Black Box Complexity

George Voutsadakis[1]

[1]Mathematics and Computer Science
Lake Superior State University

Seminar Presentation

Lake Superior State University

Black Box Problems

- A **black box problem** formalizes situations in which an unknown function on a known domain, belonging to a known class of functions must be optimized via a series of queries using specific inputs.
- The data available consist of:
  - A problem size n;
  - A search space S_n;
  - A class F_n of functions $f : S_n \to \mathbb{R}$.
- The goal is, without knowledge of the specific $f \in F_n$ under consideration, to find

$$x = \operatorname*{argmax}_{x \in S_n} \{f(x)\}.$$

Example: Traveling Salesperson Version

- In the **traveling salesperson**, a number of cities is given, together with intercity distances, and we are supposed to find a tour of the cities that minimizes the total distance traveled.
- A black box version of this problem is formalized by giving:
  - The number n of the cities $[n] = \{1, \ldots, n\}$ to be visited;
  - The collection S_n of all permutations $\pi : [n] \to [n]$, that represent all possible tours;
  - The collection $F_n : S_n \to \mathbb{R}$ of functions $f_D : S_n \to \mathbb{R}$, where for a (hidden) distance matrix D, f_D assigns to a permutation π the length of the tour represented by π according to D.
- The goal is to choose π that optimizes f_D, without access to the matrix D (which is critical in knowing the "structure" of f_D, i.e., how f_D is computed).

Randomized Search Heuristics

- Consider a black box problem $B = \{F_n : S_n \to \mathbb{R} : n \in \mathbb{N}\}$.
- A **randomized search heuristic** for B proceeds as follows:
  - In Step 1:
    - Selects a probability distribution p_1 on S_n;
    - Selects $x_1 \in S_n$ according to p_1;
    - Computes $f(x_1)$;
  - In Step $t > 1$, assuming knowledge of $(x_1, f(x_1)), \ldots, (x_{t-1}, f(x_{t-1}))$:
    - Selects a new probability distribution p_t on S_n (depending on prior knowledge);
    - Selects x_t according to p_t;
    - Computes $f(x_t)$;
  - At some t, decides (according to some criterion) to stop and outputs the x_i with the optimum $f(x_i)$.
- In specific applications (e.g., local search, evolutionary or genetic algorithms) the t-th step requires only knowledge of $(x_{t-1}, f(x_{t-1}))$.
- In all cases, the value x_i with best $f(x_i)$ must be stored for output.

Expected Optimization Time

- To obtain an accurate estimate of performance, we would have to relate the expected runtime with the probability of success.
- But to simplify analysis we make the following compromises:
  - We assume that the randomized search heuristics never halts.
  - We ignore the number of steps needed to:
    - Compute p_t;
    - Select x_t.
- As a result, we use only the number of calls to the black box in order to compute the **expected optimization time**, i.e.,

  the expected time (number of steps) until an optimal solution is given as a query to the black box.

Black Box Complexity

- Given a black box problem $B = \{F_n : S_n \to \mathbb{R} : n \in \mathbb{N}\}$, its **black box complexity** is the minimal (over all possible randomized search heuristics) worst-case (over all possible functions) expected optimization time.
- To compare black box complexity with ordinary complexity, we note two conflicting trends:
  - The fact that only F_n is known, but not the specific f to be optimized, makes black box complexity more challenging;
  - The fact that we count only the number of calls to the black box (and do not include other computational steps) makes black box complexity easier.

Example: Pseudo-Boolean Polynomials of Degree 2

- A **pseudo-Boolean polynomial of degree 2** is a function $f : \{0,1\}^n \to \mathbb{R}$ that has the form

$$f(x_1, \ldots, x_n) = w_0 + \sum_{1 \le i \le n} w_i x_i + \sum_{1 \le i < j \le n} w_{ij} x_i x_j,$$

  where w_0, w_i and w_{ij} are real constants.
- In this context, we use the following notation:
  - e_0 is the vector consisting of all 0's;
  - e_i is the vector consisting of only one 1 in position i and all other components 0;
  - e_{ij} is the vector with exactly two 1's in positions i and j and all other components 0.

Example: Randomized Search Heuristics

- Consider the black box problem $B = \{F_n : \{0,1\}^n \to \mathbb{R} : n \in \mathbb{N}\}$, where F_n is the class of all pseudo-Boolean polynomials of degree 2.
- We employ the following randomized search heuristic (which is deterministic in this case):
  - Compute $w_0 = f(e_0)$;
  - Compute $w_i = f(e_i) - w_0$;
  - Compute $w_{ij} = f(e_{ij}) - w_0 - w_i - w_j$;
    (By employing exponentially many steps which, however, do not count in black box time, optimize $f(x) = w_0 + \sum_i w_i x_i + \sum_{i,j} w_{ij} x_i x_j$;)
  - Compute $f(x_{\text{opt}})$.
- The algorithm uses

$$1 + n + \binom{n}{2} + 1 = \mathrm{O}\left(n^2\right)$$

black box calls, but it is "undesirable" (due to the exponential cost).

Finite Problems

- Our final goal is to present and apply Yao's Minimax Principle.
- To this end, we restrict black box problems to those where components are finite:
  - The domain of functions S_n is finite.
  - The range of each function is finite, say $\{0, 1, \ldots N\}$.
  - Then the set F_n of all functions $f : S_n \to \{0, 1 \ldots, N\}$ is also finite.
  - The number of all different queries that can be made to the black box $f(x_t)$, $f \in F_n$ and $x_t \in S_n$, is also finite.
  - In conclusion, the number of all possible deterministic search heuristics is finite.

Yao's Game

- Even though in reality there is only one person (the designer of the randomized search heuristic) involved,...
- ...Yao (1977) recast the framework as a two-person, zero-sum game:
  - Alice, the designer of the randomized search heuristic A;
  - Bob, an opponent (adversary) choosing $f \in F_n$.
- For fixed A and f, $T(f, A)$ denotes the expected number of black box calls needed before a call with an optimal search point for f.
  - Alice wants to minimize $T(f, A)$ so as to design the best heuristic;
  - Black box complexity being worst-case with respect to f, Bob wants to maximize $T(f, A)$ by choosing the worst $f \in F_n$.

The Payoff Matrix

- The payoff matrix has a row for each function $f \in F_n$ and a column for each deterministic search heuristic A.
- The (f, A)-entry of the matrix is $T(f, A)$.
- In regards to the game, for a given choice of A and f, Alice pays Bob $T(f, A)$.
- A and f are chosen independently.
- Both Alice and Bob are allowed to use randomized strategies.

Notation for Randomized Strategies

- We let Q be the set of all probability distributions on the set A of deterministic search heuristics.
- For a chosen $q \in Q$, and accompanying choice of A_q according to q, Alice's expected cost for fixed f is $T(f, A_q)$.
- So, for $q \in Q$, Alice's worst-case cost is

$$\max_f T(f, A_q).$$

- We let P be the set of all probability distributions on the set F_n of functions.
- For a chosen $p \in P$, and accompanying choice of f_p according to p, Bob's expected gain for fixed A is $T(f_p, A)$.
- So, for $p \in P$, Alice's best deterministic search heuristic is

$$\min_A T(f_p, A).$$

The Two Player Perspectives

- We have:
$$\begin{array}{rcl} T(f_p, A_q) & = & \sum_{f \in F_n} p(f) T(f, A_q) \\ & \leq & \max_f T(f, A_q). \end{array}$$
- Alice is seeking q^*, such that
$$\begin{array}{rcl} \max_f T(f, A_{q^*}) & = & \min_q \max_f T(f, A_q) \\ & = & \min_q \max_p T(f_p, A_q). \end{array}$$
- We also have:
$$\begin{array}{rcl} T(f_p, A_q) & = & \sum_A q(A) T(f_p, A) \\ & \geq & \min_A T(f_p, A). \end{array}$$
- Bob is seeking p^*, such that
$$\begin{array}{rcl} \min_A T(f_{p^*}, A) & = & \max_p \min_A T(f_p, A) \\ & = & \max_p \min_q T(f_p, A_q). \end{array}$$

Von Neumann's MiniMax and Yao's Minimax Theorems

- Von Neumann's Minimax Theorem (Game Theory) asserts that

$$\max_p \min_q T(f_p, A_q) = \min_q \max_p T(f_p, A_q).$$

The common value v^* is called the **value of the game**.

- In particular, we have

$$v_{\text{Bob}} := \max_p \min_A T(f_p, A) \leq \min_q \max_f T(f, A_q) =: v_{\text{Alice}}.$$

Yao's Minimax Theorem

Let F_n be a finite set of functions on a finite search space S_n, and let $\mathcal{A}$ be a finite set of deterministic algorithms on the problem class F_n. For every probability distribution p on F_n and every probability distribution q on $\mathcal{A}$,

$$\min_{A \in \mathcal{A}} T(f_p, A) \leq \max_{f \in F_n} T(f, A_q).$$

Significance of Yao's Minimax Theorems

- According to Yao's Minimax Theorem
$$\min_{A \in \mathcal{A}} T(f_p, A) \leq \max_{f \in F_n} T(f, A_q).$$
- The expected running time of an optimal deterministic algorithm with respect to an arbitrary distribution on the problem instances is a lower bound for the expected runtime of an optimal randomized algorithm with respect to the most difficult problem instance.
- So the **benefit** is that:

  We get lower bounds for randomized algorithms by proving lower bounds for deterministic algorithms.

Deterministic Search Heuristics as Search Trees

- Let A be a deterministic search heuristic.
- The tree corresponding to A is constructed as follows:
  - At the root is the first query to the black box.
  - Edges out of the root represent possible results to the query.
  - Nodes at the second level represent the second query made, depending on the specific answer to the first query.

    $\vdots$
- Given a specific $f \in F_n$, there exists a unique path starting at the root describing the behavior of the heuristic on f.
- The number of nodes on this path until the first node representing a query to an optimal point for f is equal to the running time of the heuristic on input f.

Example: Needle in a Haystack

- We deal with the following data:
  - The search space $S_n = \{0,1\}^n$;
  - The collection of functions

$$F_n = \{N_a : \{0,1\}^n \to \{0,1\} : a \in \{0,1\}^n\},$$

where

$$N_a(x) = \begin{cases} 1, & \text{if } x = a \\ 0, & \text{if } x \neq a \end{cases}, \qquad a \in \{0,1\}^n.$$

Theorem

The black box complexity of F_n is $2^{n-1} + \frac{1}{2}$.

Example: The Upper Bound

- Consider the following randomized search heuristic:

  Repeat

  Pick (a new) $x \in \{0,1\}^n$ at random;

  Compute $f(x)$;

  Expected optimization time:

  For a fixed a, since all orderings of the queries are equally likely, the probability that N_a will be queried at a at the i-th step is $\frac{1}{2^n}$.

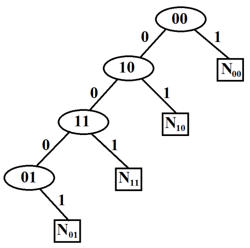
  It follows that the expected optimization time is

$$\begin{aligned} \frac{1}{2^n}\cdot 1+\frac{1}{2^n}\cdot 2+\cdots+\frac{1}{2^n}\cdot 2^n \\ &= \frac{1}{2^n}(1+2+\cdots+2^n) \\ &= \frac{1}{2^n}\frac{2^n(2^n+1)}{2}=2^{n-1}+\frac{1}{2}. \end{aligned}$$

Example: The Lower Bound

- We use Yao's Minimax Theorem.

  We must evaluate, for a given deterministic A, $\min_A T(f_p, A)$ for some arbitrary distribution p on F_n.



Choose as p the uniform distribution on F_n.
There exists an $f \in F_n$ for which A answers 1 at the 2^n-th step after having answered 0's at all previous steps.
On this path every $x \in \{0,1\}^n$ is queried.
And at each level only one query is asked.
The expected optimization time is at least

$$\sum_f p(f)T(f,A) = \frac{1}{2^n}(1+2+\cdots+2^n) = 2^{n-1}+\frac{1}{2}.$$

Thank you!

- In closing...

Thank you for your Attention!!