}\})\sum_i Z_i y_i^* && (22)\\
&\geq \sum_i Z_i y_i^* - \sqrt{2Z_{max} c \ln n \sum_i y_i^* Z_i} && (23)\\
&\geq \sum_i Z_i y_i^* - n\sqrt{2cn\ln n} && (24)
\end{aligned}
$$

Fact 3:

$$
\begin{aligned}
\sum_i x'_i ZN(x', i) &\geq \sum_i x'_i (1-\delta) ZN(y^*, i) && (25)\\
&\geq \sum_i x'_i \left(ZN(y^*, i) - \sqrt{2c \ln n ZN(y^*, i)}\right) && (26)\\
&\geq \sum_i x'_i \left(Z_i - \epsilon n - \sqrt{2c \ln n ZN(y^*, i)}\right) && (27)\\
&\geq \sum_i x'_i Z_i - (\epsilon n + \sqrt{2cn \ln n}) \sum_i x'_i && (28)\\
&\geq \sum_i y^*_i Z_i - n\sqrt{2cn \ln n} - (\epsilon n + \sqrt{2cn \ln n}) \sum_i x'_i && (29)\\
&\geq \sum_i y^*_i Z_i - 2n\sqrt{2cn \ln n} - \epsilon n^2 && (30)\\
&\geq (1 - \tfrac{2\epsilon}{\alpha}) OPT - \tfrac{2\epsilon}{\alpha} OPT - o(1) OPT && (31)\\
&\geq (1 - \tfrac{5\epsilon}{\alpha}) OPT && (32)\\
& && (33)
\end{aligned}
$$

How to yield a good approximation Z_i? I



- Remaining difficulty: how to approximate $ZN(x^*, i)$ by Z_i?
- "Random sampling + enumeration" again!

1. Random sampling:
   Suppose x^* are known. Pick random subset S of $c \log n/\epsilon^2$ vertices. Set $Z_i = \frac{n}{|S|}\sum_{(i,j)\in E, j\in S}(1 - x*_j)$. The with high probability, we have: $ZN(x^*, i) - \epsilon n \leq Z_i \leq ZN(x^*, i) + \epsilon n$.
2. Enumerating: However, x^* are unknown. How to calculate Z_i? Enumerating all possible setting of x_j for $j \in S$. This will take poly-time.

A SDP can be formulated as:

$$\begin{array}{rrcl} \max & \sum c_{ij}x_{ij} & & \\ s.t. & \sum a_{ijk}x_{ij} & = & b_k \\ & X = (x_{ij}) \text{ is symmetric and SDP} & & \end{array}$$

Equivalent to vector programming:

$$\begin{array}{rrcl} \max & \sum c_{ij}(\vec{v_i} \bullet \vec{v_j}) & & \\ s.t. & \sum a_{ijk}(\vec{v_i} \bullet \vec{v_j}) & = & b_k \\ & \vec{v_i} \in R^n & & \end{array}$$

Reason: A PSD matrix X can be decomposed as $X = V^T V$ for some $V \in R^{m \times n}$. Note: SDP (and VP) can be solved in poly-time using the ellipsoid method or the interior-point technique.

A quadratic model of MaxCut:

$$\begin{array}{ll} \max & \frac{1}{2}\sum_{i<j} w_{ij}(1 - y_i y_j) \\ s.t. & y_i \in \{+1, -1\} \end{array}$$

A vector programming relaxation VP:

$$\begin{array}{ll} \max & \sum \frac{1}{2} w_{ij}(1 - \vec{v_i} \bullet \vec{v_j}) \\ s.t. & \vec{v_i} \bullet \vec{v_i} = 1 \\ & \vec{v_i} \in R^n \end{array}$$

Note: to see VP is a relaxation of the original quadratic model, we can view y_i as a 1-dimensional vector. Thus, any feasible solution to the quadratical model is also feasible to the VP model.
Implication: $Z_{VP} \geq OPT$.

- We can solve the vector programming model in poly-time.
- Question: how to convert the solution to VP to a solution to the quadratic model? Vector rounding!
    1. Solve vector programming problem to get vectors $\vec{v^*}$;
    2. Choose a random vector $\vec{r}$ uniformly from the unit n-sphere;
    3. $S = \Phi$;
    4. for i = 1 to n
    5.   add i into S iff $\vec{v_i^*} \bullet \vec{r} \geq 0$;

Theorem

VectorRounding is a 0.878-approximation algorithm.

Proof:

- Define random variables $x_{ij} = 1$ if $i \in S$ and $j \notin S$, or $i \notin S$ and $j \in S$;

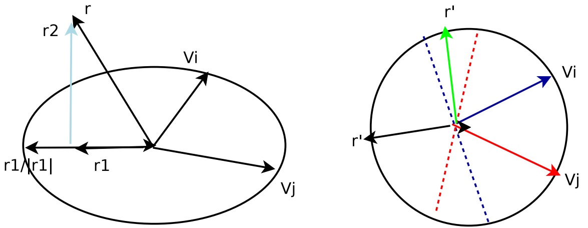
VectorRounding algorithm II

- and $W = \sum_{i<j} w_{ij} x_{ij}$;

$$
\begin{aligned}
E(W) &= \sum_{i<j} w_{ij} \Pr[i \in S \text{ and } j \notin S, \text{ or} i \notin S \text{ and } j \in S] && (34)\\
&= \sum_{i<j} w_{ij} \frac{1}{\pi} \arccos(\vec{v_i^*} \bullet \vec{v_j^*}) && (35)\\
&\geq 0.878 \frac{1}{2} \sum_{i<j} w_{ij} (1 - \vec{v_i^*} \bullet \vec{v_j^*}) && (36)\\
&= 0.878 Z_{VP} && (37)\\
&\geq 0.878 OPT && (38)
\end{aligned}
$$

Fact 1: Let $\vec{r'}$ be the projection of $\vec{r}$ onto a plane. $\frac{\vec{r'}}{||\vec{r'}||}$ is uniformaly distributed on a unit circle.

VectorRounding algorithm III



Fact 2:
$\Pr[i \in S$ and $j \notin S$, or $i \notin S$ and $j \in S] = \frac{1}{\pi}\arccos(\vec{v_i^*} \bullet \vec{v_j^*})$.
(Idea: consider the projection of $\vec{r}$ onto the plane spanned by $\vec{v_i^*}$ and $\vec{v_j^*}$. We have $\vec{v_i^*} \bullet \vec{r} = \vec{v_i^*} \bullet (\vec{r_1} + \vec{r_2}) = \vec{v_i^*} \bullet \vec{r_1})$)
Fact 3: $\frac{1}{\pi}\arccos(x) \geq 0.878\frac{1}{2}(1-x)$ for $-1 \leq x \leq 1$.

VectorRounding algorithm IV