Habib University

Algorithms: Design and Analysis

Paper Summary and Breakdown

Mahrukh Yousuf (08055)
Hammad Malik (08298)

Paper Title: "New Algorithms for All Pairs Approximate Shortest Paths"

Author Name: Liam Roditty
Conference: 55th Annual ACM Symposium on Theory of Computing (STOC 2023)
Year: 2023
DOI/Link: `https://dl.acm.org/doi/pdf/10.1145/3564246.3585197`

Fast Combinatorial Algorithms for All-Pairs Approximate Shortest Paths

Problem and Contribution

Background

The problem of computing shortest paths in graphs has been foundational as we have studied in depth in this course, with applications ranging from network routing to social network analysis. Exact computation of All Pairs Shortest Paths (APSP) in unweighted graphs can be costly for large graphs, motivating the development of efficient approximate algorithms. A well-studied variant is the **All Pairs Approximate Shortest Paths (APASP)** problem, where the goal is to estimate distances within a guaranteed approximation factor.

An important distinction in approximations is between additive and multiplicative guarantees. While additive approximations bound the difference between estimated and actual distances, multiplicative approximations bound the ratio. In unweighted, undirected graphs, a multiplicative 2-approximation ensures the estimated distance between any two vertices is at most twice the true shortest path. This is important in our paper.

The past work by Dor, Halperin, and Zwick (DHZ) in the late 1990s introduced a non-Fast Matrix Multiplication (FMM) algorithm for additive 2-approximations with runtime $\tilde{O}(\min\{n^{3/2}m^{1/2}, n^{7/3}\})$. This result could be trivially converted into multiplicative 2-approximation for distances ≥ 2. However, no significant improvements had been made in multiplicative approximations under subcubic time without FMM for over two decades.

Problem

This paper focuses on improving the efficiency of solving the APASP problem in unweighted undirectedgraphs for **multiplicative 2-approximations**. The challenge in it lies in beating the longstanding DHZ bounds in a purely combinatorial setting, specifically targeting better performance for large graphs and longer path distances.

Contribution

The paper introduces two algorithms:

1. A faster algorithm for computing multiplicative 2-approximations in $\tilde{O}(\min\{n^{1/2}m, n^{9/4}\})$ time significantly improving on the DHZ result without relying on FMM.

2. A distance sensitive algorithm optimized for vertex pairs with distances ≥ 4, achieving $\tilde{O}(\min\{n^{7/4}m^{1/4}, n^{11/5}\})$ expected time.

These contributions are based on an analysis of the DHZ algorithm's internal workings, identifying opportunities where it inherently provides tighter approximations, and extending this with new approximation schemes tailored to harder instances.

Algorithmic Description

Core Ideas and High-Level Approach

The improved algorithms build on the `apasp`$_k$ framework by Dor, Halperin, and Zwick. The insight lies in recognizing that for many vertex pairs, the DHZ algorithm already performs better than previously realized.

Main components:

- Vertex Hierarchies and Hitting Sets
- Edge Hierarchy
- Blocking Vertices
- Detailed Case Analysis

Algorithm Structure

To summarise our understanding of the algorithm in the paper:

Base Algorithm (`apasp`$_k$):

- INPUT: Unweighted undirected graph $G = (V, E)$
- PROCESS:
  - For each $i = 1$ to k:
    - For each $u \in Z_i$:
      - Run Dijkstra from u in subgraph $H = (V, W(u, V) \cup E^* \cup F_i)$
      - Update distance matrix $M(u, x)$

Post-processing: The algorithm handles different scenarios based on the structure of the shortest paths.

- **Case** C_1: Path contains a vertex from V_1
- **Case** C_2: No vertex in V_1, but at least one in V_2
- **Case** C_3: Neither endpoint in V_2; further divided into subcases

Final Multiplicative 2-Approximation: The algorithm treats vertex pairs differently based on their distances. For pairs at distance 1, it computes the exact value in $O(m)$ time. For those at distances 2 or 3, it applies an additive 2-approximation to estimate the distance efficiently. Finally, for pairs with a distance of 4 or more, it uses an additive 4-approximation to balance accuracy with performance.

Key Technique: Case Analysis for Distance ≤ 3

We consider three different cases for the paths. In **Case** C_1, the path goes through a high degree node from the set V_1. In **Case** C_2, the path does not include any node from V_1, so we use the set Z_2 to help handle it. Finally, in **Case** C_3, the path avoids both V_1 and V_2 so we use more specific strategies to deal with it.

- **Case** C_1: Path contains a high-degree vertex from V_1
- **Case** C_2: No V_1 node in path; use Z_2
- **Case** C_3: No V_1/V_2 nodes involved; apply refined strategies

Comparison with Existing Approaches

Previous State-of-the-Art

Several key results have been proposed for all pairs shortest paths (APSP) approximations. First as mentioned in the paper Dor, Halperin, and Zwick (2000) presented an additive approximation algorithm with a running time of $\tilde{O}(\min\{n^{3/2}m^{1/2}, n^{7/3}\})$. Later, Deng et al. (2022) improved this using fast matrix multiplication (FMM), achieving a time complexity of $\tilde{O}(n^{2.29})$. Baswana, Goyal, and Sen (2007) proposed a simpler algorithm with a running time of $\tilde{O}(n^2)$ for a $(2, 3)$-approximation. Finally, Baswana and Kavitha (2010) developed a method tailored for weighted graphs, which runs in $\tilde{O}(n^2 + m\sqrt{n})$ time.

Advantages of the New Algorithm

The proposed approach offers several key advantages. First, it achieves an improved run time compared to earlier methods. Second, it operates independently of fast matrix multiplication (no dependence on FMM), making it more practical. Third, it ensures comprehensive case coverage, effectively handling all possible scenarios in the input graph. Finally, it incorporates tailored strategies for different ranges of shortest path lengths, leading to effective distance optimization.

Innovation

The work presents several innovative contributions to the field. First, it provides a granular analysis of DHZ, examining its components and implications in unprecedented detail. Second, it proposes a refined classification of shortest path scenarios allowing for targeted algorithmic treatment and more efficient approximations. Finally, the research opens avenues for future work by identifying promising directions for continued investigation and highlighting unexplored aspects of the domain, including possible generalizations to weighted or dynamic graphs.

Data Structures and Techniques

Mathematical Tools

The work relies on a foundation of mathematical tools to support the design and analysis of algorithms. It draws heavily from graph theory to model relationships and define path based properties. Probability is utilized to support randomized algorithms and sampling-based methods. Additionally, combinatorics aids in analyzing complex cases and bounding the behavior of our algorithms.

Data Structures

They use several fundamental and advanced data structures to efficiently manage graph data and compute distances. A distance matrix is employed for storing and retrieving shortest path estimates. Vertex and edge hierarchies help in managing graph decompositions and prioritizing computation. Priority queues are integral to shortest path algorithms such as Dijkstra's, while adjacency lists are used to store graph structure compactly for traversal and updates.

Algorithmic Techniques

To implement the desired functionality, they have leverage a blend of classical and modern algorithmic techniques. Dijkstra's algorithm is used for exact single-source shortest path computation, while BFS is preferred for unweighted graphs and limited distance traversals. Dynamic programming aids in optimizing overlapping subproblems and recomputations. Random sampling is introduced for probabilistic estimations and constructing approximate solutions.

Key Proof Techniques

In the paper, they have employed a variety of proof techniques to establish correctness and performance guarantees. Case based reasoning helps in handling different structural scenarios within the graph. Inductive proofs are used to generalize properties over graph sizes or recursive structures. The triangle inequality serves as a fundamental tool for bounding distances in path approximations. Chebyshev's inequality is applied to derive bounds on the concentration of sampling-based estimators.

Implementation Outlook

Technical Challenges

There are several challenges we must address during implementation. Managing memory at a complexity of $O(n^2)$ can be demanding, especially for dense graphs. Accurately maintaining and updating the sets Z and F is crucial for case resolution and correctness. Ensuring sampling accuracy requires careful calibration of probabilistic thresholds. Parallelizing Dijkstra's algorithm efficiently presents another non-trivial engineering hurdle. Finally, post processing logic needs to be really good and capable of stitching together partial results.

Strategy Recommendations

To mitigate challenges and guide implementation, we thought of a set of strategic recommendations. Start by building on the $\texttt{apasp}_{\texttt{k}}$ baseline which offers a solid and tested foundation. Incrementally add post processing logic to handle specific cases uncovered during theoretical analysis. We may prefer sparse structures where possible to reduce memory consumption. Continuously tune parameters and design choices using benchmark datasets to ensure performance and scalability.

Testing Strategy

Our testing methodology will be essential for validating our implementation. We plan to use synthetic graphs to explore performance under controlled conditions and stress edge cases. Comparing results with existing algorithms will help benchmark correctness and efficiency. Moreover, we will test across diverse graph types including sparse, dense, and random graphs to evaluate robustness and generalisability.