Improved Time-Space Bounds for Grid Graph Reachability

Rahul Jain[1] **and Raghunath Tewari**[2]

1 **Indian Institute of Technology Kanpur**
`jain@cse.iitk.ac.in`
2 **Indian Institute of Technology Kanpur**
`rtewari@cse.iitk.ac.in`

Abstract

The reachability problem is to determine if there exists a path from one vertex to the other in a graph. Grid graphs are the class of graphs where vertices are present on the lattice points of a two-dimensional grid, and an edge can occur between a vertex and its immediate horizontal or vertical neighbor only.

Asano et al. presented the first simultaneous time space bound for reachability in grid graphs with n vertices by presenting an algorithm that solves the problem in polynomial time and $O(n^{1/2+\epsilon})$ space [2]. In 2018, the space bound was improved to $\tilde{O}(n^{1/3})$ by Ashida and Nakagawa [4]. In this paper, we further improve the space bound and present a polynomial time algorithm that uses $O(n^{1/4+\epsilon})$ space to solve reachability in a grid graph.

1 Introduction

Given a graph G and two vertices s and t in it, the *reachability problem* seeks to answer whether there exists a path from s to t in G. It is of fundamental importance in the field of computer science. Not only is it used as a subroutine in many graph algorithms, but its study also gives insights into space bounded computations. Reachability in a directed graph is complete for the class of problems solvable by a nondeterministic Turing machine in logspace. A deterministic logspace algorithm for it would show NL to be equal to L, thus solving an open question in the area of computational complexity. In an undirected graph, reachability was shown to be in L by Reingold [11].

Standard graph traversal algorithms solve reachability in directed graphs using linear time and space. We also know of Savitch's algorithm which can solve reachability in $O(\log^2 n)$ space but the algorithm requires $2^{\Omega(\log^2 n)}$ time [12]. So, on one end we have algorithms that require a small amount of time and a large amount of space, while on the other end, we have Savitch's algorithm which requires a small amount of space but a large amount of time. A natural question we can ask is whether there exists an algorithm that uses time and space both in small amounts. This question was formally asked by Wigderson in his survey of reachability problems, if there exists an algorithm for graph reachability which maintains the polynomial time bound while running in $O(n^{\epsilon})$ space, for some $\epsilon < 1$ [13].

Barnes, Buss, Ruzzo, and Schieber gave the first sublinear space polynomial time algorithm for reachability in directed graph [5]. The space complexity of their algorithm is $n/2^{\Theta(\sqrt{\log n})}$. We know of polynomial time algorithms with better space complexity for various subclasses of directed graphs. These include planar graphs [9][3], genus g graphs, H-minor-free graphs, $K_{3,3}$-minor-free graphs, K_5-minor free graphs [7], Layered planar graphs [8] and Unique path graphs [10].

Our concern here is with grid graphs. Grid graphs are a subclass of planar graphs. Reachability in planar graphs belongs to a subclass of NL called unambiguous logspace UL [6]. Reachability in planar graphs can be reduced to reachability in grid graphs in logspace

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.

[1]. Asano and Doerr presented a polynomial time algorithm that uses $O(n^{1/2+\epsilon})$ space for solving reachability in grid graphs [2]. Ashida and Nakagawa presented an algorithm with improved space complexity of $\tilde{O}(n^{1/3})$ [4]. Ashida and Nakagawa's algorithm proceeded by first dividing the input grid graph into subgrids. It then used a gadget to transform each subgrid into a planar graph, making the whole of the resultant graph planar. Finally, it used the planar reachability algorithm of Imai et al. [9] as a subroutine to get the desired space bound.

In this paper, we present an algorithm with a space complexity of $O(n^{1/4+\epsilon})$, thereby improving the bound of Ashida and Nakagawa.

▶ **Theorem 1.1.** *For every $\epsilon > 0$, there exists a polynomial time algorithm that decides reachability in grid graphs using $O(n^{1/4+\epsilon})$ space.*

Our algorithm works by first dividing the grid into subgrids. It then recursively solve each grid to get an *auxiliary graph*. It then solves this auxiliary graph by using a space efficient subroutine that we develop for it.

2 Preliminaries

We denote the vertex set of a graph G by $V(G)$ and its edge set by $E(G)$. For a subset U of $V(G)$, we denote the subgraph of G induced by the vertices of U as $G[U]$. For a graph G, we denote the set of all its connected components by $\mathsf{cc}(G)$. For an edge $e = (u, v)$, we let $\mathsf{tail}(e)$ be u and $\mathsf{head}(e)$ be v.

In a *drawing* of a graph on a plane, each vertex is mapped to a point of the plane, and each edge is mapped to a simple arc whose endpoints coincide with the mappings of the end vertices of the edge. Also, the interior of an arc for an edge does not contain any other vertex points.

We call a graph G an $N \times N$ grid graph if its vertices can be drawn on coordinates (i, j) where $0 \le i, j \le N$ and for all edges of G, their end vertices are at unit distance.

3 Main Result

For simplicity of discussion, we begin with an $N \times N$ grid graph and present a polynomial time algorithm with a space complexity $O(N^{1/2+\epsilon})$. When measured in terms of the number of vertices n of the graph, this space complexity translates to $O(n^{1/4+\epsilon})$.

3.1 Auxiliary Graph

For a parameter $\alpha < 1$, we first define the α-*auxiliary graph* G^α of a grid graph G. We divide our grid graph G into $N^{2\alpha}$ subgrids such that each subgrid is a $N^{1-\alpha} \times N^{1-\alpha}$ grid as shown in Figure 1a. Let $G^\alpha_{i,j}$ be the graph obtained by solving reachability in the subgrid in the i-th row and j-th column. We obtain G^α by replacing each subgrid by its corresponding solved blocks. Since each of the subgrids contains $4N^{1-\alpha}$ vertices on its boundary, the total number of vertices in G^α is at most $4N^{1+\alpha}$. An example of G^α is shown in Figure 1b. For the rest of this article, for any i and j, we will call the graph $G^\alpha_{i,j}$ a *block* of G^α. Note that G^α might have parallel edges. However, each such edge will belong to a different block of G^α.

Our algorithm for reachability constructs G^α by solving the $N^{1-\alpha} \times N^{1-\alpha}$ grids recursively. It then uses a polynomial time subroutine which solves G^α. Note that we do not store the graph G^α explicitly, since that would require too much space. Instead, we solve a subgrid recursively everytime the subroutine queries for an edge in that subgrid of G^α.

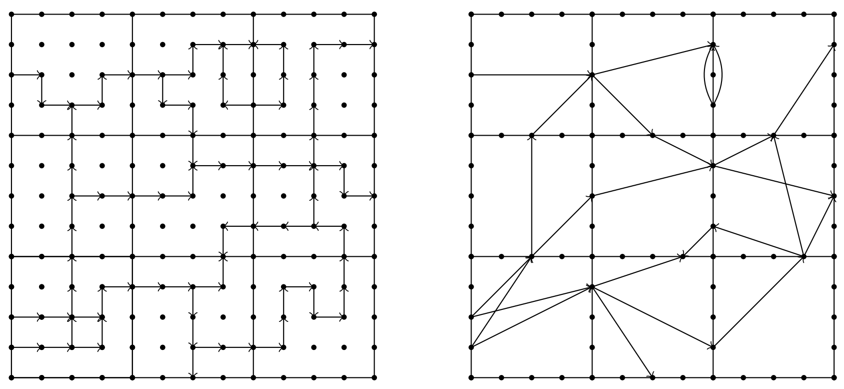

(a) Graph divided into subgrids **(b)** Auxiliary Graph

Figure 1 An example of a grid graph G and the corresponding auxiliary graph G^{α}

Our strategy is to show that for every small positive constant β, there exists a polynomial time algorithm which solves reachability in G^{α} using $\tilde{O}(\tilde{n}^{1/2+\beta/2})$ space where $\tilde{n}$ is the number of vertices in G^{α}. As discussed earlier, $\tilde{n}$ can be at most $4N^{1+\alpha}$. Hence, the main algorithm would require $\tilde{O}(N^{1/2+\beta/2+\alpha/2+\alpha\beta/2}) = O(N^{1/2+\epsilon})$ space.

3.2 Properties of the Auxiliary Graph

The auxiliary graph that we construct might not be planar; there can be edges that cross each other. However, since we construct it from a grid graph, we can make the following essential observations about the auxiliary graph G^{α}.

▶ **Observation 3.1.** *If two edges e and f of an auxiliary graph G^{α} cross each other, then the graph G also has the edges $(\mathit{tail}(e), \mathit{head}(f))$ and $(\mathit{tail}(f), \mathit{head}(e))$.*

▶ **Observation 3.2.** *If two edges e_1 and e_2 crosses a certain edge f, and e_1 is closer to $\mathit{tail}(f)$ than e_2, then the edge $(\mathit{tail}(e_1), \mathit{head}(e_2))$ is also present in the graph G^{α}.*

3.3 Constructing a pseudoseparator

Imai et al. used a separator construction to solve the reachability problem in planar graphs [9]. A separator is a *small* set of vertices whose removal disconnects the graph into *smaller components*. An essential property of a separator is that, for any two vertices, a path between the vertices must contain a separator vertex if the vertices lie in two different components with respect to the separator.

Unfortunately the graph G^{α} might not have a small separator. However, G^{α} has a different kind of separator, which we call a pseudoseparator (see Definition 3.3). The pseudoseparator allows us to decide reachability in G^{α}, in an efficient manner and obtain the claimed bounds.

▶ **Definition 3.3.** Let G be a grid graph and H be a vertex-induced subgraph of G^{α} with h vertices. Let C be a subgraph of H and $H^{(C)}$ be the subgraph of H formed by removing all the vertices of C and all the edges which crosses an edge of C. Let $f : \mathbb{N} \to \mathbb{N}$ be a function. We call C an $f(h)$-pseudoseparator if the size of every connected component in $cc(H^{(C)})$ is at most $f(h)$. The size of C is the total number of vertices and edges of C summed together.

For a vertex-induced subgraph H of G^{α}, an $f(h)$-pseudoseparator is a subgraph of H that has the property that, if we remove the vertices as well as all the edges which cross one of the edges of pseudoseparator, the graph gets disconnected into small pieces. Now, any path which connects two vertices in different components, must either contain a vertex of pseudoseparator or must contain an edge that crosses an edge of pseudoseparator (see Observation 3.4). We divide the graph using this pseudoseparator and show an algorithm which recursively solves each subgraph and then combines their solution efficiently using the above observations.

▶ **Observation 3.4.** *Let G be a grid graph and let H be a vertex-induced subgraph of G^α. Let C be a subgraph of H. Then the following holds:*

1. *For any two distinct U_1 and U_2 of $\mathsf{cc}(H^{(C)})$, $U_1 \cap U_2 = \emptyset$.*
2. *$V(C) \cup (\bigcup_{U \in \mathsf{cc}(H^{(C)})} U) = V(H)$*
3. *For every edge e in H, if there exist distinct sets U_1 and U_2 in $\mathsf{cc}(H^{(C)})$ such that one of the endpoints of e is in U and the other is in U_2, then there exists an edge f in C such that e crosses f.*

We construct a pseudoseparator using the next lemma.

▶ **Lemma 3.5.** *Let G be a grid graph, and let H be a vertex-induced subgraph of G^α with h vertices. For any constant $\beta > 0$, there exists an algorithm which takes H as input and outputs an $(h^{1-\beta})$-pseudoseparator of H of size $O(h^{1/2+\beta/2})$ in $\tilde{O}(h^{1/2+\beta/2})$ space and polynomial time.*

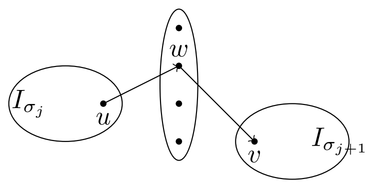
We briefly comment on how to construct a pseudoseparator of a vertex-induced subgraph H of G^α. First, we pick, in logspace, a maximal subset of edges from H so that no two edges *cross*. Then, we triangulate the resulting graph and use Imai et al.'s algorithm to find its separator. Call the triangulated graph $\widehat{H}$ and the set of separator vertices S. The vertex set of pseudoseparator of H contains all the vertices of S and four additional vertices for each edge of $\widehat{H}[S]$ that is not present in H. The edge set of pseudoseparator of H contains all edges of H which are also in $\widehat{H}[S]$ and four additional edges for each edge of $\widehat{H}[S]$ that is not present in H.

3.4 Sketch of an Algorithm to Solve Reachability in the Auxiliary Graph

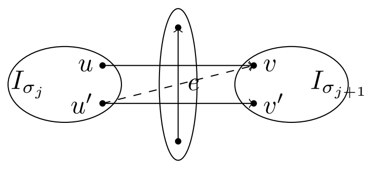
Given a vertex-induced subgraph H of G^α, we first construct its pseudoseparator using Lemma 3.5. Call this pseudoseparator C. We ensure that s and t are part of the pseudoseparator. Let $I_1, I_2, \ldots, I_l$ be the components received after dividing the graph using pseudoseparator. The subroutine performs a loop with $|H|$ iterations and updates a set of marked vertices. Initially, it marks the vertex s. After an iteration, it marks a vertex of C if there is a path from a marked vertex to it such that the internal vertices of that path all belong to only one of the components I_i. Also, for each edge e of C, the vertex v closest to $\mathsf{tail}(e)$ which satisfies the following two conditions is marked: (i) There exists an edge f which cross e and $\mathsf{tail}(f) = v$, and (ii) there is a path from a marked vertex to v such that the internal vertices of the path all belong to only one of the components I_i.

Let P be the shortest path from s to t in H. Suppose P passes through the components $I_{\sigma_1}, I_{\sigma_2}, \ldots, I_{\sigma_L}$ in this order. The length of this sequence can be at most $|H|$. As the path leaves the component I_{σ_j} and goes into $I_{\sigma_{j+1}}$, it can do this in the following two ways:

1. The path exits I_{σ_j} through a vertex of pseudoseparator as shown in Figure 2a. In this case, our algorithm would mark the vertex w.
2. The path exits I_{σ_j} through an edge (u, v) whose other endpoint is in $I_{\sigma_{j+1}}$. By Observation 3.4, this edge will cross an edge e of the pseudoseparator. In this case, the algorithm



(a) The s-t path contains a vertex of the separator



(b) The s-t path crosses an edge of the separator

Figure 2 Types of crossing of an s-t path with the separator.

would mark the vertex u' which is closer than u to $\mathsf{tail}(e)$ and an edge (u', v') crosses e. By Observation 3.2, the edge (u', v) would be present in the graph.

Thus after j iteration, the subroutine would traverse the fragment of the path in the component I_{σ_j} and either mark its endpoint or a vertex which is closer to the edge e of C which the path crosses. Finally, t will be marked after L iterations if and only if there is a path from s to t in H.

3.5 Complexity of the Algorithm

Our subroutine solves reachability in a subgraph H (having size h) of G^α. We do not explicitly store a component of $\mathsf{cc}(H^{(C)})$, since it might be too large. Instead, we identify a component with the lowest indexed vertex present in it and use Reingold's algorithm on $H^{(C)}$ to determine if a vertex is present in that component. We require $\tilde{O}(h^{1/2+\beta/2})$ space to calculate pseudoseparator by Lemma 3.5. We can potentially mark all vertices of pseudoseparator and for each edge of pseudoseparator we mark at most one additional vertex. Since the size of pseudoseparator is at most $O(h^{1/2+\beta/2})$, we require $\tilde{O}(h^{1/2+\beta/2})$ space. The algorithm recurses on a graph with $h^{1-\beta}$ vertices. Hence the depth of the recursion is $1/\beta$, which is a constant. The total space complexity is thus $\tilde{O}(\tilde{n}^{1/2+\beta/2})$.

Since the graph H is given implicitly in our algorithm, there is an additional polynomial overhead involved in obtaining its vertices and edges. However, the total time complexity remains a polynomial in the number of vertices since the recursion depth is constant.

References

1 Eric Allender, David A Mix Barrington, Tanmoy Chakraborty, Samir Datta, and Sambuddha Roy. Planar and grid graph reachability problems. *Theory of Computing Systems*, 45(4):675–723, 2009.

2 Tetsuo Asano and Benjamin Doerr. Memory-constrained algorithms for shortest path problem. In *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry (CCCG 2011)*, 2011.

3 Tetsuo Asano, David Kirkpatrick, Kotaro Nakagawa, and Osamu Watanabe. $\tilde{O}(\sqrt{n})$-space and polynomial-time algorithm for planar directed graph reachability. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS 2014)*, pages 45–56, 2014.

4 Ryo Ashida and Kotaro Nakagawa. $\tilde{O}(n^{1/3})$-space algorithm for the grid graph reachability problem. In *Proceedings of the 34th International Symposium on Computational Geometry (SoCG 2018)*, pages 5:1–5:13, 2018.

5 Greg Barnes, Jonathan F. Buss, Walter L. Ruzzo, and Baruch Schieber. A sublinear space, polynomial time algorithm for directed s-t connectivity. *SIAM Journal on Computing*, 27(5):1273–1282, 1998.

6 Chris Bourke, Raghunath Tewari, and NV Vinodchandran. Directed planar reachability is in unambiguous log-space. *ACM Transactions on Computation Theory (TOCT)*, 1(1):4, 2009.

7 Diptarka Chakraborty, Aduri Pavan, Raghunath Tewari, N. V. Vinodchandran, and Lin F. Yang. New time-space upperbounds for directed reachability in high-genus and h-minor-free graphs. In *Proceedings of the 34th Annual Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS 2014)*, pages 585–595, 2014.

8 Diptarka Chakraborty and Raghunath Tewari. An $O(n^\epsilon)$ space and polynomial time algorithm for reachability in directed layered planar graphs. *ACM Transactions on Computation Theory (TOCT)*, 9(4):19:1–19:11, 2017.

9 Tatsuya Imai, Kotaro Nakagawa, Aduri Pavan, N. V. Vinodchandran, and Osamu Watanabe. An $O(n^{\frac{1}{2}+\epsilon})$-space and polynomial-time algorithm for directed planar reachability. In *Proceedings of the 28th Conference on Computational Complexity (CCC 2013)*, pages 277–286, 2013.

10 Sampath Kannan, Sanjeev Khanna, and Sudeepa Roy. STCON in Directed Unique-Path Graphs. In *Proceedings of the 28th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2008)*, volume 2, pages 256–267, Dagstuhl, Germany, 2008. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

11 Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):17, 2008.

12 Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

13 Avi Wigderson. The complexity of graph connectivity. In *Proceedings of the 17th International Symposium on Mathematical Foundations of Computer Science (MFCS 1992)*, pages 112–132. Springer, 1992.