

Chapter 7

Approximation algorithms II

By Sariel Har-Peled, December 7, 2009^①

Version: 0.3

7.1 Max Exact 3SAT

We remind the reader that an instance of **3SAT** is a boolean formula, for example $F = (x_1 + x_2 + x_3)(x_4 + \overline{x_1} + x_2)$, and the decision problem is to decide if the formula has a satisfiable assignment. Interestingly, we can turn this into an optimization problem.

Problem: Max 3SAT

Instance: A collection of clauses: C_1, \dots, C_m .

Output: Find the assignment to x_1, \dots, x_n that satisfies the maximum number of clauses.

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 7.1.1 Algorithm **Alg** for a maximization problem achieves an approximation factor α if for all inputs, we have:

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

In the following, we present a *randomized algorithm* – it is allowed to consult with a source of random numbers in making decisions. A key property we need about random variables, is the linearity of expectation property, which is easy to derive directly from the definition of expectation.

^①This work is licensed under the Creative Commons Attribution-Noncommercial 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

Definition 7.1.2 (Linearity of expectations.) Given two random variables X, Y (not necessarily independent), we have that $\mathbf{E}[X + Y] = \mathbf{E}[X] + \mathbf{E}[Y]$.

Theorem 7.1.3 *One can achieve (in expectation) $(7/8)$ -approximation to **Max 3SAT** in polynomial time. Namely, if the instance has m clauses, then the generated assignment satisfies $(7/8)m$ clauses in expectation.*

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

Let Y_i be the indicator variables which is 1 if (and only if) the i th clause is satisfied by the generated random assignment and 0 otherwise, for $i = 1, \dots, m$. Formally, we have

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

Now, the number of clauses satisfied by the given assignment is $Y = \sum_{i=1}^m Y_i$. We claim that $\mathbf{E}[Y] = (7/8)m$, where m is the number of clauses in the input. Indeed, we have

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

by linearity of expectation. Now, what is the probability that $Y_i = 0$? This is the probability that all three literals appear in the clause C_i are evaluated to **FALSE**. Since the three literals are instance of three distinct variable, these three events are independent, and as such the probability for this happening is

$$\mathbf{Pr}[Y_i = 0] = \frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}.$$

(Another way to see this, is to observe that since C_i has exactly three literals, there is only one possible assignment to the three variables appearing in it, such that the clause evaluates to **FALSE**. Now, there are eight (8) possible assignments to this clause, and thus the probability of picking a **FALSE** assignment is $1/8$.) Thus,

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

and

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

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

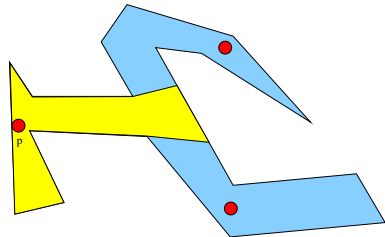
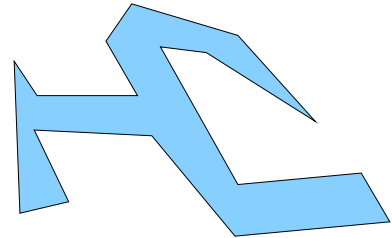
Curiously, Theorem 7.1.3 is stronger than what one usually would be able to get for an approximation algorithm. Here, the approximation quality is independent of how well the optimal solution does (the optimal can satisfy at most m clauses, as such we get a $(7/8)$ -approximation. Curiouser and curiouser², the algorithm does not even look on the input when generating the random assignment.

²“Curiouser and curiouser!” Cried Alice (she was so much surprised, that for the moment she quite forgot how to speak good English). – Alice in wonderland, Lewis Carol

7.2 Approximation Algorithms for Set Cover

7.2.1 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* a point q , if the segment pq is contained inside the polygon. See figure on the right, for an illustration of how the input looks like.



A *visibility polygon* at p (depicted as the yellow polygon on the left) is the region inside the polygon that p can see. WE would like to find the *minimal* number of guards needed to guard the given art-gallery? That is, all the points in the art gallery should be visible from at least one guard we place.

The art-gallery problem is a set-cover problem. We have a ground set (the polygon), and family of sets (the set of all visibility polygons), and the target is to find a minimal number of sets covering the whole polygon.

It is known that finding the minimum number of guards needed is **NP-HARD**. No approximation is currently known. It is also 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 *open problem* is to find a polynomial time approximation algorithm, such that given P , it computes a set of guards, such that $\#guards \leq \sqrt{nk_{opt}}$, 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.

7.2.2 Set Cover

The optimization version of **Set Cover**, is the following:

Problem: Set Cover

Instance: (S, \mathcal{F})

S - a set of n elements

\mathcal{F} - a family of subsets of S , s.t. $\bigcup_{X \in \mathcal{F}} X = S$.

Output: The set $\mathcal{X} \subseteq \mathcal{F}$ such that \mathcal{X} contains as few sets as possible, and \mathcal{X} covers S .

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

Example 7.2.1 Consider the set $S = \{1, 2, 3, 4, 5\}$ and the following family of subsets

$$\mathcal{F} = \{\{1, 2, 3\}, \{2, 5\}, \{1, 4\}, \{4, 5\}\}.$$

Clearly, the smallest cover of S is $\mathcal{X}_{opt} = \{\{1, 2, 3\}, \{4, 5\}\}$.

The greedy algorithm **GreedySetCover** for this problem is depicted on the right. Here, the algorithm always picks the set in the family that covers the largest number of elements not covered yet. Clearly, the algorithm is 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 takes time polynomial in m and n . Let $\mathcal{X}_{opt} = \{V_1, \dots, V_k\}$ be the optimal solution.

```

GreedySetCover( $S, \mathcal{F}$ )
 $\mathcal{X} \leftarrow \emptyset; T \leftarrow S$ 
while  $T$  not empty do
     $U \leftarrow$  set in  $\mathcal{F}$  covering
        largest # of elements in  $T$ 
     $\mathcal{X} \leftarrow \mathcal{X} \cup \{U\}$ 
     $T \leftarrow T \setminus U$ 

return  $\mathcal{X}$ .

```

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 7.2.2 We have $\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_k \geq \dots \geq \alpha_m$.

Proof: If $\alpha_i < \alpha_{i+1}$ then U_{i+1} covers more elements than U_i and we can exchange between them, as we found a set that in the i th iteration covers more elements than the set used by **GreedySetCover**. Namely, in the i th iteration we would use U_{i+1} instead of U_i . This contradicts the greediness of **GreedySetCover** of choosing the set covering the largest number of elements not covered yet. A contradiction. ■

Claim 7.2.3 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 S , and as such it covers $T_i \subseteq S$. This implies that one of the subsets in the optimal solution cover at least $1/k$ fraction 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 $\alpha_i \geq |T_i|/k$ elements.

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

Theorem 7.2.4 The algorithm **GreedySetCover** generates a cover of S using at most $O(k \log n)$ sets of \mathcal{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 $M = \lceil 2k \ln n \rceil$ we have

$$|T_M| \leq \left(1 - \frac{1}{k}\right)^M n \leq \exp\left(-\frac{1}{k}M\right) n = \exp\left(-\frac{\lceil 2k \ln n \rceil}{k}\right) n \leq \frac{1}{n} < 1,$$

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. ■

7.3 Biographical Notes

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.

Bibliography

- [FG95] U. Feige and M. Goemans. Approximating the value of two power proof systems, with applications to max 2sat and max dicut. In *ISTCS '95: Proceedings of the 3rd Israel Symposium on the Theory of Computing Systems (ISTCS'95)*, page 182, Washington, DC, USA, 1995. IEEE Computer Society.
- [Has97] J. Hastad. Some optimal inapproximability results. In *Proc. 29th Annu. ACM Sympos. Theory Comput.*, pages 1–10, 1997.