Context-Free Path Queries, Planar Graphs and Friends

Alexandra Istomina
The Thørväld Group
Hekla, Iceland
larst@affiliation.org

Alexandra Olemskaya
Inria Paris-Rocquencourt
Rocquencourt, France

Ekaterina Shemetova
Inria Paris-Rocquencourt
Rocquencourt, France

Semyon Grigorev
Rajiv Gandhi University
Doimukh, Arunachal Pradesh, India

ABSTRACT

Abstract is very abstract.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Alexandra Istomina, Alexandra Olemskaya, Ekaterina Shemetova, and Semyon Grigorev. 2018. Context-Free Path Queries, Planar Graphs and Friends. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Context-Free Path Querying (CFPQ) is a sublcass of Language-constrained path problem, where language is set to be Context-Free.

Importance of CFPQ. Application areas. RDF, Graph database querying, Graph Segmentation in Data provenance, Biological data analysis, static code analysis.

1.1 An Example

Example of graph and query. Should be used in explataion below.

1.2 Existing CFPQ Algorithms

Number of problem-specific solutions in static code analysis.

Hellings, Ciro et al, Kujpers, Sevon, Verbitskaya, Azimov, Ragozina

1.3 Existing Theoretical Results

Existing theoretical results

Linear input. Valiant [11], Lee [9].

Yannacacis [12]? Reps?

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

Bradford [1]
RSM [4].
C alias analysis [13]
Chatterjee [3]
For trees
Truly-subcubic for Language Editing Distance [2].
Truly-subcubic algorithm is stil an open problem.

1.4 Our Contribution

This paper is organized as follows. !!!!!

2 PRELIMINARIES

We introduce !!!!

2.1 Context-Free Path Querying

Graph, grammar, etc.

Let $i\pi j$ denote a unique path between nodes i and j of the graph and $l(\pi)$ denotes a unique string which is obtained from the concatenation of edge labels along the path π. For a context-free grammar $G = (\Sigma, N, P, S)$ and directed labelled graph $D = (Q, \Sigma, \delta)$, a triple (A, i, j) is *realizable* iff there is a path $i\pi j$ such that nonterminal $A \in N$ derives $l(\pi)$.

2.2 Tensor-Based algorithm for CFPQ

Algorithm 1 Kronecker product context-free recognizer for graphs

1: **function** contextFreePathQuerying(D, G)

2.3 Planar Graphs

A planar graph $G = (V, E)$ is a graph that can be embedded in the plane.

Outer face - unbounded face in specific embedding.

Directed graph

...

2.4 Dynamic reachability algorithms

We consider only algorithms that solve problem of reachability in planar directed graphs (*digraphs*). In the *dynamic reachability problem* we are given a graph G subject to edge updates (insertions or deletions) and the goal is to design a data structure that would allow answering queries about the existence of a path.

We need to answer the queries of type: "Is there a directed path from u to v in G?". If vertex u in all the queries is fixed we say that

algorithm is *single-source*. It is said to be *all-pairs* if vertices u, v can be any vertices of planar digraph G, it is also called *dynamic transitive closure*.

We say that the algorithm is *fully dynamic* if it supports both additions and deletions of edges. It is said to be *semi dynamic* if it supports only one of them. If semi dynamic algorithm supports additions only it is called *incremental*, if deletions only - *decremental*.

3 CFPQ ON PLANAR DIGRAPHS

We want to consider algorithms for semi dynamic transitive closure for the case of planar graph. Our algorithm needs to support only edge insertions.

There are several algorithms that look into the problem of dynamic reachability in planar digraphs and have sub linear both update and query time [10], [8], [7].

Some algorithms [5], [8] restrict edge insertions so that they respect the specific embedding of the planar graph.

maybe we can choose one embedding at random and hope that nothing will violate it. So it will be monte-carlo (why it will have good probability? where from can we get all the possible non-isomorphic embeddings?)

The algorithms that allow edge insertions not with respect to the embedding of the graph, but to the planarity, are described in [10], [6]. The main idea of both articles is the same: planar graph in separated into *clusters* - edge induced sub-graphs - for which reachability is shown by their sparse substitution S. *Sparse substitution* is a graph that contains the reachability information between some set of vertices that lie in the outer face of the graph.

We show the results of [10] more closely as they differ from the results of [6] only by logarithmic factor, have the same idea and are easier for understanding.

Updates and queries are the following: adding or deleting the edge between vertices u, v, checking reachability, checking if new edge (u, v) violates planarity. After splitting graph into clusters and looking into its sparse substitute update or query can be done by placing into the graph of sparse substitutes clusters of the corresponding vertices $u, v \in V$.

We only need planarity for each cluster and will build an embedding for each of them at the beginning and after every $O(n^{\frac{1}{3}})$ edge-insertions.

The results are the following: the amount of time for preprocessing is $O(n \log n)$ - in this time we find separation of the given graph G into $O(n^{\frac{1}{3}} \log n)$ clusters and build sparse substitution sub-graph S for this clusters. After that we can perform add operation in $O(n^{\frac{2}{3}} \log n)$ amortized time, delete operation in $O(n^{\frac{2}{3}} \log n)$ worst-case time, check if graph is planar in $O(n^{\frac{1}{2}})$ amortized time.

Theorem 3.1. *We can maintain the structure described in [10] not only for planar graphs, but for planar graphs with $O(n^{\frac{1}{3}})$ edges that violate planarity.*

Proof. In [10] number of clusters for each of which we need planarity is $O(n^{\frac{1}{3}})$. After every $O(n^{\frac{1}{3}})$ edge insertions we rebuild sparse substitute graph S. It takes $O(n^{\frac{2}{3}} \log n)$ time, that is distributed over all $O(n^{\frac{1}{3}})$ insertions. We can additionally maintain the set of non-planar edges, each edge of this set will present its own cluster and will not take part in rebuilding of sparse substitution. Edge is *non-planar* if its insertion to current planar graph G makes in non-planar. □

Let us look closely on how can we use the power of sparse substitution on our incremental transitive closure problem.

Theorem 3.2. *We can add edges in graph and print out new pairs of reachable vertices for every insertion in $O(n^{\frac{5}{3}})$ total time for every sequence of insertions if our model allow parallel computations.*

Proof. From [10] we can add an edge in $O(n^{\frac{2}{3}})$ amortized time. After that we can run DFS from the ends u, v of the edge: DFS from v and DFS on reversed edge from u. We want to get all pairs of vertices that are now connected through the new edge. DFS runs in time proportional to the size of the graph of sparse substitution S - $O(n^{\frac{2}{3}} \log n)$. After that for every cluster in original graph G we create dummy vertex s and edges from s to all boundary vertices in cluster and run DFS from this dummy vertex s. If any vertex w is reachable from v (without loss of generation), then it is a boundary vertex of lie in some cluster and is reachable from some boundary vertex of this cluster. In both cases one of DFS's will print it out.

If we can run these DFS's in parallel (there are $O(n^{\frac{1}{3}})$ clusters and the same number of DFS's), then each of them will take $O(n^{\frac{2}{3}})$ amount of time (all cluster have $O(n^{\frac{2}{3}})$ edges by definition in [10]). In this case total amount of time will be $O(n^{\frac{2}{3}})$. □

If our model can not run algorithms in parallel, then the amount of time taken by iterating these DFS's for every cluster is linear - $O(n)$. This means that usual DFS on the original graph G has the same complexity and we will spend in total $O(n^2)$ time and that planarity does not get us any advantage in solving the problem of dynamic reachability (in amortized time per edge and query).

maybe we can spare some space ($O(n^2)$?)? so we can store list of reachable vertices from any boundary vertex and somehow update them during the edge additions?

4 CONCLUSION

Conclusion and future work.

Efficient implementation?

!!!

REFERENCES

[1] P. G. Bradford. 2017. Efficient exact paths for dyck and semi-dyck labeled path reachability (extended abstract). In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. 247–253. https://doi.org/10.1109/UEMCON.2017.8249039

[2] Karl Bringmann, Fabrizio Grandoni, Barna Saha, and Virginia Vassilevska Williams. 2019. Truly Subcubic Algorithms for Language Edit Distance and RNA Folding via Fast Bounded-Difference Min-Plus Product. *SIAM J. Comput.* 48, 2 (2019), 481–512. https://doi.org/10.1137/17M112720X arXiv:https://doi.org/10.1137/17M112720X

[3] Krishnendu Chatterjee, Bhavya Choudhary, and Andreas Pavlogiannis. 2017. Optimal Dyck Reachability for Data-Dependence and Alias Analysis. *Proc. ACM Program. Lang.* 2, POPL, Article 30 (Dec. 2017), 30 pages. https://doi.org/10.1145/3158118

[4] Swarat Chaudhuri. 2008. Subcubic Algorithms for Recursive State Machines. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Francisco, California, USA) *(POPL '08)*. Association

for Computing Machinery, New York, NY, USA, 159–169. https://doi.org/10.1145/1328438.1328460
[5] Krzysztof Diks and Piotr Sankowski. 2007. Dynamic plane transitive closure. In *European Symposium on Algorithms*. Springer, 594–604.
[6] Zvi Galil, Giuseppe F Italiano, and Neil Sarnak. 1999. Fully dynamic planarity testing with applications. *Journal of the ACM (JACM)* 46, 1 (1999), 28–91.
[7] Ming-Yang Kao. 2008. *Encyclopedia of algorithms*. Springer Science & Business Media. 342–343 pages.
[8] Adam Karczmarz. 2018. *Data structures and dynamic algorithms for planar graphs*. Ph.D. Dissertation. PhD thesis, University of Warsaw.
[9] Lillian Lee. 2002. Fast Context-free Grammar Parsing Requires Fast Boolean Matrix Multiplication. *J. ACM* 49, 1 (Jan. 2002), 1–15. https://doi.org/10.1145/505241.505242
[10] Sairam Subramanian. 1993. A fully dynamic data structure for reachability in planar digraphs. In *European Symposium on Algorithms*. Springer, 372–383.
[11] Leslie G. Valiant. 1975. General Context-free Recognition in Less Than Cubic Time. *J. Comput. Syst. Sci.* 10, 2 (April 1975), 308–315. https://doi.org/10.1016/S0022-0000(75)80046-8
[12] Mihalis Yannakakis. 1990. Graph-theoretic methods in database theory. In *Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. ACM, 230–242.
[13] Qirun Zhang, Xiao Xiao, Charles Zhang, Hao Yuan, and Zhendong Su. 2014. Efficient Subcubic Alias Analysis for C. *SIGPLAN Not.* 49, 10 (Oct. 2014), 829–845. https://doi.org/10.1145/2714064.2660213