# Randomized and Approximation Algorithms

## Lecture Algorithm Design

Stefano Leonardi, Rebecca Reiffenhäuser

December 04, 2020

## Part 1

LP-relaxation and randomized rounding for the weighted Max-SAT problem

# The weighted Max-SAT problem

Given a collection $C$ of clauses in disjunctive form,

$$C = [C - 1, C_2, \ldots, C_t]$$

and weights $w_i$ denoting the weight of clause $C_i$, with

$$C_i = \mathcal{C}_{i1} \vee \mathcal{C}_{i2} \vee \cdots \vee \mathcal{C}_{ik_i}$$

and $n$ boolean variables s.t.

$$\mathcal{C}_{ij} \in \{x_1, \overline{x_1}, x_2, \overline{x_2}, \ldots, x_n, \overline{x_n}\}$$

**Goal:** Find an assignment of $x_1, \ldots, x_n \in \{0, 1\}^n$ to maximize the sum of the weights of all satisfied clauses.

# The weighted Max-SAT problem

Previously in the lecture:

- Write problem as a linear program
- Round the fractional solution to yield a feasible one that is an approximation to the (integral) optimum

**Today:**
Do the same for weighted Max-SAT, but with **randomized rounding step**!

# The weighted Max-SAT problem

Previously in the lecture:

- Write problem as a linear program
- Round the fractional solution to yield a feasible one that is an approximation to the (integral) optimum

**Today:**
Do the same for weighted Max-SAT, but with **randomized rounding step**!

$\rightarrow$ The value of the objective function obtained by the algorithm is a **random variable**:

$$\mathbb{E}[ALG(I)] \geq \frac{1}{2} OPT(I) \ \forall I$$

# The weighted Max-SAT problem

Previously in the lecture:

- Write problem as a linear program
- Round the fractional solution to yield a feasible one that is an approximation to the (integral) optimum

**Today:**
Do the same for weighted Max-SAT, but with **randomized rounding step**!

$\rightarrow$ The value of the objective function obtained by the algorithm is a **random variable**:

$$\mathbb{E}[ALG(I)] \geq \frac{1}{2} OPT(I) \ \forall I$$

$\rightarrow$ Derandomization possible?

# A 2-approximative algorithm

[Johnson, 1974]

Set $x_j$ to 1 with probability $1/2$, and to 0 otherwise.

$C_i$ is satisfied with probability

$$\geq 1 - (\frac{1}{2})^{k_i} \geq \frac{1}{2}$$

$$\mathbb{E}[ALG] \geq \frac{1}{2} \sum_{i=1}^{t} w_i \geq \frac{1}{2} OPT$$

(since the sum of all weights is an upper bound for $OPT$)

Note: The algorithm is easy to derandomize

# Better approximation ratio

[Yannakakis, 1992]

4/3-approximation

Clauses with only one literal are treated differently!

# A 4/3-approximative, LP-based algorithm

[Goemans, Williamson, 1993]

Write problem as an LP (form: $Ax \leq b$).
Then, the $x_i$ depict each literal.
The solution $x^*$ of the LP can be interpreted as a set of probabilities:

$$x_1^*, x_2^*, \ldots, x_n^*$$

where instead of '$x_r$ is 20 percent true', we say '$x_r$ is true with a probability of $1/5$':

Set $x_j$ to 1 with prob. $x_j^*$, and 0 with prob. $(1 - x_j^*)$.

# A 4/3-approximative, LP-based algorithm

Define

$$T_i = \text{set of boolean variables that occur } \textbf{unnegated} \text{ in } C_i$$
$$F_i = \text{set of boolean variables that occur } \textbf{negated} \text{ in } C_i$$
$$z_i = 1 \text{ if } C_i \text{ is } \textbf{satisfied}, 0 \text{ otherwise.}$$

**Problem formulation:**

$$\max \sum_{i=i}^{t} z_i w_i$$

such that

$$\sum_{j \in T_i} x_j + \sum_{j \in F_i} (1 - x_j) \geq z_i$$

where $x_j, z_i \in \{0, 1\}$, $j = 1, \ldots, n$ and $i = 1, \ldots, t$.

# Rounding scheme

Let $x^*, z^*$ be the optimal solution of the relaxed LP, with

$$x_j \geq 0 \text{ and } z_i \leq 1$$

Set

$$\begin{aligned} x_j &= 1 && \text{with prob. } x_j^* \\ x_j &= 0 && \text{with prob. } (1 - x_j^*) \end{aligned}$$

**Lemma**

*It holds that*

$$\begin{aligned} P[C_i \text{ is satisfied}] &\geq 1 - \Pi_{j \in T_i}(1 - x_j^*)\Pi_{j \in F_i} x_j^* \\ &\geq \alpha_{k_i} z_i^* \end{aligned}$$

*with* $\alpha_{k_i} = 1 - (1 - \frac{1}{k_i})^{k_i}$.

# Rounding scheme

**Proof:** Assume every variable in $C = \mathcal{C}_{\rangle i1} \vee \cdots \vee \mathcal{C}_{ik_i}$ is unnegated; otherwise substitute $\overline{x_j}$ by $x_j$ and $x_j$ by $\overline{x_j}$ in any clause.

$$\begin{aligned}
P[C_i \text{ satisfied}] &\geq 1 - \Pi_{j=1}^{k_i}(1 - x_j^*) \\
&\geq 1 - \left( \frac{\sum_{j=1}^{k_i}(1 - x_j^*)}{k_i} \right)^{k_i} \\
&\geq 1 - \left( 1 - \frac{\sum_{j=1}^{k_i} x_j^*}{k_i} \right)^{k_i} \\
&\geq 1 - \left( 1 - \frac{z_i^*}{k_i} \right)^{k_i} \\
&\geq \left( 1 - \left( 1 - \frac{1}{k_i} \right)^{k_i} \right) z_i^* \\
&= \alpha_{k_i} z_i^*
\end{aligned}$$

## $\frac{e}{e-1} \approx 1.57$-approximative algorithm

$$
\begin{aligned}
\mathbb{E}[ALG] &\geq \sum_{i=1}^{t} \left(1 - \left(1 - \frac{1}{k_i}\right)^{k_i}\right) w_i z_i^* \\
&\geq \left(1 - \frac{1}{e}\right) \sum_{i=1}^{t} w_i z_i^* \\
&\geq \left(\frac{e-1}{e}\right) OPT
\end{aligned}
$$

since $\left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}$.

**The LP-relaxation gives an upper bound better than $\sum_{i=1}^{k} w_i$!**

# 4/3-approximation algorithm

Johnson:

$$\mathbb{E}[ALG] \geq \sum_{i=1}^{t} \left(1 - \frac{1}{2^{k_i}}\right) w_i z_i^*$$

Goemans/Williamson:

$$\mathbb{E}[ALG] \geq \sum_{i=1}^{t} \left(1 - \left(1 - \frac{1}{k_i}\right)^{k_i}\right) w_i z_i^*$$

**Johnson:** Good for large $k_i$
**Goemans/Williamson:** Good for small $k_i$

# 4/3-approximation algorithm

**Strategy:** Combine the two algorithms with disjoint 'bad' cases!

- Run Johnson with prob. 1/2
- Run G&W with prob. 1/2

**Lemma:** This algorithm is a 4/3-approximation.

# 4/3-approximation algorithm

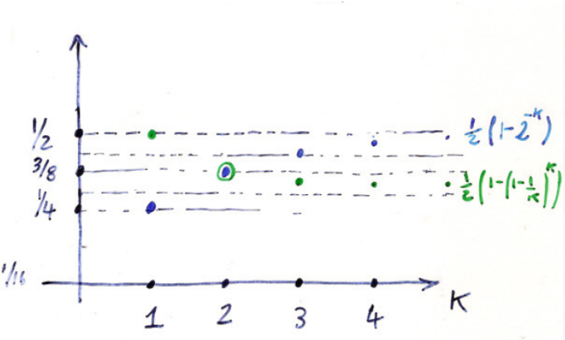
**Proof:**

$$P[C_i \text{ satisfied}] \geq \frac{1}{2}\left[\left(1 - 2^{-k_i}\right) + \left(1 - \left(1 - \frac{1}{k_i}\right)^{k_i}\right)\right] z_i^* \geq \frac{3}{4} z_i^*$$

# 4/3-approximation algorithm

**Proof:**

$$P[C_i \text{ satisfied}] \geq \frac{1}{2}\left[\left(1-2^{-k_i}\right)+\left(1-\left(1-\frac{1}{k_i}\right)^{k_i}\right)\right] z_i^* \geq \frac{3}{4} z_i^*$$




$$\mathbb{E}[ALG] \geq \frac{3}{4}\sum_{i=1}^{t} w_i z_i^* \geq \frac{3}{4} OPT$$

# Part 2

## Approximation algorithms for the Max-Cut problem

# The Max-Weighted-Cut problem

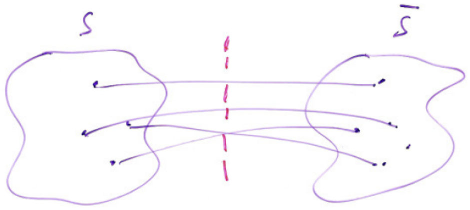
Given a graph $G = (V, E)$ and weights $w : E \to \mathbb{R}^+$, find a partition $(S, \overline{S})$ of $V$ that maximizes

$$\sum_{(i,j) \in E \text{ s.t. } i \in S, j \in \overline{S}} w_{(i,j)}$$

# The Max-Weighted-Cut problem

Given a graph $G = (V, E)$ and weights $w : E \to \mathbb{R}^+$, find a partition $(S, \overline{S})$ of $V$ that maximizes

$$\sum_{(i,j)\in E \text{ s.t. } i\in S, j\in \overline{S}} w_{(i,j)}$$



# A 2-approximative algorithm

**Algorithm:**

$$\begin{aligned} &\text{Assign } i \text{ to } S && \text{with prob. } \frac{1}{2} \\ &\text{Assign } i \text{ to } \overline{S} && \text{with prob. } \frac{1}{2} \end{aligned}$$

**Lemma:**
The algorithm is a 2-approximation.

# A 2-approximative algorithm

**Lemma:**
The algorithm is a 2-approximation.

**Proof:**

$$P\left[\left(i \in S \wedge j \in \overline{S}\right) \vee \left(i \in \overline{S} \wedge j \in S\right)\right] = \frac{1}{2}$$

(each single edge is in the cut with probability 1/2.)

$$\mathbb{E}[ALG] = \frac{1}{2} \sum_{(i,j) \in E} w_{(i,j)} \geq \frac{1}{2} OPT$$

(no more than all of the edges can be in the cut.) $\square$

## Better bounds

- We used a very pessimistic bound on $OPT$: $\sum_{(i,j) \in E} w_{(i,j)}$.
- A better $OPT$-bound can be achieved via quadratic programming
- This allows to prove even a 0.8785-approximation [Goemans, Williamson, 1994]

# Derandomization of the Max-Cut algorithm

**Goal:** Find a deterministic version of the algorithm that keeps the approximation ratio!

We define

$$
\begin{array}{lcl}
x_i = 1 & \text{IF} & v_i \in A \\
x_i = 0 & \text{IF} & v_i \in B
\end{array}
$$

$W$: An instance of the Max-Cut problem
$\mathbb{E}[W]$: Expected value of a random assignment

# Derandomization of the Max-Cut algorithm

**Strategy:**
Given all previous assignments for $x_1, \ldots, x_i$, choose for $x_{i+1}$ the set that maximizes the expectation when the rest of the choices is made at random!

$$\underbrace{\mathbb{E}\left[W \mid x_1 = a_1, \ldots, x_i = a_i\right]}_{(1)} = \frac{1}{2}\,\mathbb{E}\underbrace{\left[W \mid x_1 = a_1, \ldots, x_i = a_i, x_{i+1} = 1\right]}_{(2)} + \frac{1}{2}\,\mathbb{E}\underbrace{\left[W \mid x_1 = a_1, \ldots, x_i = a_i, x_{i+1} = 0\right]}_{(3)}$$

# Derandomization of the Max-Cut algorithm

**Strategy:**
Given all previous assignments for $x_1, \dots, x_i$, choose for $x_{i+1}$ the set that maximizes the expectation when the rest of the choices is made at random!

$$\underbrace{\mathbb{E}\left[W | x_1 = a_1, \dots, x_i = a_i\right]}_{(1)} =$$

$$\frac{1}{2}\,\underbrace{\mathbb{E}\left[W | x_1 = a_1, \dots, x_i = a_i, x_{i+1} = 1\right]}_{(2)}$$

$$+\frac{1}{2}\,\underbrace{\mathbb{E}\left[W | x_1 = a_1, \dots, x_i = a_i, x_{i+1} = 0\right]}_{(3)}$$

(2) $\geq$ (3) IF $w(v_{i+1}, B) \geq w(v_{i+1}, A)$.

# Derandomization of the Max-Cut algorithm

**Strategy:**
Given all previous assignments for $x_1, \ldots, x_i$, choose for $x_{i+1}$ the set that maximizes the expectation when the rest of the choices is made at random!

$$\underbrace{\mathbb{E}\left[W | x_1 = a_1, \ldots, x_i = a_i\right]}_{(1)} =$$

$$\frac{1}{2}\underbrace{\mathbb{E}\left[W | x_1 = a_1, \ldots, x_i = a_i, x_{i+1} = 1\right]}_{(2)}$$

$$+\frac{1}{2}\underbrace{\mathbb{E}\left[W | x_1 = a_1, \ldots, x_i = a_i, x_{i+1} = 0\right]}_{(3)}$$

$(2) \geq (3)$ IF $w(v_{i+1}, B) \geq w(v_{i+1}, A)$.
$(1) \leq (2)$ or $(1) \leq (3)$ must hold: Not all $(i+1)$-choices can be worse than a *random* $(i+1)$-choice!

# Derandomization of the Max-Cut algorithm

**Deterministic Algorithm:**
$A = \{v_1\}, \; B = \emptyset$.
**FOR** $i = 2, \ldots, n$:
Add $x_i$ to the set that maximizes the expected cut-weight.

The derandomized algorithm preserves the oringinal approximation ratio of the randomized algorithm, i.e. 2.

This is called derandomization via *conditional expectation*.

# Derandomization of Max-Cut: Efficiency

We have to compute the expected weights for both choices for $x_{i+1}$ in *polynomial time*.
What happens in step $i+1$ (randomized version) is as follows:

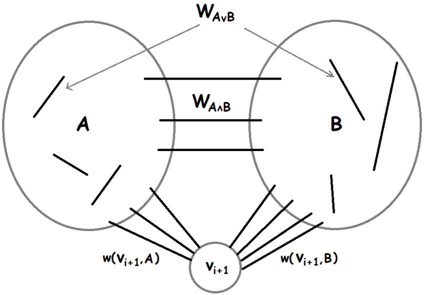
- Decisions on $x_1, \ldots, x_i$ are fixed
- For all $j = i+1, \ldots, n$:

$$\begin{array}{llll} \text{Add} & w(v_j, A) & \text{with probability} & \dfrac{1}{2} \\ \text{Add} & w(v_j, B) & \text{with probability} & \dfrac{1}{2} \end{array}$$

- For all $(v_j, v_k) \in E$, add $w(v_j, v_k)$ with probability $\frac{1}{2}$.

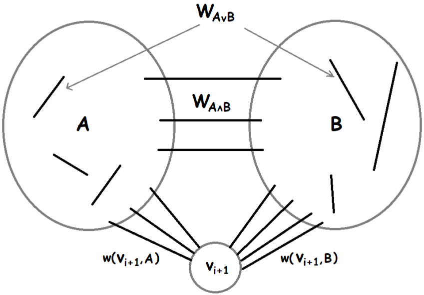
We replace the probabilities in the second point by choosing the larger one!

# Derandomization of Max-Cut: Efficiency




$$
\begin{aligned}
&\mathbb{E}\left[W \mid x_1 = a_1, \ldots, x_i = a_i, x_{i+1} = 1\right] \\
&\quad = W_{A \wedge B} + w(v_{i+1}, A) + \tfrac{1}{2} \textstyle\sum_{(v_j, v_k) \in E, j > i+1} w(e) \\
&\mathbb{E}\left[W \mid x_1 = a_1, \ldots, x_i = a_i, x_{i+1} = 0\right] \\
&\quad = W_{A \wedge B} + w(v_{i+1}, B) + \tfrac{1}{2} \textstyle\sum_{(v_j, v_k) \in E, j > i+1} w(e)
\end{aligned}
$$

# Derandomization of Max-Cut: Efficiency




The only difference in the expectations is having

$$w(v_{i+1}, A) \text{ OR } w(v_{i+1}, B)$$

Therefore, computing which is larger is easy!

# A greedy algorithm for Max-Cut

Let us define

$$w(v, A) \text{ for } A \subseteq V$$

as the total weight of edges from $v$ to vertices in $A$.

**Algorithm:**

$A = \{v_1\}$, $B = \{v_2\}$.
**FOR** $v \in V \setminus \{v_1, v_2\}$ **DO**:
  **IF** $w(v, A) \geq w(v, B)$
  **THEN** $B = B \cup \{v\}$
  **ELSE** $A = A \cup \{v\}$
Output $A$ and $B$ as $S$ and $\overline{S}$.

# A greedy algorithm for Max-Cut

**Lemma:**
The greedy algorithm is a 2-approximation.

**Proof:**
Vertices are considered in order $v_1, \ldots, v_n$. $w(v_1, v_2)$ is in the cut.
Let $w_<(v_i)$ be the total weight of edges from $v_i$ to $v_1, \ldots, v_{i-1}$.
At least $\frac{w_<(v_i)}{2}$ weight is separated by the cut.

$$ALG \geq \frac{1}{2} \sum_{i=2}^{n} w_<(v_i) \geq \frac{OPT}{2}$$

$\square$

# A greedy algorithm for Max-Cut

**Lemma:**

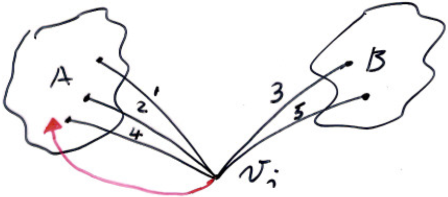
The greedy algorithm is a 2-approximation.

**Proof:**

Vertices are considered in order $v_1, \ldots, v_n$. $w(v_1, v_2)$ is in the cut.
Let $w_<(v_i)$ be the total weight of edges from $v_i$ to $v_1, \ldots, v_{i-1}$.
At least $\frac{w_<(v_i)}{2}$ weight is separated by the cut.

$$ALG \geq \frac{1}{2} \sum_{i=2}^{n} w_<(v_i) \geq \frac{OPT}{2}$$

$\square$



## Part 3

# Exercises

# An exercise: Multiway Cut

**Problem 3** (Approximation Algorithm for multiway cut)**.** We are given an undirected graph $G = (V, E)$, costs $c_e \geq 0$ for all edges $e \in E$, and $k$ distinguished vertices $s_1, ..., s_k$. The goal is to remove a minimum-cost set of edges $F$ such that no pair of distinguished vertices $s_i$ and $s_j$ for $i \neq j$ are in the same connected component of $(V, E - F)$.

We know that when $k = 2$ then the problem is just the min-cut problem, and it can be solved in polynomial time via max-flow. Consider the following algorithm for the problem with $k$ vertices. For $i = 1, ..., k$ let $F_i$ be the min-cut that separates vertex $s_i$ and vertices $s_1, s_2, ..., s_{i-1}, s_{i+1}, ..., s_k$. Solution $\bigcup_{i=1,...,k} F_i$ is obviously a feasible solution, i.e. it separates all $s_1, ..., s_k$. Show that it is also a 2-approximation of the optimal solution.