14: Approximation algorithms II

Sariel Har-Peled

July 28, 2006
CS 473g - Algorithms

Version: 0.1

1 Max Exact 3SAT

We remind the reader that an instance of the 3SAT problem looks like $F = (x_1 + x_2 + x_3)(x_4 + \overline{x}_1 + x_2)$. Interestingly, we can turn this into an optimization problem.

**Problem:** Max 3SAT

<table>
<thead>
<tr>
<th>Instance: A collection of clauses: $C_1, \ldots, C_m$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: Find the assignment to $x_1, \ldots, x_n$ that satisfies the maximum number of clauses.</td>
</tr>
</tbody>
</table>

Clearly, since 3SAT is NP-COMPLETE it implies that Max 3SAT is NP-HARD. In particular, the formula $F$ becomes the following set of two clauses:

$$x_1 + x_2 + x_3 \quad \text{and} \quad x_4 + \overline{x}_1 + x_2.$$ 

Note, that Max 3SAT is a maximization problem.

**Definition 1.1** Algorithm Alg for a maximization problem achieves an approximation factor $\alpha$ if for all inputs, we have:

$$\frac{\text{Alg}(G)}{\text{Opt}(G)} \geq \alpha.$$ 

**Theorem 1.2** There is an algorithm which achieves (in expectation) $(7/8)$-approximation in polynomial time. Namely, if the instance has $m$ clauses it satisfies $(7/8)m$.

**Proof:** Let $x_1, \ldots, x_n$ be the $n$ variables used in the given instance. The algorithm works by randomly assigning values to $x_1, \ldots, x_n$, independently, and equal probability to 0 and 1 for each one of the variables of the given instance.

Let $Y_i$ be the indicator variables which is 1 if $i$th clause is satisfied by the generated random assignment and 0 otherwise, for $i = 1, \ldots, n$.

We have that

$$Y_i = \begin{cases} 
1 & \text{if } C_i \text{ is satisfied by the random assignment} \\
0 & \text{otherwise.}
\end{cases}$$

---

*This work is licensed under the Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit [http://creativecommons.org/licenses/by-nc/2.5/](http://creativecommons.org/licenses/by-nc/2.5/) or, (b) send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.*
Clearly, the number of clauses satisfied by the given assignment is: $Y = \sum_{i=1}^{m} Y_i$. We claim that $E[Y] = (7/8) m$, where $m$ is the number of clauses in the input. Indeed, we have

$$E[Y] = E \left[ \sum_{i=1}^{m} Y_i \right] = \sum_{i=1}^{m} E[Y_i]$$

by linearity of expectation. Now, what is the probability that $Y_i = 0$?

$$\Pr[Y_i = 0] = \frac{1}{2} \times \frac{1}{2} = \frac{1}{8}$$

This is the probability that $C_i$ is not satisfied. $C_i$ is made out of exactly three literals, and as such....?

$$\Pr[Y_i = 1] = 1 - \Pr[Y_i = 0] = \frac{7}{8}.$$ 

Thus,

$$E[Y_i] = \Pr[Y_i = 0] \times 0 + \Pr[Y_i = 1] \times 1 = \frac{7}{8}.$$ 

Namely, $E[\#\text{ of clauses sat}] = E[Y] = \sum_{i=1}^{m} E[Y_i] = \frac{7}{8} m$. Since the optimal solution satisfies at most $m$ clauses, the claim follows.

\section{Approximation Algorithms for Set Cover}

\subsection{Guarding an Art Gallery}

You are given the floor plan of an art gallery, which is a two dimensional simple polygon. You would like to place guards that see the whole polygon. A guard is a point, which can see all points around it, but it can not see through walls. Formally, a point $p$ can see point $q$, if the segment $pq$ is contained inside the polygon. See example on the right for how the input looks like.

Such a \textit{visibility polygon} is depicted on the left, here the yellow polygon is the visibility polygon of $p$. The question is how many guards to we need to guard the given art-gallery? Namely, every guard can see in all directions, and we need to places guards that see all the points in the polygon.

The art-gallery is essentially a set-cover problem. It is known that finding the minimum number guards is \textsc{NP-Hard}. No approximation is currently known. It is known that a polygon with $n$ corners, can be guarded using $n/3 + 1$ guards. Note, that this problem is harder than the classical set-cover problem because the number of subsets is infinite and the underlining base set is also infinite.

An interesting \textit{open problem} is to find a polynomial time approximation algorithm, such that given $P$, it computes a set of guards, such that $\#\text{guards} \leq \sqrt{n}k_{opt}$, where $n$ is the number of vertices of the input polygon $P$, and $k_{opt}$ is the number of guards used by the optimal solution.
2.2 Set Cover

The optimization version of Set Cover, is the following:

**Problem: Set Cover**

<table>
<thead>
<tr>
<th>Instance: ((S, F))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S) - a set of (n) elements</td>
</tr>
<tr>
<td>(F) - a family of subsets of (S), s.t. (\bigcup_{X \in F} X = S).</td>
</tr>
<tr>
<td>Output: The set (X \subseteq F) such that (X) contains as few sets as possible, and (X) covers (S).</td>
</tr>
</tbody>
</table>

Note, that Set Cover is a minimization problem which is also NP-HARD.

**Example 2.1** Consider the set \(S = \{1, 2, 3, 4, 5\}\) and the following family of subsets

\[ F = \{\{1, 2, 3\}, \{2, 5\}, \{1, 4\}, \{4, 5\}\}. \]

Clearly, the smallest cover of \(S\) is \(X_{\text{opt}} = \{\{1, 2, 3\}, \{4, 5\}\}\).

The greedy algorithm for this problem is depicted on the right, and is clearly polynomial in the input size. Indeed, we are given a set \(S\) of \(n\) elements, and \(m\) subsets. As such, the input size is at most \(\Omega(m + n)\), and the algorithm can be performed in time polynomial in \(m\) and \(n\). Let \(X_{\text{opt}} = \{V_1, \ldots, V_k\}\) be the optimal solution.

Let \(T_i\) denote the elements not covered in the beginning \(i\)th iteration of GreedySetCover, where \(T_1 = S\). Let \(U_i\) be the set added to the cover in the \(i\)th iteration, and \(\alpha_i = |U_i \cap T_i|\) be the number of new elements being covered in the \(i\)th iteration.

**Claim 2.2** We have \(\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_k \geq \ldots \geq \alpha_m\).

**Proof:** If not \(\alpha_i < \alpha_{i+1}\) but then, \(U_{i+1}\) covers more elements than \(U_i\) and we can exchange between them. Contradicting the greediness of the algorithm in choosing the set covering the largest number of elements not covered yet.

**Claim 2.3** We have \(\alpha_i \geq |T_i|/k.\) Namely, \(|T_{i+1}| \leq (1 - 1/k)|T_i|\).

**Proof:** Consider the optimal solution. It is made out of \(k\) sets and it covers \(T_i\), as such one of those subsets cover at least \(1/k\) of the elements of \(T_i\). Finally, the greedy algorithm picks the set that covers the largest number of elements of \(T_i\). Thus, \(U_i\) covers at least \(|T_i|/k\) elements.

As for the second claim, we have that \(|T_{i+1}| = |T_i| - \alpha_i \leq (1 - 1/k)|T_i|\).

**Theorem 2.4** The algorithm GreedySetCover generates a cover of \(S\) using at most \(O(k \log n)\) sets of \(F\), where \(k\) is the size of the cover in the optimal solution.

**Proof:** We have that \(|T_i| \leq (1 - 1/k)|T_{i-1}| \leq (1 - 1/k)^i |T_0| = (1 - 1/k)^i n\). In particular, for \(M = \lceil 2k \log n \rceil\) we have

\[ |T_M| \leq \left(1 - \frac{1}{k}\right)^M n \leq \exp\left(-\frac{1}{k}M\right) n = \exp\left(-\frac{2k \log n}{k}\right) n \leq \frac{1}{n} < 1, \]

since \(1 - x \leq e^{-x}, \) for \(x \geq 0\). Namely, \(|T_M| = 0\). As such, the algorithm terminates before reaching the \(M\)th iteration, and as such it outputs a cover of size \(O(k \log n)\), as claimed.
3 Biographical Notes

The 3/2-approximation for TSP with the triangle inequality is due to Christofides [Chr76].

The Max 3SAT remains hard in the “easier” variant of Max 2SAT version, where every clause has 2 variables. It is known to be NP-HARD and approximable within 1.0741 [FG95], and is not approximable within 1.0476 [Has97]. Notice, that the fact that Max 2SAT is hard to approximate is surprising if one consider the fact that 2SAT can be solved in polynomial time.

References

