25 Approximation Algorithms

Many important problems are NP-hard and just ignoring them is not an option. There are indeed many things one can do. For problems of small size, even exponential-time algorithms can be effective and special subclasses of hard problems sometimes have polynomial-time algorithms. We consider a third coping strategy appropriate for optimization problems, which is computing almost optimal solutions in polynomial time. In case the aim is to maximize a positive cost, a $\varrho(n)$-*approximation algorithm* is one that guarantees to find a solution with cost $C \geq C^*/\varrho(n)$, where C^* is the maximum cost. For minimization problems, we would require $C \leq C^*\varrho(n)$. Note that $\varrho(n) \geq 1$ and if $\varrho(n) = 1$ then the algorithm produces optimal solutions. Ideally, ϱ is a constant but sometime even this is not achievable in polynomial time.

Vertex cover. The first problem we consider is finding the minimum set of vertices in a graph $G = (V, E)$ that covers all edges. Formally, a subset $V' \subseteq V$ is a *vertex cover* if every edge has at least one endpoint in V'. Observe that V' is a vertex cover iff $V - V'$ is an independent set. Finding a minimum vertex cover is therefore equivalent to finding a maximum independent set. Since the latter problem is NP-complete, we conclude that finding a minimum vertex cover is also NP-complete. Here is a straightforward algorithm that achieves approximation ratio $\varrho(n) = 2$, for all $n = \operatorname{card} V$.

$V' = \emptyset$; $E' = E$;
`while` $E' \neq \emptyset$ `do`
  select an arbitrary edge uv in E';
  add u and v to V';
  remove all edges incident to u or v from E'
`endwhile`.

Clearly, V' is a vertex cover. Using adjacency lists with links between the two copies of an edge, the running time is O($n + m$), where m is the number of edges. Furthermore, we have $\varrho = 2$ because every cover must pick at least one vertex of each edge uv selected by the algorithm, hence $C \leq 2C^*$. Observe that this result does not imply a constant approximation ratio for the maximum independent set problem. We have $\operatorname{card}(V - V') = n - C \geq n - 2C^*$, which we have to compare with $n - C^*$, the size of the maximum independent set. For $C^* = \frac{n}{2}$, the approximation ratio is unbounded.

Let us contemplate the argument we used to relate C and C^*. The set of edges uv selected by the algorithm is a *matching*, that is, a subset of the edges so that no two share a vertex. The size of the minimum vertex cover is at least the size of the largest possible matching. The algorithm finds a matching and since it picks two vertices per edge, we are guaranteed at most twice as many vertices as needed. This pattern of bounding C^* by the size of another quantity (in this case the size of the largest matching) is common in the analysis of approximation algorithms. Incidentally, for bipartite graphs, the size of the largest matching is equal to the size of the smallest vertex cover. Furthermore, there is a polynomial-time algorithm for computing them.

Traveling salesman. Second, we consider the traveling salesman problem, which is formulated for a complete graph $G = (V, E)$ with a positive integer cost function $c : E \to \mathbb{Z}_+$. A *tour* in this graph is a Hamiltonian cycle and the problem is finding the tour, A, with minimum total cost, $c(A) = \sum_{uv \in A} c(uv)$. Let us first assume that the cost function satisfies the triangle inequality, $c(uw) \leq c(uv) + c(vw)$ for all $u, v, w \in V$. It can be shown that the problem of finding the shortest tour remains NP-complete even if we restrict it to weighted graphs that satisfy this inequality. We formulate an algorithm based on the observation that the cost of every tour is at least the cost of the minimum spanning tree, $C^* \geq c(T)$.

1 Construct the minimum spanning tree T of G.

2 Return the preorder sequence of vertices in T.

Using Prim's algorithm for the minimum spanning tree, the running time is O(n^2). Figure 114 illustrates the algorithm. The preorder sequence is only defined if we have

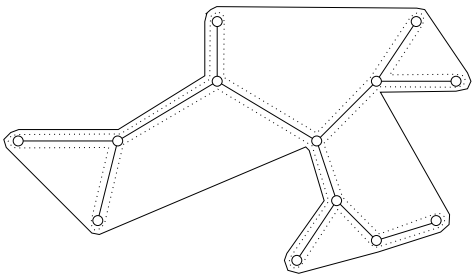

Figure 114: The solid minimum spanning tree, the dotted traversal using each edge of the tree twice, and the solid tour obtained by taking short-cuts.

a root and the neighbors of each vertex are ordered, but

we may choose both arbitrarily. The cost of the returned tour is at most twice the cost of the minimum spanning tree. To see this, consider traversing each edge of the minimum spanning tree twice, once in each direction. Whenever a vertex is visited more than once, we take the direct edge connecting the two neighbors of the second copy as a short-cut. By the triangle inequality, this substitution can only decrease the overall cost of the traversal. It follows that $C \leq 2c(T) \leq 2C^*$.

The triangle inequality is essential in finding a constant approximation. Indeed, without it we can construct instances of the problem for which finding a constant approximation is NP-hard. To see this, transform an unweighted graph $G' = (V', E')$ to the complete weighted graph $G = (V, E)$ with

$$c(uv) \quad = \quad \begin{cases} 1 & \text{if } uv \in E', \\ \varrho n + 1 & \text{otherwise.} \end{cases}$$

Any ϱ-approximation algorithm must return the Hamiltonian cycle of G', if there is one.

Set cover. Third, we consider the problem of covering a set X with sets chosen from a set system $\mathcal{F}$. We assume the set is the union of sets in the system, $X = \bigcup \mathcal{F}$. More precisely, we are looking for a smallest subsystem $\mathcal{F}' \subseteq \mathcal{F}$ with $X = \bigcup \mathcal{F}'$. The *cost* of this subsystem is the number of sets it contains, $\operatorname{card} \mathcal{F}'$. See Figure 115 for an illustration of the problem. The vertex cover problem is a special

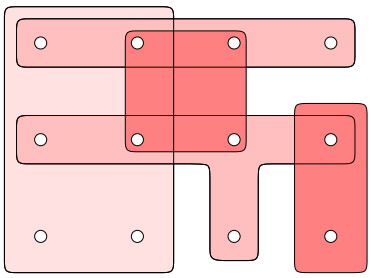

Figure 115: The set X of twelve dots can be covered with four of the five sets in the system.

case: $X = E$ and $\mathcal{F}$ contains all subsets of edges incident to a common vertex. It is special because each element (edge) belongs to exactly two sets. Since we no longer have a bound on the number of sets containing a single element, it is not surprising that the algorithm for vertex covers does not extend to a constant-approximation algorithm for set covers. Instead, we consider the following greedy approach that selects, at each step, the set containing the maximum number of yet uncovered elements.

$\mathcal{F}' = \emptyset$; $X' = X$;
`while` $X' \neq \emptyset$ `do`
  select $S \in \mathcal{F}$ maximizing $\operatorname{card}(S \cap X')$;
  $\mathcal{F}' = \mathcal{F}' \cup \{S\}$; $X' = X' - S$
`endwhile`.

Using a sparse matrix representation of the set system (similar to an adjacency list representation of a graph), we can run the algorithm in time proportional to the total size of the sets in the system, $n = \sum_{S \in \mathcal{F}} \operatorname{card} S$. We omit the details.

Analysis. More interesting than the running time is the analysis of the approximation ratio the greedy algorithm achieves. It is convenient to have short notation for the d-th harmonic number, $H_d = \sum_{i=1}^{d} \frac{1}{i}$ for $d \geq 0$. Recall that $H_d \leq 1 + \ln d$ for $d \geq 1$. Let the size of the largest set in the system be $m = \max\{\operatorname{card} S \mid S \in \mathcal{F}\}$.

CLAIM. The greedy method is an H_m-approximation algorithm for the set cover problem.

PROOF. For each set S selected by the algorithm, we distribute \$1 over the $\operatorname{card}(S \cap X')$ elements covered for the first time. Let c_x be the cost allocated this way to $x \in X$. We have $\operatorname{card} \mathcal{F}' = \sum_{x \in X} c_x$. If x is covered the first time by the i-th selected set, S_i, then

$$c_x \quad = \quad \frac{1}{\operatorname{card}(S_i - (S_1 \cup \ldots \cup S_{i-1}))}.$$

We have $\operatorname{card} \mathcal{F}' \leq \sum_{S \in \mathcal{F}^*} \sum_{x \in S} c_x$ because the optimal cover, $\mathcal{F}^*$, contains each element x at least once. We will prove shortly that $\sum_{x \in S} c_x \leq H_{\operatorname{card} S}$ for every set $S \in \mathcal{F}$. It follows that

$$\operatorname{card} \mathcal{F}' \quad \leq \quad \sum_{S \in \mathcal{F}^*} H_{\operatorname{card} S} \quad \leq \quad H_m \operatorname{card} \mathcal{F}^*,$$

as claimed. □

For $m = 3$ we get $\varrho = H_3 = \frac{11}{6}$. This implies that for graphs with vertex-degrees at most 3, the greedy algorithm guarantees a vertex cover of size at most $\frac{11}{6}$ times the optimum, which is better than the ratio 2 guaranteed by our first algorithm.

We still need to prove that the sum of costs c_x over the elements of a set S in the system is bounded from above by $H_{\operatorname{card} S}$. Let u_i be the number of elements in

S that are not covered by the first i selected sets, $u_i = \mathrm{card}\,(S - (S_1 \cup \ldots \cup S_i))$, and observe that the numbers do not increase. Let u_{k-1} be the last non-zero number in the sequence, so $\mathrm{card}\,S = u_0 \geq \ldots \geq u_{k-1} > u_k = 0$. Since $u_{i-1} - u_i$ is the number of elements in S covered the first time by S_i, we have

$$\sum_{x \in S} c_x \quad = \quad \sum_{i=1}^{k} \frac{u_{i-1} - u_i}{\mathrm{card}\,(S_i - (S_1 \cup \ldots \cup S_{i-1}))}.$$

We also have $u_{i-1} \leq \mathrm{card}\,(S_i - (S_1 \cup \ldots \cup S_{i-1}))$, for all $i \leq k$ because of the greedy choice of S_i. The problem thus reduces to bounding the sum of ratios $\frac{u_{i-1}-u_i}{u_{i-1}}$. It is not difficult to see that this sum can be at least logarithmic in the size of S. Indeed, if we choose u_i about half the size of u_{i-1}, for all $i \geq 1$, then we have logarithmically many terms, each roughly $\frac{1}{2}$. We use a sequence of simple arithmetic manipulations to prove that this lower bound is asymptotically tight:

$$\begin{aligned} \sum_{x \in S} c_x \quad &\leq \quad \sum_{i=1}^{k} \frac{u_{i-1} - u_i}{u_{i-1}} \\ &= \quad \sum_{i=1}^{k} \sum_{j=u_i+1}^{u_{i-1}} \frac{1}{u_{i-1}}. \end{aligned}$$

We now replace the denominator by $j \leq u_{i-1}$ to form a telescoping series of harmonic numbers and get

$$\begin{aligned} \sum_{x \in S} c_x \quad &\leq \quad \sum_{i=1}^{k} \sum_{j=u_i+1}^{u_{i-1}} \frac{1}{j} \\ &= \quad \sum_{i=1}^{k} \left(\sum_{j=1}^{u_{i-1}} \frac{1}{j} - \sum_{j=1}^{u_i} \frac{1}{j} \right) \\ &= \quad \sum_{i=1}^{k} (H_{u_{i-1}} - H_{u_i}), \end{aligned}$$

which is equal to $H_{u_0} - H_{u_k} = H_{\mathrm{card}\,S}$. This fills the gap left in the analysis of the greedy algorithm.