Introduction to Algorithms and Data Structures

Lecture 27: Dealing with NP-completeness (Approximation algorithms)

Mary Cryan

School of Informatics
University of Edinburgh

Implications of NP-complete status

> *When prove a problem is* NP-complete, *we no longer expect to be able to design polynomial-time algorithms to generate exact solutions (to the Decision problem or to an Optimization version)*

What are our options?

- Heuristic methods ("rules of thumb") that might not *guarantee* good results, but behave well in practice.
- Might there be a polynomial-time algorithm to search for an *approximate* solution rather than an exact one? (today)
- Brute-force methods that run in exponential-time (L28)
- Recursive backtracking (L28)

We might use randomness ... and may later want to de-randomize.

Mostly we are dealing with the optimization version of the decision problem.

What is an approximation algorithm?

For an *optimization problem*, we ask questions of the form

"For a given instance $\mathcal{I}$ of the problem, what is the value of a best solution y for that instance?

best will usually either be

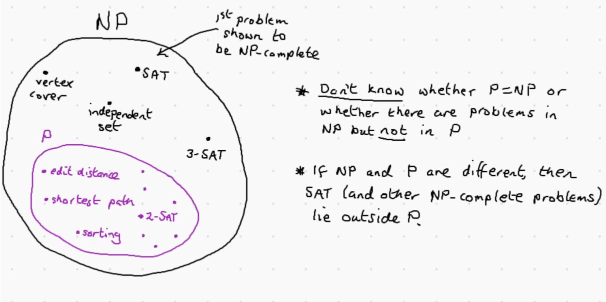
- max-valued (eg Ind.Set, solutions being Ind.Sets.) or
- min-valued (eg Edit distance, solutions being alignments).

Definition: Consider some optimization problem OPT where for a given instance $\mathcal{I}$, and the set of feasible solutions y, $\mathrm{OPT}(\mathcal{I})$ is the cost/value of the optimum y.

An algorithm A is said to be an α-approximation algorithm for OPT if for every instance $\mathcal{I}$, the algorithm returns a value $A(\mathcal{I})$ satisfying

$$A(\mathcal{I}) \begin{cases} \leq & \alpha \cdot \mathrm{OPT}(\mathrm{I}) & \text{if OPT is a minimization problem} \\ \geq & \alpha^{-1} \cdot \mathrm{OPT}(\mathrm{I}) & \text{if OPT is a maximization problem} \end{cases}$$

Polynomial-time approximation: examples



Vertex Cover Will see a simple algorithm which gives a 2-approximation.

MAX 3-SAT Will see a randomized algorithm (and a derandomization) that gives a $\frac{8}{7}$-approximation for satisfying a max number of clauses.

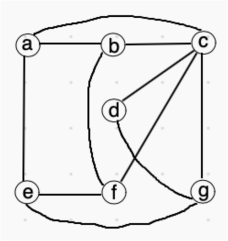
note: The α-value is called the approximation ratio of the algorithm.

Vertex Cover (minimization)

Definition

Given an undirected graph $G = (V, E)$, a subset $V' \subseteq V$ is a Vertex Cover (VC) for G if every edge $e \in E$ has at least one endpoint in V'.

VERTEX COVER: Determine the size of the minimum cardinality VC for G.



Optimum VC has size 4 (c, 1 of $\{d, g\}$, 2 non-adjacent vertices of $\{a, b, f, e\}$).

2-approximation for Vertex Cover

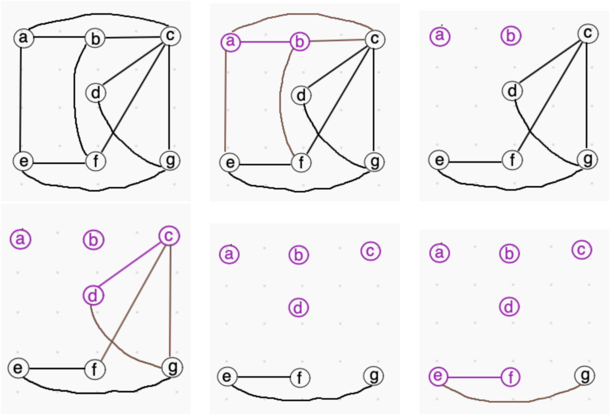
The decision version of VERTEX COVER (is the minimum VC of size $\leq k$) is NP-complete $\Rightarrow$ we do not believe that the optimization VERTEX COVER problem can have a polynomial-time algorithm.

Here is an *approximation algorithm*:

Algorithm Approx-Vertex-Cover($G = (V, E)$)

1. $C \leftarrow \emptyset$
2. $E' \leftarrow E$
3. **while** $E' \neq \emptyset$
4.   **do** take any edge $(u, v) \in E'$
5. $C \leftarrow C \cup \{u, v\}$ // add **both** u and v to the cover
6.   Remove every edge g with u or v endpoint from E'
7. Print("There is a VC of size ", $|C|$)

2-approximation for Vertex Cover: example



We end up with a VC of size 6.

2-approximation for Vertex Cover

Algorithm Approx-Vertex-Cover($G = (V, E)$)

1. $C \leftarrow \emptyset$
2. $E' \leftarrow E$
3. **while** $E' \neq \emptyset$
4.   **do** take any edge $(u, v) \in E'$
5. $C \leftarrow C \cup \{u, v\}$ // add **both** u and v to the cover
6.   Remove every edge g with u or v endpoint from E'

Why a 2-approximation?

- Compare to some unknown minimum vertex cover C^*.
- Consider the set F of all edges chosen by line 4 (the "purple edges").
  - Each $f \in F$ must have *one endpoint* in C^*.
    For $f, f' \in F, f \neq f'$, f and f' share *no* endpoints. So $|C^*| \geq |F|$.
  - The alg chooses each f to have no overlapping endpoints with the earlier purple edges. So $|C| = 2|F|$. Hence $|C| \leq 2|C^*|$.

Optimization/search problem MAX 3-SAT

MAX 3-SAT: Given a 3-CNF formula $\phi = C_1 \wedge \ldots \wedge C_m$ over the variables $\{x_1, \ldots, x_n\}$, determine the maximum number of clauses k such that there is an assignment of binary values to $\{x_1, \ldots, x_n\}$ that makes k clauses satisfied.

3-SAT is NP-complete $\Rightarrow$ we do not expect a polynomial-time algorithm to exactly solve the MAX 3-SAT problem. Why?

We will design an algorithm to find an assignment satisfying $\geq \frac{7}{8} \cdot m$ clauses.

- Will show we are *guaranteed* there is some assignment satisfying $\geq \frac{7}{8} \cdot m$ clauses $\Rightarrow$ get an $\frac{8}{7}$-approximation algorithm for MAX 3-SAT.
- In fact, for optimizing the *value*, we could just output $\frac{7}{8} \cdot m$ without checking anything ... this would already be a $\frac{8}{7}$-approximation algorithm.
- Search problem: find an assignment to satisfy a high number of clauses.
- We assume/require a 3-CNF formula where each clause C_j has *exactly* 3 literals (on different logical variables). Need this condition.

Uniform Random Assignment and MAX 3-SAT

We are given $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, and each of the C_j is $(\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$ for three *literals* over $\{x_1, \ldots, x_n\}$ (for example, $(x_4 \vee \bar{x_1} \vee \bar{x_9})$).

want: *assignment* to the $\{x_1, \ldots, x_n\}$ which satisfies a high number of clauses.

Consider a single clause $C_j = (\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$

- C_j is satisfied if *at least one* of its literals are satisfied: $\ell_{j,1}$ or $\ell_{j,2}$, or $\ell_{j,3}$
- C_j will fail to be satisfied *only if all* its literals are False.
- Suppose we generate a uniform random assignment (uar) to the logical variables $x_1, \ldots, x_n$ (each x_i gets 0/1 with probability $\frac{1}{2}$).
  - The probability that C_j is *not* satisfied is exactly $\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2}$, ie, $\frac{1}{8}$.
  - This is where we use the assumption that each clause has 3 literals over different x_i.
  - The probability that C_j *is* satisfied is exactly $\frac{7}{8}$.

The expected number of clauses of ϕ satisfied by a uar assignment is $\frac{7}{8}m$.

Uniform Random Assignment and Max 3-SAT

$$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

Let Y_j be the $0/1$ random variable that is 1 if C_j is satisfied.

Then $Y = \sum_{j=1}^{m} Y_j$ is the number of clauses that are satisfied by the assignment. The expected number of clauses $\mathrm{E}[Y]$ satisfied by a uar assignment is $\frac{7}{8}m$ (by *linearity of expectation*)

- Must be *at least one* assignment to $\mathbf{x} = x_1 \ldots x_n$ which satisfies $\geq \frac{7}{8}m$ clauses.
  - If *all* assignments to the variables satisfied $< \frac{7}{8}m$ clauses, the *expected* number could not be $\frac{7}{8}m$.(under uar random assignment, the *expectation* is the *average* over all $\{0,1\}^n$).
- Naïve (randomized) algorithm: generate a random assignment $b \in \{0,1\}^n$ until we achieve $\geq \frac{7}{8}m$ satisfied clauses.

De-randomized $\geq \frac{7}{8}m$ algorithm for MAX 3-SAT

$$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

$Y = \sum_{j=1}^{m} Y_j$ the number of clauses that are satisfied by the uar assignment.

$$\begin{aligned}
\mathrm{E}[Y] &= \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} \sum_{j=1}^{m} Y_j \\
&= \frac{1}{2^{n-1}} \sum_{\mathbf{x} \in \{0,1\}^{n-1}} \sum_{j=1}^{m} \left(\frac{(Y_j | x_1 = 0)}{2} + \frac{(Y_j | x_1 = 1)}{2} \right) \\
&= \frac{\mathrm{E}[Y | x_1 = 0]}{2} + \frac{\mathrm{E}[Y | x_1 = 1]}{2}
\end{aligned}$$

Observe: If $\mathrm{E}[Y] \geq \frac{7}{8}m$, then *either* $\mathrm{E}[Y \mid x_1 = 0] \geq \frac{7}{8}m$ or $\mathrm{E}[Y \mid x_1 = 1] \geq \frac{7}{8}m$.

Idea: Compute $\mathrm{E}[Y \mid x_1 = 0]$ and $\mathrm{E}[Y \mid x_1 = 1]$ (how?), compare values, fix x_1 to have that maximizing binary value ... iterate.

De-randomized $\geq \frac{7}{8}m$ algorithm for MAX 3-SAT

"Method of conditional expectations"

Algorithm Greedy-3-SAT(ϕ, n, m)

for $i = 1, \ldots, n$
    Compute $Exp_0 \leftarrow \mathrm{E}[Y \mid x_1 = b_1 \ldots x_{i-1} = b_{i-1}, x_i = 0]$
    Compute $Exp_1 \leftarrow \mathrm{E}[Y \mid x_1 = b_1 \ldots x_{i-1} = b_{i-1}, x_i = 1]$
    if $Exp_0 \geq Exp_1$
        then $b_i \leftarrow 0$; Update ϕ by fixing $x_i = 0$
        else $b_i \leftarrow 1$; Update ϕ by fixing $x_i = 1$
return $\mathbf{b}$

Computing the Exp_0, Exp_1 values:

- After fixing $b_1, \ldots, b_{i-1}$, $\phi(b_1, \ldots, b_{i-1})$ will have some already satisfied clauses, some already failed clauses, and some unresolved clauses of lengths 1, 2, and 3.
- Evaluate Exp_0 by setting $x_i = 0$. For any C_j with $\bar{x}_i \in C_j$, $\mathrm{E}[Y_j \mid \ldots x_i = 0] = 1$. For any C_j with $x_i \in C_j$, $\mathrm{E}[Y_j \mid \ldots]$ drops from $\frac{7}{8} \rightarrow \frac{3}{4}$, from $\frac{3}{4} \rightarrow \frac{1}{2}$, or $\frac{1}{2} \rightarrow 0$.
- Evaluation of Exp_1 is symmetric.

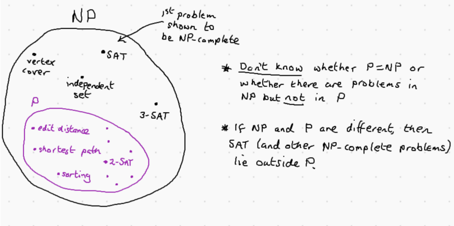
De-randomized $\geq \frac{7}{8}m$ algorithm for MAX 3-SAT

At every iteration of Greedy-3-SAT, we have

$$\mathrm{E}[Y \mid x_1 = b_1 \dots x_{i-1} = b_{i-1}] \quad = \quad \frac{\mathrm{E}[Y|x_1=b_1 \dots x_{i-1}=b_{i-1} x_1=0]}{2} + \frac{\mathrm{E}[Y|x_1=b_1 \dots x_{i-1}=b_{i-1} x_1=1]}{2}$$

- At every iteration we assign x_i to have the maximizing binary value b_i.
- We eventually obtain a assignment $\mathbf{x} \leftarrow \mathbf{b} \in \{0,1\}^n$ that is guaranteed to satisfy $\geq \frac{7}{8}m$ clauses. This is deterministic, randomness has been removed.
- $\frac{8}{7}$-approximation for the MAX 3-SAT search problem.
- Greedy-3-SAT does n iterations, doing $O(1)$ work for each clause C_j on that iteration $\Rightarrow O(n \cdot m)$ running-time.
- In 1997, Johan Håstad proved that if P $\neq$ NP, then no polynomial-time algorithm can guarantee a better approximation ratio for MAX 3-SAT.

Wrapping up



Vertex Cover 2-approximation in polynomial-time.

MAX 3-SAT $\frac{8}{7}$-approximation (best possible) in polynomial-time.

Ind.Set. The MAX-IND-SET problem cannot be approximated to any constant approximation factor α (assuming $\mathrm{P} \neq \mathrm{NP}$).

All NP-complete problems can be reduced to one another. However, the reductions typically do not preserve approximation ratios.

Reading and Working

Reading:

- CLRS Section 35 "intro", 35.1 (Vertex Cover), 35.4 (Max 3-SAT)
- KT Alternatively, you can read 13.3, 13.3 of Kleinberg and Tardos.

Working: Run Greedy-3-SAT on the following formula:

$$\begin{aligned}\Phi \quad = \quad & (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (x_1 \vee x_2 \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (\bar{x_1} \vee x_2 \vee x_3) \\ & \wedge (x_1 \vee x_2 \vee \bar{x_4}) \wedge (x_1 \vee \bar{x_2} \vee x_4) \wedge (x_2 \vee \bar{x_3} \vee \bar{x_4}).\end{aligned}$$