Algorithm Selection: A Predictive Model for Optimal Sorting

Artem Kiselev

April 5, 2025

1 Introduction

Sorting is a fundamental task that appears across various applications in computer science, from database management to data analytics and real-time processing systems. Due to its critical importance, hundreds of sorting algorithms have been developed, each with unique performance characteristics optimized for particular scenarios. Some algorithms excel at sorting nearly-sorted data, others at handling large datasets, and others at optimizing memory usage. As a consequence, no single algorithm universally outperforms all others for all problem instances. This naturally leads to the following research question:

> *Can we design a model that dynamically and intelligently selects the optimal sorting algorithm based on the characteristics of a given dataset?*

Successfully addressing this question and constructing an effective predictive model would represent a meaningful advancement. Such a model, capable of analyzing a dataset's characteristics and predicting the optimal sorting strategy based on that analysis, could deliver substantial performance improvements over current static approaches. In practical terms, this could significantly enhance efficiency in real-world scenarios, including large-scale database operations, data-intensive computations, and latency-sensitive applications, providing tangible benefits over existing sorting implementations.

2 Problem Formalization

This question is an instance of the algorithm selection problem, formulated by John Rice in 1976 [2]. It is formally stated as follows:

Given:

- A problem space $\mathcal{P}$, containing all possible problem instances.
- A feature space $\mathcal{F}$, where each $f(x) \in \mathcal{F}$ represents measurable characteristics of problem $x \in \mathcal{P}$.
- An algorithm space $\mathcal{A}$, containing all applicable algorithms A_i.
- A performance space $\mathbb{R}^n$, where $p(A_i, x)$ represents the performance of algorithm $A_i \in \mathcal{A}$ on problem $x \in \mathcal{P}$.
- The final algorithm performance metric $||p||$, obtained by normalizing the raw performance measures.

Find: A selection mapping

$$S : \mathcal{F} \to \mathcal{A}$$

that maximizes performance according to some criteria.

In our context, the problem space $\mathcal{P}$ consists of datasets that require sorting. An instance $x \in \mathcal{P}$ is a list to be sorted. The feature space $\mathcal{F}$ comprises measurable characteristics of these datasets, such as size, distribution patterns, Shannon entropy [**?**], and, most importantly, measures of presortedness. These measures tell us the pre-existing order of a given list [1].

The algorithm space $\mathcal{A}$ includes various sorting algorithms like QuickSort, MergeSort, InsertionSort, and others. For simplicity, our performance space is defined as $\{0, 1\}$, where a value of 1 indicates that the selection algorithm has correctly identified the fastest sorter for a given problem instance.

Given:

- Datasets $\mathcal{P} \ni x$, where x is a list that requires sorting.
- A feature space $\mathcal{F}$, where each $f(x) \in \mathcal{F}$ represents measurable characteristics of a list $x \in \mathcal{P}$.
- An algorithm space $\mathcal{A}$, containing all applicable sorting algorithms A_i.
- A performance space $\{0, 1\}$, where the performance metric is defined as

$$p(A_i, x) = \begin{cases} 1, & \text{if } A_i \text{ is the fastest sorter for } x, \\ 0, & \text{otherwise.} \end{cases}$$

Find: A selection mapping

$$S : \mathcal{F} \to \mathcal{A}$$

that maximizes the average number of fastest sorter selections, i.e.,

$$\max_S \frac{1}{|\mathcal{P}|} \sum_{x \in \mathcal{P}} p(S(f(x)), x).$$

3 Methodology

A good measure of presortedness must satisfy the following criteria: *Let m be a measure of presortedness, and X, Y be lists*:

1. $m(X) = 0$ if X is *fully sorted in ascending order.*

2. if $X = [x_1, \cdots, x_n], Y = [y_1, \cdots, y_n]$ and $x_i < x_j$ iff $y_i < y_j$ for all i and j, then $m(X) = m(Y)$.

3. if $X \subseteq Y, m(X) \leqslant m(Y)$.

4. if $X < Y$, then $m(XY) \leqslant m(X) + m(Y)$.

5. *For any element* a, $m(a + X) \leqslant |X| + m(X)$.

References

[1] Heikki Mannila. Measures of presortedness and optimal sorting algorithms. *IEEE Transactions on Computers*, C-34:318–325, 1985.

[2] John R. Rice. The algorithm selection problem. volume 15 of *Advances in Computers*, pages 65–118. Elsevier, 1976.