Yao’s Minimax Principle

Complexity of algorithms

The complexity of an algorithm is usually measured with respect to the size of the input, where size may for example refer to
- the length of a binary word describing the input,
- the number of nodes in an input graph,
- the number of elements in an input list.
Furthermore, one may be interested in
- the worst-case complexity, i.e., the complexity on the worst input of a given size,
- the average-case complexity, i.e., the average complexity over all inputs of the same size.
Finally, the complexity may refer to
- running time, the number of comparisons made, . . . .

Yao’s Minimax Principle

Las Vegas and Monte Carlo algorithms

A randomized algorithm is a
- Las Vegas algorithm if the algorithm is always correct,
- Monte Carlo algorithm if the algorithm may be incorrect.
For a Las Vegas algorithm, the outcomes of the underlying source of randomness have no influence on the correctness of the result, however they may influence the running time or other parameters.

Examples

A typical example of a Las Vegas algorithm is randomized quicksort (i.e., the pivot elements are chosen uniformly at random).
A typical example of a Monte Carlo algorithm is integration by randomized sampling (i.e., in order to determine to a certain precision the area of a geometric figure inside a unit square, pick points in the square uniformly at random and let the area be equal to the fraction of points inside the figure).

Yao’s Minimax Principle

Complexity of problems: lower and upper bounds

The complexity of a problem is specified by lower and upper bounds on the complexity of algorithms that solve the problem:
- lower bounds are obtained by proving (e.g., using combinatorial arguments) that no algorithm can have a complexity smaller than the lower bound under consideration,
- upper bounds are usually obtained by constructing an algorithm for the given problem that is correct and has complexity of at most the upper bound under consideration.

The aim is to find matching lower and upper bounds

If one can derive matching lower and upper bounds for a problem, then the algorithm that yields the upper bound has minimum complexity. Similarly, close lower and upper bounds imply that the complexity of the corresponding algorithm is close to minimum.

Yao’s Minimax Principle

Worst-case complexity of Las Vegas algorithms

In what follows, we consider Las Vegas algorithms and their expected complexity in worst case, that is, the expected complexity on the worst input of any given size.
On first sight, it appears to be hard to obtain good lower bounds on the expected complexity of a Las Vegas algorithm in worst case.
Yao’s Minimax Principle, which is discussed in detail below, asserts that for any problem and
- for any fixed probability distribution on the inputs of some given size, any lower bound on the average-case complexity of deterministic algorithms (where the algorithm may depend on the probability distribution)
  is also a lower bound on the expected complexity in worst case of Las Vegas algorithms.
Inputs, programs, and the cost function

We consider problems with a notion of size for the inputs such that for each size $n$ there are
- a finite set $I_n$ of inputs of this size,
- a finite set $A_n$ of all correct deterministic algorithms for inputs in $I_n$,
- a cost function $k_n : A \times I \rightarrow \mathbb{N}$, where $k_n(A, I)$ is equal to the cost of applying algorithm $A$ to input $I$.

This rather abstract setup is particularly suited for black-box algorithms like randomized Quicksort, which is considered below.

Definition

An algorithm for sorting is a black-box algorithm if an input $x_1, \ldots, x_n$ can only be accessed by queries of the form "$x_i < x_j$?".

Representing Las Vegas algorithms

Yao’s Minimax Principle below does not apply directly to Las Vegas algorithms but to situations where a deterministic algorithm is chosen randomly from the finite set $A$ of all correct deterministic algorithms according to a probability distribution on $A$.

We will apply the principle to Las Vegas algorithms that induce for each $n$ a probability distribution $\sigma_n$ on the set $A_n$ of all correct deterministic algorithms in such a way that all relevant features of the algorithm are represented.

For the induced probability distributions we have that $\text{Prob}_{\sigma_n}[A]$ is the probability that on inputs of size $n$ the Las Vegas algorithm behaves like algorithm $A$ in $A_n$.

As an example, we argue next that a representation by probability distributions on the sets $A_n$ is possible for all black-box Las Vegas sorting algorithms.

Example | Randomized Quicksort
--- | ---
Randomized Quicksort is a black-box algorithm that sorts inputs of pairwise distinct items with respect to a strict linear ordering $<$. Randomized Quicksort works like deterministic Quicksort, but in the recursive calls the pivot element for splitting the current input list is chosen uniformly at random.

The size of an input is equal to the number of items in the list and $A_n$ is the set of all deterministic black-box algorithms that correctly sort lists of $n$ items, $I_n$ can be assumed to be equal to the set of all permutations of the set $\{1, \ldots, n\}$ (because of the black-box access), $k(A, I)$ is equal to the number of queries of the form "$x_i < x_j$?" that algorithm $A$ asks on input $I$.

The set $A_n$ can be chosen to be finite by considering only algorithms that never ask the same query twice and by identifying an algorithm with its behaviour as discussed below.

Yao’s Minimax Principle

Representing Las Vegas sorting algorithms by distributions

When asking only for the expected number of comparisons a given black-box Las Vegas makes when sorting lists of $n$ pairwise distinct items, the relevant features of the Las Vegas algorithm can be represented by a probability distribution $\sigma$ on the set $A$ of correct deterministic black-box algorithms for sorting such lists.

First, since we are only interested in the number of comparisons made, any correct deterministic black-box algorithm for sorting can be identified with its behavior, i.e.,

- with the mapping that determines for any given situation whether another and which comparison is made,
- where by situation we refer to the previously asked queries "$x_i < x_j$?" and their answers.

Note in this connection that the output of a black-box algorithm must be determined by the comparisons made together with the corresponding answers.
Randomized sorting: a lower bound

Representing Las Vegas sorting algorithms by distributions

Since it suffices to consider black-box algorithms that never ask the same query twice, there are only finitely many possible behaviors of such algorithms, hence we can indeed assume that the set $A$ of correct deterministic black-box algorithms is finite.

Then any Las Vegas algorithm can be identified with the probability distribution $\sigma$ on $A$ where the probability of any algorithm $A$ in $A$ is just the probability that the given Las Vegas algorithm and $A$ behave the same (on all possible inputs).

Conversely, any probability distribution $\sigma$ on $A$ can be viewed as a randomized algorithm $A_\sigma$ where on any given input initially some deterministic algorithm is chosen according to $\sigma$.

When going from a Las Vegas algorithm $A$ to the corresponding probability distribution $\sigma$ and then going to the algorithm $A_\sigma$, the initial Las Vegas algorithm $A$ and $A_\sigma$ will have the same probabilities for the various possible behaviors.

Yao’s Minimax Principle

Theorem (Yao’s Minimax Principle)

Let $A$ and $I$ be nonempty finite sets. Let $k : A \times I \rightarrow \mathbb{N}$ be a cost function, and let $\sigma$ and $\tau$ be probability distributions on $A$ and $I$. Let $A_\sigma$ be a random variable with values in $A$ and distribution $\sigma$, and let $I_\tau$ be a random variable with values in $I$ and distribution $\tau$. Then we have

$$\min_{\sigma \in A} \mathbb{E}[k(A_\sigma, I_{\tau})] \leq \max_{I \in I} \mathbb{E}[k(A, I)] . \quad (1)$$

Remark

In Yao’s Minimax Principle, we have in inequality (1) on the left-hand side the average costs on inputs chosen according to $\tau$ for the best deterministic algorithm (which “knows” $\tau$), right-hand side the expected costs for the randomized algorithm determined by $\sigma$ on the worst input in $I$. 

Yao’s Minimax Principle
Yao’s Minimax Principle

Proof of Yao’s Minimax Principle.

In order to demonstrate the theorem, we show for
\[ \tilde{k} = \sum_{(A, l) \in A \times I} \text{Prob}_\sigma[A] \cdot \text{Prob}_\tau[l] \cdot k(A, l) \]
that we have
\[ \min_{A \in A} \mathbb{E}[k(A, l)] \leq \tilde{k} \leq \max_{l \in I} \mathbb{E}[k(A_\sigma, l)]. \]

While this is not used in what follows, observe that in case the random variables \( l_\tau \) and \( A_\sigma \) are mutually independent, we have
\[ \tilde{k} = \mathbb{E}[k(A_\sigma, l_\tau)]. \]

Proposition (Upper bound)

There is a black-box Las Vegas algorithm that correctly decides whether a Boolean matrix has an empty column such that for matrices of size \((n \times n)\) the expected number of queries in worst case is at most
\[ g_{\text{upper}}(n) = \frac{n(n+1)}{2}. \]

Proof.

Consider the algorithm that chooses a permutation \( \pi \) of \( \{1, \ldots, n\} \) uniformly at random and then, until enough information has been obtained, successively checks the columns \( \pi(1), \pi(2), \ldots \) by checking in each column successively the rows \( \pi(1), \pi(2), \ldots \).

On an input that has an empty column, the expected number of checked columns is at most \((n+1)/2\), while on any other input the expected number of checks per column is at most \((n+1)/2\). \( \square \)
Yao's Minimax Principle

**Proposition (Lower bound)**

For any black-box Las Vegas algorithm that correctly decides whether a Boolean \((n \times n)\) matrix has an empty column, the expected number of queries in worst case is at least

\[
g_{\text{lower}}(n) = \frac{n(n+1)}{2}.
\]

**Proof.**

By the discussing above, we can assume that any correct black-box Las Vegas algorithm can be represented by a probability distribution \(\sigma_n\) on \(A_n\), hence the assertion of the proposition can be rephrased as

\[
g_{\text{lower}}(n) \leq \max_{I \in I_n} E[k(A_{\sigma_n}, I)].
\]

Yao's Minimax Principle

**Proof (continued).**

By Yao's Minimax Principle, we have for all probability distributions \(\sigma_n\) on \(A_n\) and \(\tau_n\) on \(I_n\),

\[
\min_{A \in A_n} E[k(A, I_n)] \leq \max_{I \in I_n} E[k(A_{\sigma_n}, I)]. \tag{2}
\]

Thus in order to prove the proposition, it suffices to specify for all \(n\) a probability distribution \(\tau_n\) on \(I_n\) such that the function \(g_{\text{lower}}\) provides lower bounds for the left-hand side of inequality (2). In fact, it suffices to specify for any fixed \(n\) and for any \(\varepsilon > 0\) a probability distribution \(\tau_{\varepsilon}^n\) on \(I_n\) such that

\[
(1 - \varepsilon) g_{\text{lower}}(n) \leq \min_{A \in A_n} E[k(A, i_{\tau_{\varepsilon}^n})].
\]

Yao's Minimax Principle

**Proof (continued).**

In order to specify for given \(n\) and \(\varepsilon > 0\) a probability distribution \(\tau_{\varepsilon}^n\) on \(I_n\),

- let \(D_n\) be the set of all Boolean \((n \times n)\) matrices that have exactly one entry 1 per column.

Then define \(\tau_{\varepsilon}^n\) such that

- the subset \(D_n\) of \(I_n\) has probability \(1 - \varepsilon\) and the distribution within this set is uniform,
- the set \(I_n \setminus D_n\) has probability \(\varepsilon\) and the distribution within this set is again uniform.

Yao's Minimax Principle

**Proof (continued).**

A correct deterministic algorithm for the empty column problem can reject an input in \(D_n\) only after reading at least one symbol 1 in every column.

For a matrix chosen uniformly at random from \(D_n\), for each column the expected number of entries that have to be read before the single 1 is found is

\[
\frac{n}{\sum_{i=1}^{n} i} = \frac{n+1}{2}.
\]

Hence the expected number of entries that have to be read in total is \(\frac{n(n+1)}{2} = g_{\text{lower}}(n)\), where the expectation is with respect to choosing the input uniformly at random from \(D_n\), which is done with probability \((1 - \varepsilon)\). \(\square\)
Excursus on game theory

Matrix games

A matrix game \((A, I, k)\) is played by two players. The first player selects a strategy \(A\) from a finite set \(A\), the second player selects a strategy \(I\) from a finite set \(I\), and the first and second player try to minimize and to maximize, respectively, the payoff \(k(A, I)\).

A matrix game can be represented by a matrix with \(|A|\) rows and \(|I|\) columns that has entries of the form \(k(A, I)\).

As indicated by the notation used above, the situation of Yao’s Minimax Principle can be viewed as a matrix game where the first player selects an algorithm from \(A\) and the second player selects an input from \(I\), while the matrix contains entries of the form \(k(A, I)\).

In technical terms, matrix games are finite two-player zero-sum games with incomplete information.

Mixed strategies

In a matrix game \((A, I, k)\), the strategies in \(A\) and \(I\) are called pure strategies for the first and second player, respectively.

A mixed strategy for a player in a matrix game is a probability distribution on the set of strategies for this player, which we identify with a corresponding random variable \(A_\sigma\) or \(I_\tau\).

When mixed strategies are allowed, the first and second player try to minimize and maximize, respectively, the expected payoff

\[
E[k(A_\sigma, I_\tau)] = \sum_{(A,I)\in(A,I)} \text{Prob}_\sigma[A] \cdot \text{Prob}_\tau[I] \cdot k(A, I).
\]

Example Two finger morra.

In the game of two finger morra each of two players show simultaneously either one or two fingers.

Let \(k \in \{2, 3, 4\}\) be the total count of fingers shown.

In case \(k\) is even, Player I has to pay \(k\) units to Player I,

In case \(k\) is odd, Player I is paid \(k\) units by Player I.

The payoff in this game can be represented by the matrix

\[
\begin{pmatrix}
-2 & 3 \\
3 & -4
\end{pmatrix},
\]

where the rows and columns correspond to the strategies in the set \(A = I = \{1, 2\}\) of Player I and II, respectively, and the entry \(k(i, j)\) in row \(i\) and column \(j\) is equal to the gain of Player I.

Recall the payoff matrix \(K\) for the game of two finger morra

\[
K = \begin{pmatrix}
-2 & 3 \\
3 & -4
\end{pmatrix}.
\]

Any player who plays a known pure strategy will lose.

In case both players play mixed strategies with probabilities of 1/2 for each of their two respective pure strategies, the expected payoff of Player I (and then also of Player II) is

\[
E[k(A_\sigma, I_\tau)] = \sum_{(A,I)\in(A,I)} \text{Prob}_\sigma[A] \cdot \text{Prob}_\tau[I] \cdot k(A, I)
= \frac{1}{4}(-2 + 3 + 3 + -4) = 0.
\]

In fact, one of the players has a better strategy (see exercises).
Definition
For a matrix game \((A, I, k)\), a mixed strategy \(\sigma^*\) for Player I is optimal, if it holds that
\[
\max_{\tau} E[k(A_{\sigma^*}, I_{\tau})] = \min_{\sigma} \max_{\tau} E[k(A_{\sigma}, I_{\tau})]
\] (3)

Similarly, a mixed strategy \(\tau^*\) for Player II is optimal, if it holds that
\[
\min_{\sigma} E[k(A_{\sigma}, I_{\tau^*})] = \max_{\tau} \min_{\sigma} E[k(A_{\sigma}, I_{\tau})]
\] (4)

Remark
In a matrix game, each player has an optimal strategy, which, however, is not necessarily unique. Here it suffices to observe that the expected payoffs, hence also expected payoffs are bounded, and that the set of mixed strategies of each player is compact (in the sense of calculus), hence all minima and maxima in the definition of optimal strategy exist and are attained for appropriate mixed strategies.

Remark
By playing an optimal mixed strategy \(\sigma^*\), Player I can enforce that the expected payoff is at most
\[
k_1^* = \max_{\tau} E[k(A_{\sigma^*}, I_{\tau})] = \min_{\sigma} \max_{\tau} E[k(A_{\sigma}, I_{\tau})],
\]
o matter what mixed strategy Player II plays. Similarly, by playing an optimal mixed strategy \(\tau^*\), Player II can enforce that the expected payoff is at least
\[
k_2^* = \min_{\sigma} E[k(A_{\sigma}, I_{\tau^*})] = \max_{\tau} \min_{\sigma} E[k(A_{\sigma}, I_{\tau})].
\]

Remark
Observe that the values \(k_1^*\) and \(k_2^*\) do not depend on the choice of the optimal strategies \(\sigma^*\) and \(\tau^*\) that occur in their definitions.

Remark
For any matrix game it holds that \(k_2^* \leq k_1^*\). For a proof, consider a situation, where both players play optimal strategies \(\sigma^*\) and \(\tau^*\). Then by definition of optimality, the expected payoff \(k\) is at least \(k_2^*\) and at most \(k_1^*\), hence
\[
k_2^* \leq k \leq k_1^*.
\]

Von Neumann’s celebrated Minimax Theorem asserts that for any matrix game the values \(k_2^*\) and \(k_1^*\) are the same. Before we review this theorem (without proving it), we derive as an easy consequence that matrix games always have equilibrium points.

Remark
Assume \(k_2^* = k_1^*\) and let \(\sigma^*\) and \(\tau^*\) be arbitrary optimal strategies for Player I and II. Then the corresponding expected payoff
\[
k^* = E[k(A_{\sigma^*}, I_{\tau^*})]
\]
is equal to \(k_2^* = k_1^*\), because as already seen the expected gain of a pair of optimal strategies must be between \(k_2^*\) and \(k_1^*\).

Furthermore, the pair \(\sigma^*\) and \(\tau^*\) is an equilibrium point in the sense that the gain of a player cannot be improved by changing this players strategy only. For example, if Player I switches to a different strategy, the new expected payoff is at least \(k_2^* = k^*\) because by assumption \(\tau^*\) is optimal for Player II.
Excursus on game theory

**Von Neumann’s Minimax Theorem**

For any matrix game \((A, I, k)\), it holds that

\[
\max_{\tau} \min_{A} \mathbb{E}[k(A_{\tau}, I_{\tau})] = \min_{\sigma} \max_{I} \mathbb{E}[k(A_{\sigma}, I)]
\]  

(5)

The following reformulation of von Neumann’s Minimax Theorem relies on the fact that the optimum strategy against a fixed mixed strategy can always be chosen to be a pure strategy.

**Variant of von Neumann’s Minimax Theorem**

For any matrix game \((A, I, k)\), it holds that

\[
\max_{\tau} \min_{A} \mathbb{E}[k(A_{\tau}, I_{\tau})] = \min_{\sigma} \max_{I} \mathbb{E}[k(A_{\sigma}, I)]
\]

Proof that the variant is equivalent to the Minimax Theorem.

For arbitrary mixed strategies \(\sigma\) and \(\tau\) it holds that

\[
\mathbb{E}[k(A_{\sigma}, I_{\tau})] = \sum_{(A, I) \in (A, I)} \text{Prob}_{\sigma}[A] \cdot \text{Prob}_{\tau}[I] \cdot k(A, I)
\]

\[
= \sum_{I \in I} \text{Prob}_{\tau}[I] \cdot \mathbb{E}[k(A_{\sigma}, I)] \leq \max_{I \in I} \mathbb{E}[k(A_{\sigma}, I)]
\]

hence for any given \(\sigma\), \(\max_{\tau} \mathbb{E}[k(A_{\sigma}, I_{\tau})] = \max_{I \in I} \mathbb{E}[k(A_{\sigma}, I)]\), where \(\geq\) is trivial and \(\leq\) holds by the preceding discussion. Then

\[
\min_{\sigma} \max_{\tau} \mathbb{E}[k(A_{\sigma}, I_{\tau})] = \min_{\sigma} \max_{I \in I} \mathbb{E}[k(A_{\sigma}, I)]
\]

i.e., the right-hand sides of the equations that asserted in the Minimax Theorem and in its variant have the same value. A similar argument for the left-hand sides then concludes the proof.  

\(\Box\)

---

**Proof**

The variant of von Neumann’s Minimax Theorem yields

\[
\min_{A} \mathbb{E}[k(A, I_{0})] \leq \max_{I \in I} \mathbb{E}[k(A_{\sigma_0}, I)]
\]

---

**Theorem (Equivalent form of Yao’s Minimax Principle)**

Let \((A, I, k)\) be a matrix game and let \(A_{\sigma_0}\) and \(I_{\tau_0}\) be mixed strategies for the first and second player. Then it holds that

\[
\min_{A} \mathbb{E}[k(A, I_{0})] \leq \max_{I \in I} \mathbb{E}[k(A_{\sigma_0}, I)]
\]

Proof.

The variant of von Neumann’s Minimax Theorem yields

\[
\min_{A} \mathbb{E}[k(A, I_{0})] \leq \max_{A} \min_{I} \mathbb{E}[k(A, I_{\tau})] \\
= \min_{\sigma} \max_{I \in I} \mathbb{E}[k(A_{\sigma}, I)] \leq \max_{I \in I} \mathbb{E}[k(A_{\sigma_0}, I)]
\]

\(\Box\)