Spring 2025 Final Report

Evan Brody & Kerry Huang

05.14.2025

Prof. Lisa Hellerstein

1 Introduction

Our research this semester focused around the representative problem in the field of stochastic boolean function evaluation, which concerns a function f defined by

$$f : [d]^n \to \{0, 1\}$$

$$x \mapsto \begin{cases} 1 & \forall c \in [d](\exists i \in [n] : x_i = c) \\ 0 & \text{otherwise} \end{cases}$$

That is, $f(x) = 1$ if every number from 1 to d is present in x, and $f(x) = 0$ otherwise. In the vector x, each entry is an independent random variable that takes on values in $[d]$ according to a given probability mass function (not necessarily identical between entries), and the cost of querying each variable is exactly 1. Our investigation centered around finding efficient algorithms for generating provably cost-efficient strategies for evaluating f given the probability mass function of each x_i. In the following sections we use the terms *representative* and *color* interchangeably to mean an element of $[d]$.

2 The Submodular Goal Function Approach

One approach to cost-efficient boolean function evaluation is to reduce the problem to submodular cover. In submodular cover, a finite set of elements (the "ground set" or "universe") is provided with along with a function termed the "goal function" that acts as an oracle for the value of any subset of the ground set. The goal in this case is to collect enough elements of the ground set to reach a predetermined goal value Γ, while minimizing the number of elements collected. The goal function must satisfy monotonicity, which entails that for any subsets $A \subseteq B \subseteq \mathcal{U}$ where $\mathcal{U}$ is the ground set,

$$u(A) \leq u(B)$$

where $u : 2^{\mathcal{U}} \to \mathbb{Z}^{\geq 0}$ is the goal function. Furthermore, the goal function must be submodular, which entails that for any $A \subseteq B \subseteq \mathcal{U}$ with $x \in \mathcal{U}$ such that $x \notin B$,

$$u(A \cup \{x\}) - u(A) \geq u(B \cup \{x\}) - u(B)$$

That is, one cannot gain more value from a single element x when adding it to some set as opposed to some subset of that set. In addition, u must be such that

$$u(\varnothing) = 0$$

$$u(\mathcal{U}) = \Gamma$$

When applied to stochastic boolean function evaluation, the ground set consists of the parameters of the function to be evaluated, collecting an element corresponds to querying a parameter, and a goal function must be constructed as

$$u : \{*, 0, 1\}^n \to \mathbb{Z}^{\geq 0}$$

where a vector in $\{*, 0, 1\}^n$ represents unknown parameters, parameters revealed to be 0, and parameters revealed to be 1. The requirement for monotonicity translates to the requirement that one cannot lose value by revealing a parameter; that is, for any $\alpha, \beta \in \{*, 0, 1\}^n$ that are identical except for an entry that is $*$ in α but is 0 or 1 in β, $u(\beta) \geq u(\alpha)$ must hold. The requirement for submodularity translates to the requirement that if two vectors α and β are identical except that some entries are $*$ in α but are 0 or 1 in β (in this case, we call β an *extension* of α), revealing an entry that is $*$ in both α and β should produce at least as much value in α as it does in β. That is, if the vector produced by revealing that element in α (resp. β) is α' (resp. β'), then

$$u(\alpha') - u(\alpha) \geq u(\beta') - u(\beta)$$

The function must also satisfy

$$u(*, ..., *) = 0$$

(the all-empty assignment) and

$$u(\gamma) = \Gamma$$

for any assignment γ such that the value of the boolean function can be determined only by the assigned variables in γ (in this case, γ is called a *certificate* of f; with more specificity, a *1-certificate* or a *0-certificate* depending on the determined value of f). One approach to evaluating a function using submodular cover is to use a greedy algorithm that simply chooses the next query to perform by choosing the query with the maximum expected increase in value according to u. This greedy algorithm achieves an approximation factor on the order of $\log(\Gamma)$ to the optimal adaptive strategy, which means that the number of tests it performs is upper bounded by $\mathcal{O}(\log(\Gamma))\mathbf{OPT}$, where $\mathbf{OPT}$ is the cost of the optimal adaptive strategy.

The above restrictions and analysis also hold in the case of the representative problem, where our assignment vectors are in $(\{*\} \cup [d])^n$ instead of $(\{*\} \cup \{0, 1\})^n$.

We took the following approach to constructing a submodular goal function for the representative problem. Consider constructing auxiliary goal functions as follows:

- $u_1(\beta) = |\{c \in [d] : \exists i \in [n] : \beta_i = c\}|$
  - That is, $u_1(\beta)$ is the number of unique representatives found in β
  - Goal value $Q_1 = d$
  - $u_1(\beta) = Q_1$ if and only if β is a 1-certificate
- $\forall S \in 2^{[d]} \backslash \{\emptyset, [d]\}$ define:
  - $u_S(\beta) = |\{\beta_i : \beta_i \in S\}|$
  - That is, the $u_S(\beta)$ number of rolls that have resulted in an element of S
  - Goal value $Q_S = |S| + n - d + 1$

The significance of u_S is that if for some $\beta \in (\{*\} \cup [d])^n$, $u_S(\beta) = Q_S$, then β must be a 0-certificate of f, since we have only seen $|S|$ representatives but have at most $|S| - 1$

remaining tests. We can combine these via an OR construction [2] to define:

$$u(\beta) = Q_1 \left[\prod_{S \in 2^{[d]} \setminus \{\emptyset, [d]\}} Q_S \right] - (Q_1 - u_1(\beta)) \left[\prod_{S \in 2^{[d]} \setminus \{\emptyset, [d]\}} (Q_S - u_S(\beta)) \right]$$

Each goal value is $\mathcal{O}(n)$, and there are exactly $2^d - 1$ different goal functions (1 for 1-certificates and $2^d - 2$ for 0-certificates), which means, after taking their product, our resulting goal value $Q = \mathcal{O}(n^{(2^d)})$. This implies that our approximation factor for the greedy submodular cover algorithm is

$$\mathcal{O}(\log Q) = \mathcal{O}(\log \mathcal{O}(n^{(2^d)})) = \mathcal{O}(2^d \log n)$$

We can do better, even with a nonadaptive strategy, which we show in the next section.

3 The Round Robin Approach

We start with some background necessary to create the round robin approach.

In the unit-cost case (i.e., each test costs exactly 1) nonadaptive round robin is defined by ordering a set of nonadaptive strategies $S_1, S_2, \ldots, S_j$ according to some permutation of $[j]$ and proceeding through this ordering by performing one test from each strategy at a time. That is, the first test of all strategies will be performed according to the ordering, then the second test of all strategies, etc., skipping repeated tests.

We also consider strategy trees and AND-OR trees. Strategy trees specify a strategy to evaluate a boolean vector, enabling the expression of both adaptive and nonadaptive strategies. AND-OR trees are structures that represent a boolean function. For example, if we were to evaluate whether the boolean vector contains an instance of color c, our boolean function $f = y_1 \vee y_2 ... \vee y_j$ where $y_i = 1$ if x_i evaluates to be c and $y_i = 0$ otherwise. The diagram below represents the strategy and AND-OR tree for this problem.

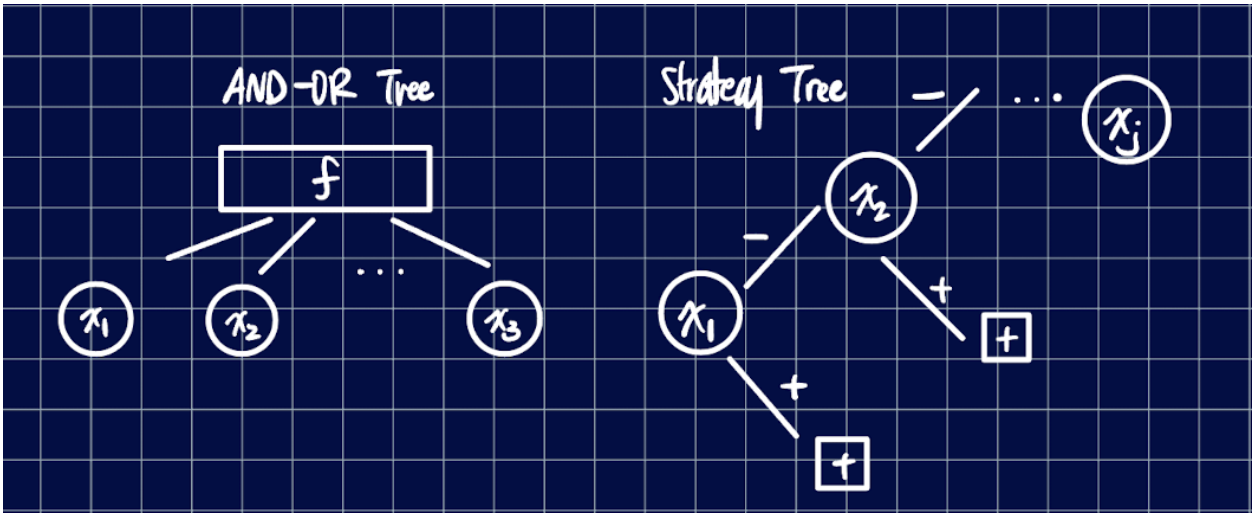

Figure 1: The strategy tree and AND-OR tree for the OR function

3.1 Relative Ordering

A conjecture we explored is that the relative ordering of the dice will not change conditioned on the fact that $f(x) = 1$. This would be a conjecture that could potentially help us demonstrate a d approximation for the round robin in the case where $d < \frac{n+1}{2}$ and $f(x) = 1$. Unfortunately, this conjecture is not true. We provide the following counterexample. Consider the case where $n = d = 3$ and a distribution defined as follows:

	x_1	x_2	x_3
$\mathbb{P}(x = 1)$	9/10	8/10	7/10
$\mathbb{P}(x = 2)$	1/10	0	0
$\mathbb{P}(x = 3)$	0	2/10	3/10

Ordering the dice in decreasing order of $\mathbb{P}(x = 1)$ produces x_1, x_2, x_3. However, note that $\mathbb{P}(x_1 = 2|f(x) = 1) = 1$, so therefore $\mathbb{P}(x_1 = 1|f(x) = 1) = 0$. This means that x_1 transforms from being the most likely to equal 1 to being the least likely after conditioning on $f(x) = 1$.

3.2 Round Robin Proofs

In this section we show that an adaptive round robin approach RR yields a $2d$ approximation to the optimal adaptive strategy, whereas a nonadaptive round robin approach $\hat{RR}$ yields an approximation factor of $2d^2$ to the optimal adaptive strategy.

We define the adaptive round robin approach as follows. Given a set of nonadaptive strategies $S = \{S_1, S_2, \ldots, S_d\}$ (in the below analysis, we use the term *thread* to refer to one of these strategies), where for any $i \in [d]$, S_i is the nonadaptive strategy that orders tests x_j, $1 \leq j \leq n$, in decreasing order of $\mathbb{P}(x_j = i)$, we begin with the first test from S_1, then proceed to the first test from S_2, continuing through S_d. Then, we remove individual strategies from consideration based on the elements found. For example, if we found a 2 and a 3 during this first pass, we remove S_2 and S_3 from consideration. Then, we iterate again over the strategies that are still under consideration, this time performing the second test of each. We then again remove strategies from consideration based on the revealed elements. This process continues until we have either found all d representatives (a 1-certificate of f) or we have found at most $k < d$ representatives but have performed at least $n - d + k + 1$ tests (a 0-certificate of f).

We define the nonadaptive round robin strategy exactly as above, but with the step of removing strategies from consideration omitted.

We claim the following:

$$\mathbb{E}[\text{cost}(RR, x)] \leq 2d\mathbb{E}[\text{cost}(S^*, x)]$$
$$\mathbb{E}[\text{cost}(\hat{RR}, x)] \leq 2d^2\mathbb{E}[\text{cost}(S^*, x)]$$

Our analysis is broken down into two cases. Note that the analysis of Case 1 applies to both RR and $\hat{RR}$, and that we denote the distribution of x as D.

Case 1: $d \geq \frac{n+1}{2}$

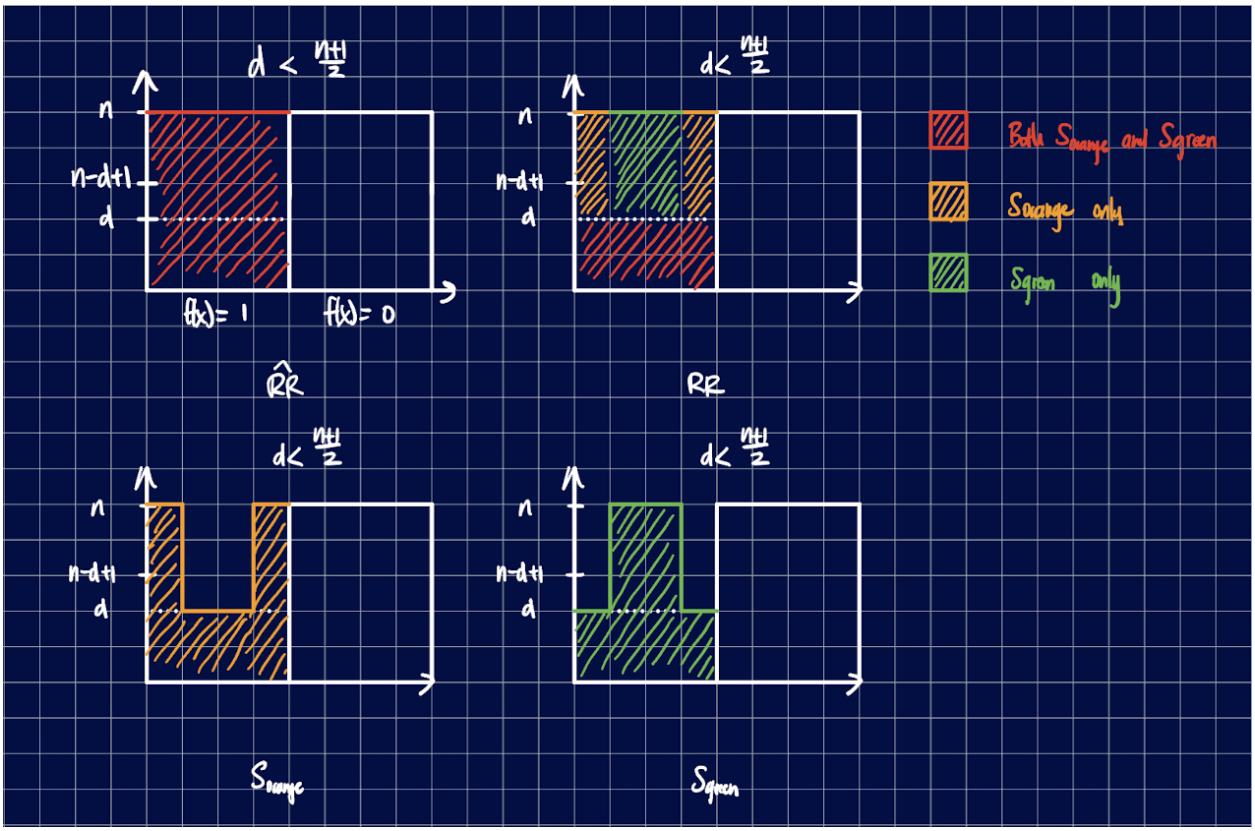

Figure 2: Graphical representations of strategy cost

We claim that any nonredundant strategy S is within a d approximation of the optimal strategy.

We assume that $2 \leq d \leq n$, since if $d = 1$ or $d > n$ the problem is trivial. We have that

$$\text{cost}(S^*, x) \geq \begin{cases} 2 & \text{if } f(x) = 0 \\ d \geq \frac{n+1}{2} & \text{if } f(x) = 1 \end{cases}$$

If $f(x) = 0$, then

$$d \,\text{cost}(S^*, x) \geq 2d \geq n + 1 \geq n \geq \text{cost}(S, x)$$

If $f(x) = 1$, then

$$d \,\text{cost}(S^*, x) \geq d^2 \geq \left(\frac{n+1}{2}\right)^2 \geq n \geq \text{cost}(S, x)$$

Thus, in either case, $\text{cost}(S, x) \leq d \,\text{cost}(S^*, x)$.

Case 2: $d < \frac{n+1}{2}$

First note that

$$\begin{aligned} d &< \frac{n+1}{2} \\ \Leftrightarrow 2d-1 &< n \\ \Leftrightarrow 2(d-1) &< n \\ \Leftrightarrow 0 &< n-2(d-1) \\ \Leftrightarrow n &< 2n-2(d-1) \\ \Leftrightarrow n &< 2(n-d-1) \\ \Leftrightarrow \frac{n}{2} &< n-d-1 \\ \Leftrightarrow \frac{n+1}{2} &< n-d+1 \end{aligned}$$

We then observe that:

$$\text{cost}(S^*, x) \geq \begin{cases} n-d+1 \geq \frac{n+1}{2} & \text{if } f(x)=0 \\ d & \text{if } f(x)=1 \end{cases}$$

We can subsequently generate a new strategy S^+ which behaves exactly as S^* does, except that

$$\forall x \in [d]^n (f(x)=0 \rightarrow \text{cost}(S^+, x) = n)$$

In other words, S^+ is optimal except that it cannot terminate early after recognizing a 0-certificate. It is evident that after increasing the cost only in the case where $f(x)=0$ that

$$\begin{aligned} 2\,\text{cost}(S^*, x) &\geq 2(n-d+1) \\ &\geq n \\ &\geq \text{cost}(S^+, x) \end{aligned}$$

Note that S^+ will find all representatives or continue until it exhausts all options. Thus, for any color $c \in [d]$, if x contains an instance of c, S^+ will find the entry in x that is equal to c. We can trim S^+'s strategy tree such that it terminates upon finding c or upon exhausting all n entries. This means that S^+ can be transformed into a strategy to search for c. Using the adjacency interchange argument presented in [1], it can be shown that the DFA strategy is optimal for depth-1 AND-OR trees. When evaluating whether any variable takes the value c, the problem reduces to computing the logical OR over all variables. This implies that the strategy S_c, which follows a DFA strategy, is optimal for evaluating such a tree. This proves our claim that for any $S_c \in S$

$$\mathbb{E}_{x\sim D}[\text{cost}(S^+, x)] \geq \mathbb{E}_{x\sim D}[\text{cost}(S_c, x)]$$

which therefore implies

$$2\mathbb{E}_{x\sim D}[\text{cost}(S^*, x)] \geq \mathbb{E}_{x\sim D}[\text{cost}(S_c, x)]$$

again for any $S_c \in S$.

3.2.1 RR is a $2d$ approximation for S^*

Proof. Recall that the adaptive RR alternates tests between each thread, terminating threads once they identify the color they were designed to seek. In effect, RR is no different from performing each thread until it identifies the color, and moving on to the next thread until all threads are exhausted. This behavior indicates that:

$$\text{cost}(RR, x) = \sum_{i=1}^{d} \text{cost}(S_i, x)$$

Over every vector x drawn from the distribution, by definition, the expected value of RR is:

$$\begin{aligned}
\mathbb{E}_{x\sim D}[\text{cost}(RR, x)] &= \sum_{x\in[d]^n} \text{cost}(RR, x)\mathbb{P}(x) \\
&= \sum_{x\in[d]^n} \sum_{i=1}^{d} \text{cost}(S_i, x)\mathbb{P}(x) \\
&= \sum_{i=1}^{d} \sum_{x\in[d]^n} \text{cost}(S_i, x)\mathbb{P}(x) \\
&= \sum_{i=1}^{d} \mathbb{E}_{x\sim D}[\text{cost}(S_i, x)] \\
&\leq d\mathbb{E}_{x\sim D}[\text{cost}(S^+, x)] \\
&\leq 2d\mathbb{E}_{x\sim D}[\text{cost}(S^*, x)]
\end{aligned}$$

□

3.2.2 $\hat{RR}$ is a $2d^2$ approximation for S^*

Proof. For some fixed $x \in [d]^n$, denote $\tilde{S}_x$ a strategy in S satisfying

$$\text{cost}(\tilde{S}_x, x) = \max_{1\leq c\leq d} \text{cost}(S_c, x)$$

Note that by definition

$$\text{cost}(\hat{RR}, x) \leq d\,\text{cost}(\tilde{S}_x, x)$$

and therefore that

$$\begin{aligned}
\mathbb{E}_{x\sim D}[\text{cost}(\hat{RR}, x)] &= \sum_{x\in[d]^n} \text{cost}(\hat{RR}, x)\mathbb{P}(x) \\
&\leq d \sum_{x\in[d]^n} \text{cost}(\tilde{S}_x, x)\mathbb{P}(x)
\end{aligned}$$

Now define $\tilde{S}$ as a strategy satisfying

$$\mathbb{E}_{x\sim D}[\text{cost}(\tilde{S}, x)] = \max_{1\leq c\leq d} \mathbb{E}_{x\sim D}[\text{cost}(S_c, x)]$$

and note that for any x

$$\text{cost}(\tilde{S}_x, x) \leq \sum_{c=1}^{d} \text{cost}(S_c, x)$$

Using these two facts, we have that

$$\begin{aligned}
\sum_{x \in [d]^n} \text{cost}(\tilde{S}_x, x)\mathbb{P}(x) &\leq \sum_{x \in [d]^n} \left[\sum_{c=1}^{d} \text{cost}(S_c, x) \right] \mathbb{P}(x) \\
&= \sum_{x \in [d]^n} \sum_{c=1}^{d} \mathbb{P}(x)\, \text{cost}(S_c, x) \\
&= \sum_{c=1}^{d} \sum_{x \in [d]^n} \mathbb{P}(x)\, \text{cost}(S_c, x) \\
&= \sum_{c=1}^{d} \mathbb{E}_{x \sim D}[\text{cost}(S_c, x)] \\
&\leq d\mathbb{E}_{x \sim D}[\text{cost}(\tilde{S}, x)]
\end{aligned}$$

Therefore, returning to the previous inequality,

$$\begin{aligned}
\mathbb{E}_{x \sim D}[\text{cost}(\hat{RR}, x)] &\leq d \sum_{x \in [d]^n} \text{cost}(\tilde{S}_x, x)\mathbb{P}(x) \\
&\leq d^2 \mathbb{E}_{x \sim D}[\text{cost}(\tilde{S}, x)]
\end{aligned}$$

Recalling our previous result that

$$\mathbb{E}_{x \sim D}[\text{cost}(\tilde{S}, x)] \leq 2\mathbb{E}_{x \sim D}[\text{cost}(S^*, x)]$$

we therefore have that

$$\mathbb{E}_{x \sim D}[\text{cost}(\hat{RR}, x)] \leq 2d^2 \mathbb{E}_{x \sim D}[\text{cost}(S^*, x)]$$

as desired.

□

3.3 Simulations

At the outset of our investigation into the feasibility of using the round robin strategy as an approximation algorithm, we attempted to disprove its approximation guarantee by constructing a counterexample. Recall that S_c is the nonadaptive strategy that orders tests x_i in decreasing order of $\mathbb{P}(x_i = c)$; it is the optimal strategy for determining whether the value c appears in the input vector. Let S_{RR} denote the round robin strategy constructed by interleaving S_c for all $c \in [d]$. The optimal strategy S^*, in contrast, may terminate early in hopeless cases when the remaining number of tests is insufficient to find the representatives needed to satisfy $f(x) = 1$. As a result, the relationship between S_c and S^* is not

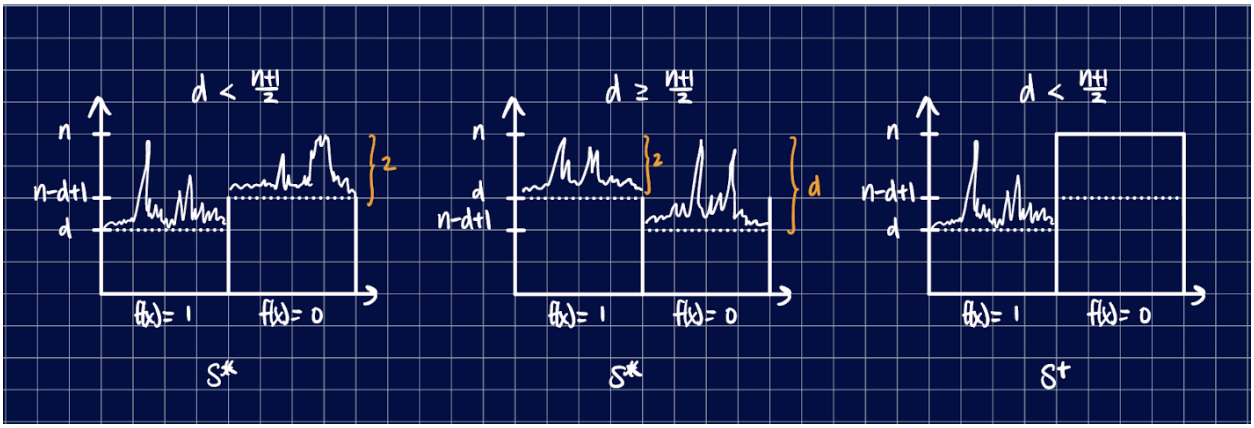

Figure 3: Graphical comparison of optimal strategies for different values of d and n

immediately clear: while S_c focuses on detecting a specific value c, it lacks the flexibility of early termination and is not designed to collect all necessary representatives. Our initial approach was therefore to show that S_{RR} fails to be a d-approximation of S^* by identifying a counterexample.

We include a summary of the strategies employed to find the optimal strategy:

1. There are 2 methods to calculate the expected value of any given strategy. Method 1 is to enumerate all possible vectors $x \in [d]^n$ and calculate its probability. Afterwards, evaluate the cost of running a given strategy S on x, that is, calculate $\text{cost}(S, x)$. Subsequently, weight the cost by the probability $p(x)$ to get the expected cost of the strategy on all $x \sim D$. Method 1 has a runtime of $O(nk^n)$.

2. Method 2 to calculate the expected value of any given strategy is to apply a Markov's process to calculate the expected value. The idea is as follows, let our states be $s \in \{1, 0\}^d$ where s represents the presence of a representative within x. At the start, all states have a probability of being 0 except for the state $s = (0, 0, ..., 0)$ which has a probability of 1. In each iteration, the next variable specified by the strategy S is selected and its probabilities act as edges to transform the probabilities associated with each markov vertex, $p(s)$. This continues until all probabilities completely concentrate on the terminating vertex $s = (1, 1, ..., 1)$, or until we exhaust all tests within S. Along the way, if our algorithm recognizes that any state s has reached a hopeless state, meaning that there is no chance that the $p(s)$ can concentrate on the terminating vertex within the remaining boolean tests, it will drop the vertex s and reset its probabilities to 0. Method 2 has a runtime of $O(n2^d)$.

3. For smaller n and d, it is possible to exhaustively search through $n!$ nonadaptive strategies to identify the optimal order to test variables.

4. To accelerate the search for an optimal strategy for bigger n and d, we adopted a local optimization approach. Given a strategy to test variables in a given order, swap the order of two tests and calculate its expected value. After generating $\binom{n}{2}$ new strategies and testing their expected runtime, update the optimal strategy to be the new strategy

that minimizes the expected cost. Terminate this process upon reaching a strategy that cannot be improved through local optimizations.

4 Conclusion

In sum, we have proposed and analyzed several approaches to evaluating the representative problem. The best adaptive approach achieves an approximation factor of $2d$, whereas the best nonadaptive strategy achieves an approximation factor of $2d^2$.

References

[1] R. Greiner, R. Hayward, M. Jankowska, and M. Molloy.
Finding optimal satisficing strategies for AND-OR trees.
Artificial Intelligence, 170(1):19–58, 2006.

[2] A. Guillory, J. Bilmes.
Active semi-supervised learning using submodular functions
Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, 2011, pp. 274–282.