Notes, 8(c)

ECE 606

Efficient Approximation

As we will soon study, there exist discrete optimization problems of interest for which it is unlikely that a polynomial-time algorithm exists. For such problems, instead of seeking an algorithm that is guaranteed to output a solution that is correct, what if we ease our expectation to only require an algorithm whose output is guaranteed to be a "good enough" approximate of the correct solution?

It turns out that there indeed exist such problems. The intent of this portion of the lecture and course is to look at a few examples of: (i) meaningful notions of "good enough" approximations, and, (ii) algorithms that guarantee those in polynomial-time.

Example: minimum-sized vertex cover.

- Input: an undirected graph $G = \langle V, E \rangle$.
- Output: a subset of vertices, $R \subseteq V$, of smallest size, such that $\langle u, v \rangle \in E \implies (u \in R \vee v \in R)$.

  That is, given any edge in G, at least one of its endpoints is in R, and R must be a smallest set that has this property.

It turns out that a polynomial-time algorithm is unlikely to exist for the above problem. Indeed, the problem is **NP**-hard, which is something we will study soon. Consider the following efficient algorithm for this problem.

ApproxMinVertexCover($G = \langle V, E \rangle$)

$R \leftarrow \emptyset$
while $E \neq \emptyset$ **do**
    pick some $\langle u, v \rangle \in E$
    $R \leftarrow R \cup \{u, v\}$
    remove every edge incident on either u or v, or both, from E
return R

Claim 1. *Suppose r^* is the minimum size of a vertex cover for undirected $G = \langle V, E \rangle$. Then,* ApproxMinVertexCover *is guaranteed to output an R such that $r \leq 2r^*$, where $r = |R|$.*

Proof: in any run of the algorithm, for every edge $\langle u, v \rangle$ we pick in Line (3), we must include at least one of u or v in any vertex cover. We happen to include both in R, and so our final solution is at worst twice as bad as it could possibly be.

ApproxMinVertexCover is what is called a "2-approximation algorithm."

In general, assume that every instance of every optimization problem we consider has solution $r^* > 0$. Then, given an input of size n, we call an algorithm for an optimization problem a $\rho(n)$-approximation algorithm for that problem if we are guaranteed, given that the algorithm outputs a solution of size r, that:

$$\max\{r/r^*, r^*/r\} \leq \rho(n)$$

The reason we have both r/r^* and r^*/r is to account for both maximization and minimization problems. If our problem is a minimization problem, like vertex cover above, then r/r^* would be the result of the max, because no solution smaller than r^* can possibly be sound. Similarly, if we have a maximization algorithm, r^*/r would yield the max of the left-hand side.

In the case of the problem of determining a minimum-sized vertex cover, $\rho(n) = 2$, i.e., a constant in n, for the above algorithm. Which is actually pretty darn good.

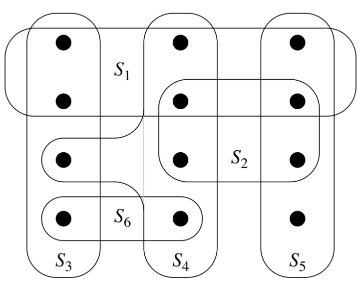
Your textbook includes also the example of the so-called travelling salesman problem with the triangle inequality, for which also there exists a 2-approximation algorithm.

Set Cover

An instance of the set-covering problem is a pair, $\langle \mathcal{X}, \mathcal{F} \rangle$, where $\mathcal{X}$ is a finite set, and $\mathcal{F}$ is a family of subsets of $\mathcal{X}$ such that every $x \in \mathcal{X}$ belongs to at least one subset in $\mathcal{F}$. That is: $\mathcal{X} = \bigcup_{S \in \mathcal{F}} S$.

The problem, given as input such an $\langle \mathcal{X}, \mathcal{F} \rangle$, is to find a minimum-size subset $\mathscr{C} \subseteq \mathcal{F}$ such that the union of members of $\mathscr{C}$ is $\mathcal{X}$, i.e., $\mathcal{X} = \bigcup_{S \in \mathscr{C}} S$.

An example from your textbook (CLRS), where the members of $\mathcal{X}$ are represented by black dots and $\mathcal{F} = \{S_1, S_2, \ldots, S_6\}$ is shown below. The union of the sets in the following subset of $\mathcal{F}$ contains all the members of $\mathcal{X}$ and is therefore a set cover for $\mathcal{X}$: $\{S_1, S_2, S_4, S_5, S_6\}$. However, it is not a minimum-size set cover. $\{S_3, S_4, S_5\}$ is a minimum-size set cover.



The set cover problem can legitimately be seen as a generalization of vertex cover. Given an undirected graph $G = \langle V, E \rangle$, we can map it to an instance of set cover $\langle \mathcal{X}, \mathcal{F} \rangle$ as follows. Let $\mathcal{X} = E$, $S_u \in \mathcal{F}$ if and only if $u \in V$, and $S_u = \{e \in E \mid u \text{ is incident on } e\}$.

Like vertex cover, it is unlikely that there exists a polynomial-time algorithm for minimum-size set cover. Consider the following greedy algorithm instead, which we propose as an approximation algorithm.

Greedy-Set-Cover($\mathcal{X}, \mathcal{F}$)

$\mathscr{C} \leftarrow \emptyset$
while $\mathcal{X} \neq \emptyset$ **do**
    select $S \in \mathcal{F}$ that maximizes $|S \cap \mathcal{X}|$
    $\mathcal{X} \leftarrow \mathcal{X} \setminus S$, $\mathscr{C} \leftarrow \mathscr{C} \cup S$
return $\mathscr{C}$

A sequence of choices the algorithm makes for the above example is S_1, S_4, S_5, S_3.

The analysis for the goodness of GREEDY-SET-COVER in your textbook (CLRS) relies on bounding the dth Harmonic number, $H(d) = \sum_{i=1}^{d} 1/i = 1/1 + 1/2 + 1/3 + \ldots + 1/d$.

Claim 2. *$H(n) = O(\lg n)$.*

Proof. We prove that for all $n \geq 1$, there exist positive constants c_1, c_2 such that $H_n \leq c_1 \cdot \log n + c_2$. Indeed, $c_1 = c_2 = 1$, and a base of 2 for the log should work.

For the base case, $n = 1$ and $H(n) = 1 \leq 1 = \log n + 1$.

For the step, it suffices to prove that $\log n + 1 + 1/(n+1) \leq \log(n+1) + 1$. We observe:

$$
\begin{aligned}
&\log n + 1 + \frac{1}{n+1} \leq \log(n+1) + 1 \\
&\qquad \Longleftarrow \log n + \frac{1}{n+1} \leq \log(n+1) \\
&\qquad \Longleftarrow n \cdot 2^{1/(n+1)} \leq n+1 \\
&\qquad \Longleftarrow 2^{1/(n+1)} \leq 1 + 1/n \\
&\qquad \Longleftarrow 2 \leq (1 + 1/n)^{n+1} \\
&\qquad \Longleftarrow 2 \leq \binom{n+1}{0} \cdot 1^{n+1} \cdot (1/n)^0 + \binom{n+1}{1} \cdot 1^n \cdot (1/n)^1 + \sum_{i=2}^{n+1} \left(\binom{n+1}{i} \cdot 1^{n+1-i} \cdot (1/n)^i \right) \\
&\qquad \Longleftarrow 2 \leq 1 + 1 + 1/n + \sum_{i=2}^{n+1} \left(\binom{n+1}{i} \cdot 1^{n+1-i} \cdot (1/n)^i \right)
\end{aligned}
$$

$\square$

Claim 3. Greedy-Set-Cover *is a* $H(\max\{|S| : S \in \mathcal{F}\})$*-approximation algorithm.*

A challenge in the proof is to *upper-bound* $|\mathscr{C}|$ with a multiple of $|\mathscr{C}^*|$, where $\mathscr{C}^*$ is a minium-size set cover.

Your textbook (CLRS) assigns a "cost" of 1 for every set $S \in \mathcal{F}$ that is added to $\mathscr{C}$ in Line (4) of the algorithm. And it "spreads" this cost equally across every item $x \in \mathcal{X}$ that is covered for the first time by S.

Thus, if $\overset{x}{\cancel{c_x}}$ is covered for the first in the ith iteration for the first time by including $S_i \in \mathcal{F}$, then:

$$c_x = \frac{1}{|S_i \setminus (S_1 \cup \ldots \cup S_{i-1})|}$$

And now:

$$\begin{aligned} |\mathscr{C}| &= \sum_{x \in \mathcal{X}} c_x \\ \sum_{S \in \mathscr{C}^*} \sum_{x \in S} c_x &\geq \sum_{x \in \mathcal{X}} c_x \\ \sum_{x \in S} c_x &\leq H(|S|) \qquad \text{— see textbook for proof} \end{aligned}$$

$$\begin{aligned} \text{So, } |\mathscr{C}| &\leq \sum_{S \in \mathscr{C}^*} H(|S|) \\ &\leq |\mathscr{C}^*| \cdot H(\max\{|S| : S \in \mathcal{F}\}) \end{aligned}$$