Analysis of Randomized Incremental Construction for Convex Hulls

Druhan Shah

December 2, 2025

Abstract

This report investigates the efficiency and behavior of randomized algorithms in computational geometry, specifically focusing on the Randomized Incremental Construction (RIC) for Convex Hulls. We contrast this randomized approach with the deterministic Graham Scan algorithm to evaluate the practical implications of randomization. While Graham Scan offers a guaranteed $O(N \log N)$ time complexity, RIC leverages random input permutation to achieve a similar expected time complexity of $O(N \log N)$. Our empirical analysis demonstrates that while RIC performs exceptionally well on uniform random distributions, often surpassing the deterministic baseline, it remains vulnerable to specific worst-case structures when implemented without auxiliary conflict graphs, degrading to $O(N^2)$.

Introduction

Randomized algorithms often provide simpler and more efficient solutions to geometric problems compared to their deterministic counterparts by avoiding worst-case input configurations through random permutations. This project implements and analyzes the Randomized Incremental Construction (RIC) algorithm for the 2D Convex Hull problem. The Convex Hull of a set of points is the smallest convex polygon enclosing the set.

The primary objective is to study the performance characteristics of the RIC algorithm. By comparing it against the standard deterministic Graham Scan algorithm, we aim to highlight the trade-offs between theoretical worst-case guarantees and expected behavior on average inputs.

Algorithm Analysis

The Randomized Incremental Construction builds the hull by adding points one by one in a random order. The algorithm begins with a simple triangle formed by the first three points. For each subsequent point, the algorithm checks if it lies inside the current hull. If the point is external, the algorithm updates the hull by identifying and removing the edges visible from the new point and connecting the new point to the remaining chain via two tangent segments.

The efficiency of RIC relies on the random order of insertion. Backward analysis shows that the probability of a random point causing a structural change to the hull decreases as the hull grows. Consequently, the expected time complexity is $O(N \log N)$. However, a naive implementation that searches the entire current hull for tangents at each step can degrade to $O(N^2)$ if the hull size grows linearly, such as when points lie on a circle.

In contrast, the Graham Scan algorithm is a deterministic method that sorts all points based on their polar angle relative to a pivot. It then iterates through the sorted list, maintaining a convex chain on a stack. The sorting step dominates the runtime, resulting in a consistent $O(N \log N)$ time complexity regardless of the input distribution.

Implementation and Methodology

The algorithms were implemented in C++ to maximize computational efficiency, utilizing double-precision arithmetic for coordinate representation. A key challenge in geometric algorithms is handling floating-point precision; we addressed this by employing an epsilon threshold for all geometric predicates.

For the RIC implementation, a doubly linked list was chosen to represent the hull vertices. This structure supports $O(1)$ insertion and deletion operations, which are frequent during the update phase. The implementation performs a linear traversal of the hull to determine point visibility. While this simplifies the data structure requirements, it introduces a dependency on the instantaneous size of the hull. The Graham Scan implementation utilizes a standard vector and the standard library sorting algorithm to order points angularly, followed by a linear scan using a stack-based approach.

The experimental evaluation was conducted on a Linux environment using the GCC compiler with optimization level 3. We evaluated performance on two distinct datasets: uniformly distributed points in a square, representing the average case, and points arranged on a circle, representing the worst-case scenario for RIC where every point belongs to the hull.

Results and Discussion

The empirical results reveal a stark contrast between the two algorithms depending on the input distribution. On uniformly distributed data, while both algorithms were demonstrably close to linear (theoretically log-linear, but the test sizes don't make it very visible), the RIC implementation demonstrated superior performance, executing in under 0.002 seconds for large inputs. This efficiency arises because the convex hull of uniform random points has a small expected size (logarithmic relative to N), making the linear visibility check negligible. The Graham Scan, while fast, incurs a fixed cost due to sorting, consistently taking approximately 0.003 seconds for similar input sizes.

However, the limitations of the naive RIC implementation became evident with the circular dataset. Since every point on a circle belongs to the convex hull, the hull size grows linearly with the number of inserted points. Consequently, the linear visibility check forces the algorithm into a quadratic runtime behavior, taking nearly 1.4 seconds for the largest inputs. In comparison, the Graham Scan maintained its stability, processing the worst-case input in approximately 0.004 seconds, consistent with its $O(N \log N)$ guarantee.

Conclusion

The study confirms that randomized algorithms like RIC can offer excellent performance for typical inputs, often outperforming deterministic methods due to smaller constant factors and the low probability of expensive structural updates. However, the reliance on randomness does not eliminate the possibility of worst-case behavior if the underlying implementation does not account for specific degenerate cases (e.g., via conflict graphs). For general-purpose applications where input distribution is unpredictable, deterministic algorithms like Graham Scan provide a safer, more consistent performance guarantee.