Approximation Algorithms

How to Tackle *NP*-Hard Optimization Problems?

Most interesting discrete optimization problems are *NP*-hard.

Thus, unless $P = NP$, for such problems we cannot find algorithms that simultaneously

1. find optimal solutions
2. in polynomial time
3. for any instance.

To deal with *NP*-hard optimization problems, we need to relax at least one of the three requirements.

In this course, we study approximation algorithms. That is, we relax the first requirement, i. e., we search for algorithms that produce solutions that are "good enough".

What is "good enough"?
We would like to have some sort of performance guarantee.

Approximation Algorithms

Definition 1.1.

An α-approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of α of the optimum value.

For an α-approximation algorithm, we call α the performance guarantee (or approximation ratio/factor) of the algorithm.

Convention:

- $\alpha > 1$ for minimization problems, and
- $\alpha < 1$ for maximization problems.

Thus, a $\frac{1}{2}$-approximation algorithm for a maximization problem is a polynomial-time algorithm that always returns a solution whose value is at least half the value of the optimum value.

Example:
Christofides' algorithm is a $\frac{3}{2}$-approximation algorithm for the TSP.

Polynomial-Time Approximation Schemes

Remarks:

- The worst-case bounds are often due to pathological cases that hardly arise in practice.
- Often, approximation algorithms are much better in practice than indicated by their performance guarantee.

Definition 1.2.

A polynomial-time approximation scheme (PTAS) is a family of algorithms $\{A_\varepsilon\}$, where there is an algorithm for each $\varepsilon > 0$, such that A_ε is a

- $(1+\varepsilon)$-approximation algorithm (for minimization problems), or a
- $(1-\varepsilon)$-approximation algorithm (for maximization problems).

Examples: PTASes exist for the Knapsack Problem and the Euclidean TSP.

1 The Set Cover Problem

The minimum set cover problem can be formalized as follows. We are given sets $T_1, \ldots, T_m$ that cover some universe with n elements, and the goal is to find a family of sets with minimal cardinality whose union covers all the elements in the universe. We assume that the number of sets m and the number of elements in the universe n are polynomially related, i.e. $m \approx n^c$, for some constant $c > 0$. We will express the approximation ratio in terms of n (the number of elements in the universe), but the results are asymptotically equivalent to in terms of m.

1.1 The greedy algorithm for set cover

Notice that the decision version of the min-set cover is exactly the decision version of max-cover: *given a family of sets $T_1, \ldots, T_m$ is there a family of sets of size k that covers at least d elements in the universe?* We know that this decision problem is NP-complete, and hence min-set cover is an NP-hard optimization problem. Our goal will therefore be to design a good approximation algorithm for this problem. A natural candidate is the greedy algorithm presented in Algorithm 1, which is a straightforward adaptation of the greedy algorithm for max-cover.

Theorem 1. *The greedy algorithm is a $\Theta(\log n)$ approximation.*

Algorithm 1 Greedy algorithm for Min Set Cover

```
S ← ∅
while not all elements in the universe are covered do
    T set that covers the most elements that are not yet covered by S
    S ← S ∪ {T}
end while
return S
```

Proof. First, observe that the algorithm terminates after at most m stages. Since there are m sets and in each iteration of the `while` loop we add a set to the solution, we will terminate after at most m steps.

Let u_j denote the number of elements in the universe that are still *not* covered at iteration j of the while loop, and let k denote the number of sets in the optimal solution, i.e. $k = \texttt{OPT}$. In each iteration j we can use all the k sets in the optimal solution to cover the entire universe, and in particular to cover u_j. Therefore, there must exist at least one set in the optimal solution that covers at least u_j/k elements. Since we select the set whose marginal contribution is largest at each iteration, this implies that in every iteration we include at least u_j/k elements into the solution. Put differently, we know that after iteration j we are left with at most $u_j - u_j/k$ elements. That is:

$$u_{j+1} \leq \left(u_j - \frac{u_j}{k}\right) \leq \left(1 - \frac{1}{k}\right) u_j \leq \left(1 - \frac{1}{k}\right)\left(1 - \frac{1}{k}\right) u_{j-1} \leq \ldots \leq \left(1 - \frac{1}{k}\right)^{j+1} u_0 = \left(1 - \frac{1}{k}\right)^{j+1} n$$

where the last equality is due to the fact that $u_0 = n$ since there are exactly n elements in the universe before the first iteration. Notice also that once we get to stage i for which $u_i \leq 1$ we're done, since this implies that we need to select at most one more set and obtain a set cover. So the question is how large does i need to be to guarantee that $u_i \leq 1$? A simple bound shows that whenever $i \geq k \cdot \ln n$ we have that $u_i \leq 1$:

$$\left(1 - \frac{1}{k}\right)^i = \left(\left(1 - \frac{1}{k}\right)^k\right)^{\frac{i}{k}} \leq e^{-\frac{i}{k}}$$

we can approximate the number of iterations as if the size is reduced by a factor of $1/e$:

$$n \cdot e^{-\frac{i}{k}} \leq 1 \iff e^{-\frac{i}{k}} \leq n^{-1} \iff -i/k \leq -\ln n \iff i \geq k \ln n$$

and we therefore have that after $i = k \cdot \ln n$ steps the remaining number of elements u_i is smaller or equal to 1. Thus, after at most $k \cdot \ln n + 1 = \texttt{OPT} \ln n + 1$ iterations the algorithm will terminate with a set cover whose size is at most $k \cdot \ln n + 1 = \texttt{OPT} \ln n + 1 \in \Theta(\ln n \cdot \texttt{OPT}) = \Theta(\log n \cdot \texttt{OPT})$. □

Until this point we have been fortunate to find constant-factor approximation algorithms, which makes the $\log n$ approximation factor seem somewhat disappointing. It turns out however that improving over the $\log n$ approximation ratio is impossible in polynomial time unless P=NP[1]. We can look at the glass as half-full: we have an optimal (unless P=NP) approximation algorithm.

1.2 A rounding algorithm for small frequencies

The minimum set cover problem is a generalization of a simpler problem we studied. Recall the *vertex cover problem* (problem set 9) where we have an undirected graph, and wish to find the fewest

nodes that cover all the edges in the graph. We can model this as a minimum set cover problem by associating each vertex v_i with a set T_i and each edge e_j becomes an element a_j in the universe that we aim to cover. While this is an instance of the minimum set cover problem, it is an easier one. In particular, the *frequency* of elements in sets is bounded.

Definition. *For a given instance of set cover we say that an element $j \in [n]$ appears with* ***frequency*** *c_j if it is covered by at most c_j sets. The* ***frequency of the instance*** *is $c = \max_{j \in [n]} c_j$.*

In general, the frequency of the instance can be as large as m which is the number of sets. In some cases however, it is very reasonable to assume that we have instances with bounded frequencies. In the vertex cover problem, for example, the frequency is 2 since each edge can belong to at most two vertices. We have already seen in the problem set (or at least in the solution) that one can achieve an approximation ratio of 2 for this problem, which is far better than the $O(\log n)$ we have for the general minimum set cover case. Can we generally obtain better guarantees when the frequency is bounded? The following algorithm gives an approximation ratio that depends on c.

Algorithm 3 Rounding algorithm

1: $S \leftarrow \emptyset$
2: $\mathbf{p} \leftarrow$ Solution to LP
3: $S \leftarrow$ all sets $i \in [m]$ for which $p_j \geq 1/c$
4: **return** S

Theorem 3. *For any instance of the minimum set cover problem whose frequency is c, the rounding algorithm above returns a solution which is a c-approximation to* `OPT`.

Proof. Let a be an element in the universe, which, w.l.o.g. is covered by sets $T_1, \ldots, T_k$, the requirement of the LP is that the corresponding variables $p_1, \ldots, p_k$ respect: $p_1 + \ldots + p_k \geq 1$. Therefore, there must be at least one variable p_i s.t. $p_i \geq 1/k$. Since the instance frequency is c we know that $k \leq c$ and therefore $p_i \geq 1/k \geq 1/c$. By the condition of the algorithm this implies that we selected at least one set that covers a. Since this holds for all elements in the universe we have a set cover.

Regarding the approximation ratio, consider the cost of the LP: $\sum_{i=1}^{m} p_i$, and let the I denote the set of indices for which $p_i \geq 1/c$, notice that the solution that we selected has cost $\sum_{i \in I} c \cdot p_i$ since we turned each set whose cost was greater or equal to $1/c$ to 1. Thus:

$$|S| = \sum_{i \in I} c \cdot p_i = c \cdot \sum_{i \in I} p_i \leq c \cdot \sum_{i=1}^{m} p_i = c \cdot \texttt{OPT}_{LP} \leq c \cdot \texttt{OPT}. \qquad \square$$