Algorithms and Uncertainty, Winter 2025/26 Lecture 14 (5 pages)

Stochastic Two-Stage Set Cover

Thomas Kesselheim Last Update: November 24, 2025

In the analysis of online algorithms, we assumed that we have to make commitments right away. In practice often restrictions are not as strict. Just suppose you have to fly to New York City two months from now. You could either buy the ticket now for a cheap price or later on. Now the ticket is cheap but there is a chance that you actually cannot go on the trip. So, it might also make sense to wait and buy the ticket for a higher price when it is certain that you have to go.

This is a typical example of a multi-stage optimization problem. These are problems in which the optimization instance gets more and more concrete over time and decisions can be made on the way. There are both models with stochastic as well as adversarial inputs. Today, we will consider simple examples of such stochastic problems.

1 Stochastic Two-Stage Set Cover

Recall that in the offline Set Cover problem, there is a universe of m elements U and a family of subsets $\mathcal{S} \subseteq 2^U$. Each set $S \in \mathcal{S}$ has a cost c_S. We have to select a cover $\mathcal{C} \subseteq \mathcal{S}$ such that for all $e \in U$ there is some $S \in \mathcal{C}$ with $e \in \mathcal{S}$. We want to minimize the cost $\sum_{S \in \mathcal{C}} c_S$.

In the two-stage version, only a subset $A \subseteq U$ has to be covered. That is, only for $e \in A$, there has to be $S \in \mathcal{C}$ with $e \in \mathcal{S}$. It is uncertain which set A is. We consider the stochastic variant, in which the set A is drawn from a known probability distribution. We let p_A, $A \subseteq U$, denote the probability that A has to be covered.

Eventually, we will have to cover all of A. We have two opportunities to select sets: Before A is revealed and afterwards. Before A is revealed (stage I), adding $S \in \mathcal{S}$ costs c_S^{I}; after A is revealed (stage II), it costs $c_S^{\mathrm{II}} \geq c_S^{\mathrm{I}}$.

Important special cases are as follows. We might have $c_S^{\mathrm{I}} = c_S^{\mathrm{II}}$ for all S. In this case, choosing sets in the first stage does not make any sense and we might as well wait until the second stage. If $c_S^{\mathrm{II}} = \infty$, then we want to cover all elements that can possibly show up already in the first stage.

We know the distribution $(p_A)_{A \subseteq U}$ and well as both cost vectors $(c_S^{\mathrm{I}})_{S \in \mathcal{S}}$ and $(c_S^{\mathrm{II}})_{S \in \mathcal{S}}$ in advance. The goal is to minimize the expected cost

$$\sum_{S \in \mathcal{C}_0} c_S^{\mathrm{I}} + \mathbf{E}\left[\sum_{S \in \mathcal{C}_A} c_S^{\mathrm{II}} \right] ,$$

where $\mathcal{C}_0 \subseteq \mathcal{S}$ denotes the choice in the first stage and $\mathcal{C}_A \subseteq \mathcal{S}$ denotes the choice in the second stage when $A \subseteq U$ is active.

2 Our Goal

Observe that the stochastic set-cover problem can be modeled as a Markov decision process with time horizon $T = 2$. So, we could in principle use the algorithm based on dynamic programming to compute an optimal policy. However, the number of states will be huge. Computing it is at least as hard as solving the Set Cover problem optimally because one special case is that $p_A = 1$ for one set A. Set Cover is an NP-hard problem, so we cannot hope to find an exact algorithm that runs in polynomial time. Therefore, we will be interested in *approximating the optimal policy* in polynomial time.

Given any instance $\mathcal{I}$ of the problem, that is the probability distribution over sets of elements and the different cost vectors, let $C_{\mathcal{I}}(\pi)$ denote the expected cost of policy π. There is an optimal policy $\pi^*_{\mathcal{I}}$ such that $C_{\mathcal{I}}(\pi^*_{\mathcal{I}}) \leq C_{\mathcal{I}}(\pi)$ for any policy π. Our goal is to design a polynomial time algorithm with the following property. Given an instance $\mathcal{I}$, it is supposed to compute a policy π such that $C_{\mathcal{I}}(\pi) \leq \alpha \cdot C_{\mathcal{I}}(\pi^*_{\mathcal{I}})$, where $\alpha > 1$ is as small as possible.

Note that $\pi^*_{\mathcal{I}}$ is *not* the offline optimum. Indeed, there is not a lot we can do if we are compared to the offline optimum. Suppose we have only a single element e, which has to be covered with probability ϵ. Covering it in the first phase costs ϵ; in the second phase it costs 1. Any policy has expected cost ϵ but the offline optimum has expected cost ϵ^2.

3 Offline Set Cover and the Greedy Algorithm

Let us first revisit the offline problem. That is, we only have to find a cover $\mathcal{C}$ such that all of U is covered. We have already seen the LP relaxation before:

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{S}} c_S x_S & \\ \text{subject to} \quad & \sum_{S \colon e \in S} x_S \geq 1 & \text{for all } e \in U \\ & x_S \geq 0 & \text{for all } S \in \mathcal{S} \end{aligned}$$

Every solution to the Set Cover problem also corresponds to a feasible solution to the LP relaxation. However, the best *fractional* solution can be cheaper but not arbitrarily so. We have seen before how to round LP solutions to feasible Set Cover solutions. But there is an even easier approach: Run a simple greedy algorithm. We will show that the cost of its solution is also bounded in terms of the cheapest LP solution.

The greedy algorithm for offline Set Cover is truly simple. It works as follows

- Initially, set $U' := U$
- While $U' \neq \emptyset$
  - Let S be the set that minimizes $\frac{c_S}{|S \cap U'|}$
  - Add S to $\mathcal{C}$, set $U' := U' \setminus S$.

So, in every step, the algorithm chooses the set S of minimum cost *per newly covered element.*

Theorem 14.1. *Let $\mathcal{C}$ be the cover computed by the greedy algorithm, let x^* be the optimal solution to the LP relaxation. Then $\sum_{S \in \mathcal{C}} c_S \leq O(\log m) \sum_{S \in \mathcal{S}} c_S x^*_S$, where $m = |U|$.*

For completeness, we will prove Theorem 14.1 later. But before, we use it to derive an algorithm for the two-stage problem.

4 Algorithm for Stochastic Two-Stage Set Cover

We formulate an LP relaxation as follows. Given an arbitrary policy, let $x_S = 1$ if set S is selected in the first stage, 0 otherwise. Let $y_{A,S} = 1$ if set S is selected in the second stage if set A has to be covered, 0 otherwise. Based on these variables, we can write the LP

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{S}} c_S^{\mathrm{I}} x_S + \sum_{A \subseteq U} p_A \sum_{S \in \mathcal{S}} c_S^{\mathrm{II}} y_{A,S} \\ \text{subject to} \quad & \sum_{S\colon e \in S} x_S + \sum_{S\colon e \in S} y_{A,S} \geq 1 && \text{for all } A \subseteq U,\ e \in A \\ & x_S, y_{A,S} \geq 0 && \text{for all } S \in \mathcal{S},\ A \subseteq U \end{aligned}$$

It is easy to observe that every policy corresponds to an LP solution whose value is the expected cost of this policy. So, the optimal LP solution can only be cheaper than the optimal policy. If $p_A > 0$ only for a small number of sets, we can solve this linear program in polynomial time.

Now, we can proceed to the multi-stage variant. As said before, our algorithm first solves the LP relaxation and obtains an optimal solution (x^*, y^*). We turn it into a policy as follows.

- Let U_0 be the set of all elements e such that $\sum_{S\colon e \in S} x_S^* \geq \frac{1}{2}$. Cover these elements in first stage: Compute $\mathcal{C}_0$ by running the greedy algorithm on U_0 with costs $(c_S^{\mathrm{I}})_{S \in \mathcal{S}}$.
- Cover $A \setminus U_0$ in the second stage: Compute $\mathcal{C}_A$ by running the greedy algorithm on $A \setminus U_0$ with costs $(c_S^{\mathrm{II}})_{S \in \mathcal{S}}$.

The policy is clearly feasible because whatever A is drawn, each $e \in A$ is covered in the second stage at the latest.

Theorem 14.2. *The algorithm turns any fractional solution to the LP into a feasible policy of at most $O(\log m)$-times the optimal cost in polynomial time.*

Proof. Let us understand the cost of the first stage of our policy. We defined it to cover U_0. The (deterministic) LP relaxation of this problem is the following.

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{S}} c_S^{\mathrm{I}} x_S \\ \text{subject to} \quad & \sum_{S\colon e \in S} x_S \geq 1 && \text{for all } e \in U_0 \\ & x_S \geq 0 && \text{for all } S \in \mathcal{S} \end{aligned}$$

Observe that $2x^*$ is a feasible solution by the way we defined U_0. So the optimal value is at most $2 \sum_{S \in \mathcal{S}} c_S^{\mathrm{I}} x_S^*$. This means that, by Theorem 14.1, our first-stage selection has a cost of at most

$$\sum_{S \in \mathcal{C}_0} c_S^{\mathrm{I}} \leq O(\log m) \cdot 2 \sum_{S \in \mathcal{S}} c_S^{\mathrm{I}} x_S^* \enspace .$$

In the second stage, we only have to cover $A \setminus U_0$. Observe that for all $e \in A \setminus U_0$

$$\sum_{S\colon e \in S} y_{A,S}^* \geq \frac{1}{2}$$

because (x^*, y^*) is a feasible LP solution. So, we can follow just the same idea as above and get a cover of cost at most

$$\sum_{S \in \mathcal{C}_A} c_S^{\mathrm{II}} \leq O(\log m) \cdot 2 \sum_{S \in \mathcal{S}} c_S^{\mathrm{II}} y_{A,S}^* \enspace .$$

In combination, our cover will cost in expectation

$$\sum_{S \in \mathcal{C}_0} c_S^{\mathrm{I}} + \mathbf{E}\left[\sum_{S \in \mathcal{C}_A} c_S^{\mathrm{II}}\right] = \sum_{S \in \mathcal{C}_0} c_S^{\mathrm{I}} + \sum_A p_A \sum_{S \in \mathcal{C}_A} c_S^{\mathrm{II}} \leq O(\log m) \cdot \left(\sum_{S \in \mathcal{S}} c_S^{\mathrm{I}} x_S^* + \sum_A p_A \sum_{S \in \mathcal{S}} c_S^{\mathrm{II}} y_{A,S}^* \right) \enspace .$$

□

5 Analysis of the Greedy Algorithm

It remains to prove Theorem 14.1. To this end, we introduce some more notation. Every element gets removed from U' at some point. Let e_k be the k^{th} element that is removed from the set U', breaking ties arbitrarily. Element e_k gets removed from U' because it is covered by some S for the first time; later more sets covering e_k can follow, which we ignore. Let S_k denote this set S which covers e_k for the first time and let U'_k denote the state of U' at the beginning of the iteration in which e_k is removed.

We define

$$p_k = \frac{c_{S_k}}{|S_k \cap U'_k|}$$

as the cost per newly-covered element that we incur when covering element e_k. Note that while covering e_k we may cover elements for the first time as well and we split up the cost of set c_{S_k} evenly among them. By this definition, we can write the cost that our algorithm incurs as

$$\sum_{S \in \mathcal{C}} c_S = \sum_{k=1}^{m} p_k \quad . \tag{1}$$

Lemma 14.3. *For all k, we have*

$$p_k \leq \frac{\sum_{S \in \mathcal{S}} c_S x^*_S}{m - k + 1} \quad .$$

Proof. Recall that S_k minimizes $\frac{c_S}{|S \cap U'_k|}$. That is, for every S we have

$$p_k \leq \frac{c_S}{|S \cap U'_k|} \quad .$$

Therefore,

$$\sum_{S \in \mathcal{S}} c_S x^*_S \geq \sum_{S : S \cap U'_k \neq \emptyset} c_S x^*_S = \sum_{S : S \cap U'_k \neq \emptyset} \frac{c_S}{|S \cap U'_k|} |S \cap U'_k| x^*_S \geq \sum_{S : S \cap U'_k \neq \emptyset} p_k |S \cap U'_k| x^*_S \geq p_k |U'_k| \quad .$$

Here the last step uses that x^* is a feasible LP solution, i.e., $\sum_{S : e \in S} x^*_S \geq 1$ for all e. So,

$$\sum_{S : S \cap U'_k \neq \emptyset} |S \cap U'_k| x^*_S = \sum_{S : S \cap U'_k \neq \emptyset} \sum_{e \in S \cap U'_k} x^*_S = \sum_{e \in U'_k} \sum_{S : e \in S} x^*_S \geq |U'_k| \quad .$$

In combination, we have

$$p_k \leq \frac{1}{|U'_k|} \sum_{S \in \mathcal{S}} c_S x^*_S \quad .$$

The lemma now follows because $|U'_k| \geq m - k + 1$ because before the k^{th} element is removed, there are at least $m - k + 1$ left. There might be even more because other elements get removed in the same iteration. □

Proof of Theorem 14.1. Note that Equation (1) together with Lemma 14.3 now implies

$$\sum_{S \in \mathcal{C}} c_S = \sum_{k=1}^{m} p_k \leq \sum_{k=1}^{m} \frac{\sum_{S \in \mathcal{S}} c_S x^*_S}{m - k + 1} = \sum_{S \in \mathcal{S}} c_S x^*_S \sum_{k=1}^{m} \frac{1}{k} = O(\log m) \sum_{S \in \mathcal{S}} c_S x^*_S \quad ,$$

which is exactly what we claimed. □

References

- Stochastic optimization is (almost) as easy as deterministic optimization, D. Shmoys, C. Swamy, FOCS 2004 (Set Cover and generalizations)