236521 Approximation Algorithms	1st Semester, 1995–6
Lecture 4: 20.10.95	
Lecturer: Yuval Rabani	*Scribe: Yuri Miroshnik*

Randomized Rounding

1 MAX CUT problem

1.1 Definitions:

1.1.1 Cuts in graphs

A ***cut*** $(\boldsymbol{S}, \bar{\boldsymbol{S}})$ in the graph $G(V, E)$ is a vertex partitioning into two sets: S and $\bar{S}$ — such that $S \cup \bar{S} = V$. The edges cut by a cut $(S, \bar{S})$ are those connecting a vertex in S and a vertex in $\bar{S}$.

1.1.2 MAX CUT problem

Input : *$G(V, E)$ - indirected graph.*

Objective : *find a cut in the graph cutting the maximum number of edges.*

Remarks:

- The problem is $NP - hard$.
- Weights may be assigned to the edges.
- The MIN CUT problem has a polynomial solution.

1.2 A simple algorithm for MAX CUT:

Independently for every vertex v, it is chosen randomly with the probability $\frac{1}{2}$ whether v belongs to S.

1.3 Analysis of the algorithm:

Let's look at the edge e. What is the probability for it to be in the cut? The answer is $\frac{1}{2}$ — the probability for the two ends of the edge to be in the different sets: S and $\bar{S}$. Therefore the expectation of the cut size is $\frac{1}{2}|E|$. The maximum cut size is at most $|E|$, so we've got 2-approximation algorithm.

Conclusion: The analysis of the algorithm proves, that every graph has a cut with at least a half of its edges.

2 MAX SAT problem

2.1 Definition:

Input :

- CNF formula $\Psi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, consisted of:
- $C_1,\ C_2,\ \ldots,\ C_m$ — clauses, each clause is a sum of literals:
- $C_j = (x_{i_1} \vee \bar{x_{i_2}} \vee \ldots \vee x_{i_{k-1_j}} \vee \bar{x_{i_{k_j}}})$

Objective : *Find the substitution of the truth values for the variables, such that the number of true clauses is maximized.*

2.2 A simple algorithm for MAX SAT (by Johnson):

Independently for every variable x_i, it is chosen randomly with the probability $\frac{1}{2}$, whether the variable gets TRUE.

2.3 Analysis of Johnson's algorithm:

Let's look at the clause $C_j = (x_{i_1} \vee x_{i_2} \vee \ldots \vee x_{i_{k_j}})$. (Suppose without loss of generality, that there are no logical negations in it.) The probability that C_j gets TRUE is $1 - 2^{-k}$. Therefore the expectation of the clauses number receiving the TRUE is more than $\frac{m}{2}$, and thus this algorithm is 2-approximation of the MAX SAT problem.
If every clause has at least 2 variables, we get a $\frac{4}{3}$-approximation.
The problem is to find an algorithm working well when a great number of the clauses consist of only one literal. Here is a solution:

2.4 An algorithm for MAX SAT (by Goemans and Williamson):

For every clause C_j we match a variable $z_j \in \{0, 1\}$, for every variable x_j we match a variable $y_j \in \{0, 1\}$. Let's sign *a set of literals* appearing in C_j *without negation with* $\boldsymbol{S_j}^+$ and with $\boldsymbol{S_j}^-$ — *literals appearing with negation.*

2.4.1 The integral program

Let's define the integral program:

$$(IP) = max \sum_j z_j \ subject \ to$$

1. $\forall y \quad \sum_{i \in S_j^+} y_i \ + \sum_{i \in S_j^-} (1 - y_i) \ \geq \ z_j$

   (this rule says that if a clause C_j receives TRUE, there is at least one literal receiving TRUE in this clause).

2. $\forall i, j \ \ y_i, z_j \in \{0, 1\}$

2.4.2 The linear program

Let's pass to a linear program — relaxation of the integral one, which has the same form with the only change in the second set of constraints:

$$(LP) = max \sum_j x_j \ subject \ to$$

1. $\forall y \quad \sum_{i \in S_j^+} y_i \ + \sum_{i \in S_j^-} (1 - y_i) \ \geq \ z_j$

2. $\forall i, j \ \ y_i, z_j \in [0, 1]$

After solving this (LP) we'll obtain values $\bar{y_i}$ and $\bar{z_j}$ for every i and j.
Independently for every x_i, it's chosen randomly with the probability $\bar{y_i}$ whether x_i gets TRUE.

2.5 Analysis of this algorithm:

Let's look at a clause:

$$C_j = (x_1 \vee x_2 \vee \ldots \vee x_{k_j})$$

What's the probability this clause receives the truth value? Let's begin with the opposite event:

$$P_j = Pr[C_j \ unsatisfied] = \prod_{i=1}^{k_j} (1 - \bar{y_j})$$

According to the constraint 1:

$$\bar{y_1} + \bar{y_2} + \ldots + \bar{y_{k_j}} \ \geq \ \bar{z_j}$$

So,

$$P_j \;\le\; \prod_{i=1}^{k_j}(1-\frac{\bar{z}_j}{k_j}) = (1-\frac{\bar{z}_j}{k_j})^{k_j}$$

Thus, the wanted probability is

$$1-P_j \;\ge\; (1-\frac{\bar{z}_j}{k_j})^{k_j} \overset{home\ exercise}{\ge} (1-(1-\frac{1}{k_j})^{k_j})\bar{z}_j \quad (because\ \bar{z}_j \in [0,1]).$$

The expectation:

$$E = \sum_j (1-(1-\frac{1}{k_j})^{k_j})\bar{z}_j \;\ge\; (1-\frac{1}{e})\sum_j \bar{z}_j$$

According to the relaxation definition:

$$\sum_j \bar{z}_j \;\ge\; OPT = (IP)$$

Therefore, this is $\frac{1}{1-\frac{1}{e}} \;\ge\; \frac{1}{2}$-approximation.

2.6 The Goemans and Williamson algorithm: Integration of Johnson's algorithm and the above

Let's sign by n_J — expectation of Johnson's algorithm solution and
by n_{LP} — expectation of the linear programming algorithm solution.

We'll run randomly one of the algorithms (with the probability $\frac{1}{2}$) and the expectation of this integration will be:

$$E = \frac{n_J + n_{LP}}{2}$$

2.7 Analysis of the integration algorithm

The following table shows the number of the satisfied clauses of the different sizes in both algorithms, relatively to the linear programming algorithm's solution. As the linear programming algorithm is a relaxation of the (IP), we can evaluate the quality of the integration one.

	Johnson	linear programming	E
Clause of the size 1	$\geq \frac{1}{2} * \sharp clauses \geq \frac{1}{2} * weight$ of satisfied clauses in LP	weight of satisfied clauses in LP $= 1$	$\geq \frac{3}{4} n_{LP}$
Clause of the size 2	$\geq \frac{3}{4} * \sharp clauses \geq$ weight of satisfied clauses in LP	weight of satisfied clauses in LP $= \frac{3}{4}$	$\geq \frac{3}{4} n_{LP}$
...	...	...	...

Johnson's algorithm has a weakness with the 1-literal clauses, and the LP one repairs it; in its turn the LP algorithm is less successfull in satisfying long clauses where Johnson's one is good.

Conclusion: Expectation of the satisfied clauses number is more than $\frac{3}{4} \sum_j \bar{z}_j \geq \frac{3}{4} OPT$, so integration of the two algorithms gives $\frac{4}{3}$-approximation.

3 A routing problem

3.1 Definition:

Input : *Given graph $G(V, E)$ and a list of the vertex pairs (s_1, t_1), (s_2, t_2), $\ldots$, (s_k, t_k).*

Objective : *Find the paths connecting s_i to t_i for every i, such that the maximal number of the paths passing through an edge is minimal.*

3.2 Solution of the routing problem

3.2.1 The integral program

The integral program is:

$$OPT = min\ w\ s.t.$$

1. $\forall_i \sum x_i^j = 1$
2. $\forall_i \sum_j x_i^j q_i^j(e) \leq w$
3. $\forall_{i,j}\ x_i^j \in \{0, 1\}$

Descriptions to the integer program:

- For every pair (s_i, t_i) there are variables x_i^j, j goes over all possible paths from s_i to t_j, thus we have an exponential number of the x_i^j variables.
- Variables $q_i^j(e)$ are the indicators whether an edge e is on the path j from s_i to t_j.

3.2.2 The linear program and randomized rounding

Let's pass to the linear program:

1. $LP = min\ w\ s.t.$
2. $\forall_i\ \sum x_i^j = 1$
3. $\forall_i\ \sum_j x_i^j q_i^j(e)\ \leq\ w$
4. $\forall_{i,j}\ x_i^j \in [0,1]$

After getting a solution of this program: $\bar{w}$ and $\bar{x_i^j}$ — such that the number of nonzero solutions $\bar{x_i^j}$ is polynomial, for every pair (s_i, j_i) choose randomly the only path j basing upon the probabilities $\bar{x_i^j}$.

3.3 Analysis of the algorithm

It's known that $\bar{w}\ \leq\ OPT - IP$, since the LP is a relaxation of the IP. We have to find $A\ \leq\ \alpha * \bar{w}$, obtained obtained after choosing the only path for every pair (s,t), to evaluate the quality of the solution.
We get $A\ \leq\ \bar{w}(1+\delta)$ with a great probability such that δ depends on $\bar{w}$:

if $\bar{w}\ =\ O(\log n)$, then $\delta\ =\ O(1)$;
if $\bar{w}\ =\ 1$, then $\delta\ =\ O(\frac{\ln n}{\ln\ln n})$ can be taken.