An $O(n^{1/4+\epsilon})$ Space and Polynomial Algorithm for Grid Graph Reachability

Rahul Jain
Indian Institute of Technology Kanpur, India
jain@cse.iitk.ac.in

Raghunath Tewari
Indian Institute of Technology Kanpur, India
rtewari@cse.iitk.ac.in

Abstract

The reachability problem is to determine if there exists a path from one vertex to another 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 by presenting an algorithm that solves the problem in polynomial time and $O(n^{1/2+\epsilon})$ space. In 2018, the space bound was improved to $\tilde{O}(n^{1/3})$ by Ashida and Nakagawa.

In this paper, we show that reachability in an n vertex grid graph can be decided by an algorithm using $O(n^{1/4+\epsilon})$ space and polynomial time simultaneously.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases graph reachability, grid graph, graph algorithm, sublinear space algorithm

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.19

Funding *Rahul Jain*: Ministry of Human Resource Development, Government of India
Raghunath Tewari: DST Inspire Faculty Grant, Visvesvaraya Young Faculty Fellowship

Acknowledgements We thank anonymous reviewers for their helpful comments on an earlier version of this paper.

1 Introduction

The problem of graph reachability is to decide whether there is a path from one vertex to another in a given graph. This problem has several applications in the field of algorithms and computational complexity theory. Reachability in directed and undirected graphs capture the complexity of nondeterministic and deterministic logarithmic space respectively [12]. It is often used as a subroutine in various network related problems. Hence designing better algorithms for this problem is of utmost importance to computer scientists.

Standard graph traversal algorithms such as DFS and BFS give a linear time algorithm for this problem, but they require linear space as well. Savitch's divide and conquer based algorithm can solve reachability in $O(\log^2 n)$ space, but as a tradeoff, it requires $n^{O(\log n)}$ time [13]. Hence it is natural to ask whether we can get the best of both worlds and design an algorithm for graph reachability that runs in polynomial time and uses polylogarithmic space. Wigderson asked a relaxed version of this question in his survey - whether graph reachability can be solved by an algorithm that runs simultaneously in polynomial time and uses $O(n^{1-\epsilon})$ space [15]. In this paper, we address this problem for a certain restricted class of directed graphs.

Barnes et al showed that reachability in general graphs can be decided simultaneously in $n/2^{\Theta(\sqrt{\log n})}$ space and polynomial time [6]. Although this algorithm gives a sublinear space bound, it still does not give a positive answer to Wigderson's question.

However, for certain topologically restricted graph classes, we know of better space bounds simultaneously with polynomial time. Planar graphs are graphs that can be drawn on the plane such that no two edges of the graph cross each other at an intermediate point. Imai et al. showed that reachability in planar graph can be solved in $O(n^{1/2+\epsilon})$ space for any $\epsilon > 0$ [9] . Later this space bound was improved to $\tilde{O}(n^{1/2})$ [4]. For graphs of higher genus, Chakraborty et al. gave an $\tilde{O}(n^{2/3}g^{1/3})$ space algorithm which additionally requires an embedding of the graph on a surface of genus g, as input [7]. They also gave an $\tilde{O}(n^{2/3})$ space algorithm for H minor-free graphs which requires tree decomposition of the graph as an input and $O(n^{1/2+\epsilon})$ space algorithm for $K_{3,3}$-free and K_5-free graphs. For layered planar graphs, Chakraborty and Tewari showed that for every $\epsilon > 0$ there is an $O(n^{\epsilon})$ space algorithm [8]. Stolee and Vinodchandran presented a polynomial time algorithm that, for any $\epsilon > 0$ solves reachability in a directed acyclic graph with $O(n^{\epsilon})$ sources and embedded on the surface of genus $O(n^{\epsilon})$ using $O(n^{\epsilon})$ space [14]. For unique-path graphs, Kannan et al. presented a $O(n^{\epsilon})$ space and polynomial time algorithm [10].

Grid graphs are a subclass of planar graphs whose vertices are present at the integer lattice points of an $m \times m$ grid and edges can only occur between a vertex and its immediate vertical or horizontal neighbor. It was known that reachability in planar graphs can be reduced to reachability in grid graphs in logarithmic space [1]. The reduction, however, causes atleast a quadratic blow-up in size with respect to the input graph. In this paper, we study the simultaneous time-space complexity of reachability in grid graphs.

Asano and Doerr presented a polynomial time algorithm that uses $O(n^{1/2+\epsilon})$ space for solving reachability in grid graphs [3]. Ashida and Nakagawa presented an algorithm with improved space complexity of $\widetilde{O}(n^{1/3})$ [5]. The latter 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 a $O(n^{1/4+\epsilon})$ space and polynomial time algorithm for grid graph reachability, thereby significantly improving the space bound of Ashida and Nakagawa.

▶ **Theorem 1** (Main Theorem). *For every $\epsilon > 0$, there exists a polynomial time algorithm that can solve reachability in an n vertex grid graph, using $O(n^{1/4+\epsilon})$ space.*

To solve the problem we divide the given grid graph into subgrids and replace paths in each grid with a single edge between the boundary vertices to get an *auxiliary graph*. This reduces the size of the graph and preserves reachability. Instead of trying to convert the auxiliary graph into planar graph while preserving reachability (which was the approach of Ashida and Nakagawa [5]), we use a divide and conquer strategy to directly solve reachability problem in the auxiliary graph. We define and use a new type of graph separator of the auxiliary graph, that we call as *pseudoseparator* and use it to divide the auxiliary graph into small components and then combine the solution in a space-efficient manner.

In Section 2 we state the definitions and notations that we use in this paper. In Section 3 we define the auxiliary graph and state various properties of it that we use later. In Section 4 we discuss the concept of a pseudoseparator. We give its formal definition and show how a pseudoseparator can be computed efficiently. In Section 5 we give the algorithm to solve reachability in an auxiliary graph and prove its correctness. Finally in Section 6 we use the algorithm of Section 5 to give an algorithm to decide reachability in grid graphs and thus prove Theorem 1.

2 Preliminaries

Let $[n]$ denote the set $\{0, 1, 2, \ldots, n\}$. 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 $\mathsf{cc}(G)$ as the set of all connected components in the underlying undirected graph of G, where the undirected version is obtained by removing orientations on all edges of G. Henceforth, whenever we talk about connected components, we will mean the connected components of the underlying undirected graph.

In a *drawing* of a graph on a plane, each vertex is mapped to a point on the plane, and each edge is mapped to a simple arc whose endpoints coincide with the mappings of the end vertices of the edge. Moreover, the interior points of an arc corresponding to an edge does not intersect with any other vertex points. A graph is said to be *planar* if it can be drawn on the plane such that no two edges of the graph intersect except possibly at the endpoints. We will not be concerned with details about the representation of planar graphs. We note that the work of [2], and subsequently [12], implies a deterministic logarithmic space algorithm that decides whether or not a given graph is planar, and if it is, outputs a planar embedding. Hence when dealing with planar graphs, we will assume without loss of generality that a planar embedding is provided as well.

A $m \times m$ *grid graph* is a directed graph whose vertices are $[m] \times [m] = \{0, 1, \ldots, m\} \times \{0, 1, \ldots, m\}$ so that if $((i_1, j_1), (i_2, j_2))$ is an edge then $|i_1 - i_2| + |j_1 - j_2| = 1$. It follows from definition that grid graphs are a subset of planar graphs.

3 Auxiliary Graph

Let G be an $m \times m$ grid graph. We divide G into $m^{2\alpha}$ subgrids such that each subgrid is a $m^{1-\alpha} \times m^{1-\alpha}$ grid. Formally, for $1 \leq i, j \leq m^{\alpha}$, the (i, j)-th *subgrid* of G, denoted as $G[i, j]$ is the subgraph of G induced by the set of vertices, $V(G[i, j]) = \{(i', j') \mid (i-1) \cdot m^{1-\alpha} \leq i' \leq i \cdot m^{1-\alpha}$ and $(j-1) \cdot m^{1-\alpha} \leq j' \leq j \cdot m^{1-\alpha}\}$.

For $0 < \alpha < 1$ and $1 \leq i, j \leq m^{\alpha}$, we define $\mathsf{Aux}_{\alpha}(G)[i, j]$ as follows. The vertex set of $\mathsf{Aux}_{\alpha}(G)[i, j]$ is $V(\mathsf{Aux}_{\alpha}(G)[i, j]) = \{(i', j') \mid i' = k \cdot m^{1-\alpha}$ or $j = l \cdot m^{1-\alpha}$, such that $k \in \{i-1, i\}$ and $l \in \{j-1, j\}\}$. For two vertices u, v in $\mathsf{Aux}_{\alpha}(G)[i, j]$, (u, v) is an edge in $\mathsf{Aux}_{\alpha}(G)[i, j]$ if there is a path from u to v in the subgrid $G[i, j]$. In a drawing of $\mathsf{Aux}_{\alpha}(G)[i, j]$, we use a straight line to represent the edge if u and v do not lie on a single side of $\mathsf{Aux}_{\alpha}(G)[i, j]$, and an arc present inside the grid to represent it otherwise.

Now for $0 < \alpha < 1$, we define the *α-auxiliary graph*, $\mathsf{Aux}_{\alpha}(G)$ as follows. The vertex set of $\mathsf{Aux}_{\alpha}(G)$, $V(\mathsf{Aux}_{\alpha}(G)) = \{(i, j) \mid i = k \cdot m^{1-\alpha}$ or $j = l \cdot m^{1-\alpha}$, such that $0 \leq k, l \leq m^{\alpha}\}$. The edges of $\mathsf{Aux}_{\alpha}(G)$ are the edges of $\mathsf{Aux}_{\alpha}(G)[i, j]$ taken over all pairs (i, j). Note that $\mathsf{Aux}_{\alpha}(G)$ might have parallel edges, since an edge on a side of a block might be present in the adjacent block as well. In such cases we preserve both the edges, however in their different blocks of $\mathsf{Aux}_{\alpha}(G)$ in the drawing of $\mathsf{Aux}_{\alpha}(G)$ on the plane. Figure 1 contains an example of a grid graph partitioned into subgrids and its corresponding auxiliary graph. Since each block $\mathsf{Aux}_{\alpha}(G)[i, j]$ contains $4m^{1-\alpha}$ vertices, the total number of vertices in $\mathsf{Aux}_{\alpha}(G)$ would be at most $4m^{1+\alpha}$.

Our algorithm for reachability first constructs $\mathsf{Aux}_{\alpha}(G)$ by solving each of the $m^{1-\alpha} \times m^{1-\alpha}$ grids recursively. It then uses a polynomial time subroutine to decide reachability in $\mathsf{Aux}_{\alpha}(G)$. Note that we do not store the graph $\mathsf{Aux}_{\alpha}(G)$ explicitly, since that would require too much space. Rather we solve a subgrid recursively whenever the subroutine queries for an edge in that subgrid of $\mathsf{Aux}_{\alpha}(G)$.

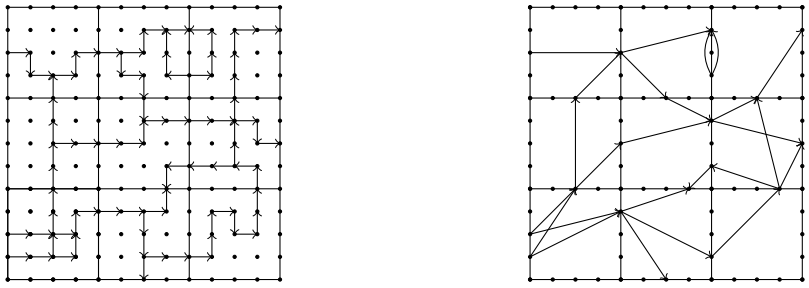

Figure 1 A grid graph G divided into subgrids and its corresponding auxiliary graph $\mathsf{Aux}_\alpha(G)$.

Our strategy is to develop a polynomial time algorithm which solves reachability in $\mathsf{Aux}_\alpha(G)$ using $\tilde{O}(\tilde{m}^{1/2+\beta/2})$ space where $\tilde{m}$ is the number of vertices in $\mathsf{Aux}_\alpha(G)$. As discussed earlier, $\tilde{m}$ can be at most $4m^{1+\alpha}$. Hence, the main algorithm would require $\tilde{O}(m^{1/2+\beta/2+\alpha/2+\alpha\beta/2})$ space. For a fixed constant $\epsilon > 0$, we can pick $\alpha > 0$ and $\beta > 0$ such that the space complexity becomes $O(m^{1/2+\epsilon})$.

3.1 Properties of the Auxiliary Graph

In the following definition, we give an ordered labeling of the vertices on some block of the auxiliary graph. The labeling is defined with respect to some vertex present in the block.

▶ **Definition 2.** *Let G be a $m \times m$ grid graph, $l = \mathsf{Aux}_\alpha(G)[i,j]$ be a block of $\mathsf{Aux}_\alpha(G)$ and $v = (x,y)$ be a vertex in $\mathsf{Aux}_\alpha(G)[i,j]$. Let $t = m^{1-\alpha}$. We define the counter-clockwise adjacent vertex of v with respect to the block l, $c_l(v)$ as follows:*

$$c_l(v) = \begin{cases} (x+1, y) & \text{if } x < (i+1)t \text{ and } y = jt \\ (x, y+1) & \text{if } x = (i+1)t \text{ and } y < (j+1)t \\ (x-1, y) & \text{if } x > it \text{ and } y = (j+1)t \\ (x, y-1) & \text{if } x = it \text{ and } y > jt \end{cases}$$

Similarly we also define the r-th counter-clockwise adjacent neighbour of v with respect to the block l inductively as follows. For $r = 0$, $c_l^r(v) = v$ and otherwise we have $c_l^{r+1}(v) = c_l(c_l^r(v))$.

Note that for a block l and vertices v and w in it, we write v as $c_l^p(w)$ where p is smallest non-negative integer for which $c_l^p(w) = v$. Next we formalize what it means to say that two edges of the auxiliary graph cross each other.

▶ **Definition 3.** *Let G be a grid graph and l be a block of $\mathsf{Aux}_\alpha(G)$. For two distinct edges e and f in the block, such that $e = (v, c_l^p(v))$ and $f = (c_l^q(v), c_l^r(v))$. We say that edges e and f cross each other if* $\min(q,r) < p < \max(q,r)$.

Note the definition of cross given above is symmetric. That is, if edges e and f cross each other then f and e must cross each other as well. For an edge $f = (c_l^q(v), c_l^r(v))$, we define $\overleftarrow{f} = (c_l^r(v), c_l^q(v))$ and call it the *reverse* of f. We also note that if e and f cross each other, then e and $\overleftarrow{f}$ also cross each other.

In Lemma 4 we state an equivalent condition of crossing of two edges, and in Lemmas 6 and 7 we state certain properties of the auxiliary graph that we use later.

▶ **Lemma 4.** *Let G be a grid graph and l be a block of $\mathsf{Aux}_\alpha(G)$. Let w be an arbitrary vertex in the block l and $e = (c_l^p(w), c_l^q(w))$ and $f = (c_l^r(w), c_l^s(w))$ be two distinct edges in l. Then e and f cross each other if and only if either of the following two conditions hold:*

- $\min(p,q) < \min(r,s) < \max(p,q) < \max(r,s)$
- $\min(r,s) < \min(p,q) < \max(r,s) < \max(p,q)$

Proof. We prove that if $\min(p,q) < \min(r,s) < \max(p,q) < \max(r,s)$ then e and f cross each other. We let $p < r < q < s$. Other cases can be proved by reversing appropriate edges. We thus have integers $r_1 = r - p$, $q_1 = q - p$ and $s_1 = s - p$. Clearly, $r_1 < q_1 < s_1$. Let $v = c_l^p(w)$. Thus we have $e = (v, c_l^q(w)) = (v, c_l^{q_1}(c_l^p(w))) = (v, c_l^{q_1}(v))$ and $f = (c_l^r(w), c_l^s(w)) = (c_l^{r_1}(c_l^p(w)), c_l^{s_1}(c_l^p(w))) = (c_l^{r_1}(v), c_l^{s_1}(v))$ The proof for the second condition is similar.

Now, we prove that if $e = (c_l^p(w), c_l^q(w))$ and $f = (c_l^r(w), c_l^s(w))$ cross each other then either of the given two condition holds. We assume that p is the smallest integer among p, q, r and s. Other cases can be proved similarly. Now, let $v = c_l^p(w)$. We thus have integers $q_1 = q - p, r_1 = r - p$ and $s_1 = s - p$ such that $e = (v, c_l^{q_1}(v))$ and $f = (c_l^{r_1}(v), c_l^{s_1}(v))$. Since e and f cross each other, we have $\min(r_1, s_1) < q_1 < \max(r_1, s_1)$. Thus $\min(r_1 + p, s_1 + p) < q_1 + p < \max(r_1 + p, s_1 + p)$. It follows that $\min(r,s) < q < \max(r,s)$. Since we assumed p to be smallest integer among p, q, r and s; we have $\min(p,q) < \min(r,s) < \max(p,q) < \max(r,s)$, thus proving the lemma. ◀

We see that we can draw an auxiliary graph on a plane such that the arcs corresponding to two of its edges intersect if and only if the corresponding edges cross each other. Henceforth, we will work with such a drawing.

▶ **Definition 5.** *Let G be a grid graph and l be a block of $\mathsf{Aux}_\alpha(G)$. For a vertex v and edges f, g such that $f = (c_l^q(v), c_l^r(v))$ and $g = (c_l^s(v), c_l^t(v))$, we say that f is closer to v than g if $\min(q,r) < \min(s,t)$.*

We say f is closest *to v if there exists no other edge f' which is closer to v than f.*

▶ **Lemma 6.** *Let G be a grid graph and $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ be two edges in $\mathsf{Aux}_\alpha(G)$. If e_1 and e_2 cross each other, then $\mathsf{Aux}_\alpha(G)$ also contains the edges (u_1, v_2) and (u_2, v_1).*

Proof. Let $e_1 = (v, c_l^p(v))$ and $e_2 = (c_l^q(v), c_l^r(v))$ be two edges that cross each other in $\mathsf{Aux}_\alpha(G)$. Let l be the block of $\mathsf{Aux}_\alpha(G)$ to which e_1 and e_2 belong. Consider the subgrid of G which is solved to construct the block l. Since the edge e_1 exists in block l, there exists a path P from v to $c_l^p(v)$ in the underlying subgrid. This path P divides the subgrid into two parts such that the vertices $c_l^q(v)$ and $c_l^r(v)$ belong to different parts of the subgrid. Thus, a path between $c_l^q(v)$ and $c_l^r(v)$ necessarily take a vertex of path P. Hence, there is a path from v to $c_l^r(v)$ and a path from $c_l^p(v)$ to $c_l^r(v)$. Thus the lemma follows. ◀

▶ **Lemma 7.** *Let G be a grid graph and $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ be two edges in $\mathsf{Aux}_\alpha(G)$. If e_1 and e_2 cross a certain edge $f = (x, y)$, and e_1 is closer to x than e_2, then the edge (u_1, v_2) is also present in $\mathsf{Aux}_\alpha(G)$.*

Proof. Let $e = (v, c_l^p(v))$, $f = (c_l^q(v), c_l^r(v))$ and $g = (c_l^s(v), c_l^t(v))$. If $c_l^q(v) = c_l^s(v)$ then the lemma trivially follows. Otherwise, we have two cases to consider:

Case 1 (f crosses g): In this case, we will have $(c_l^q(v), c_l^t(v))$ present in $\mathsf{Aux}_\alpha(G)$ by Lemma 6.

Case 2 (f does not cross g): In this case, we have $\min(q,r) < \min(s,t) < p < \max(s,t) < \max(q,r)$. Since f crosses e, we have the edge $(c_l^q(v), c_l^p(v))$ in $\mathsf{Aux}_\alpha(G)$ by Lemma 6. This edge will cross g. Hence $(c_l^q(v), c_l^t(v))$ is present in $\mathsf{Aux}_\alpha(G)$. ◀

4 Pseudoseparator in a Grid Graph

Imai et al. used a separator construction to solve the reachability problem in planar graphs [9]. A separator is a set of vertices whose removal disconnects the graph into *small 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.

Grid graphs are subclasses of planar and are known to have good separators. However, for a grid graph G, the graph $\mathsf{Aux}_\alpha(G)$ might not have a small separator. Here we show that $\mathsf{Aux}_\alpha(G)$ has a different kind of separator, which we call as a PseudoSeparator (see Definition 8). PseudoSeparator allows us to decide reachability in $\mathsf{Aux}_\alpha(G)$, by using a divide and conquer strategy and obtain the claimed time and space bounds.

For a graph $H = (V_1, E_1)$ given along with its drawing, and a subgraph $C = (V_2, E_2)$ of H, define the graph $H \diamond C = (V_3, E_3)$ as $V_3 = V_1 \setminus V_2$ and $E_3 = E_1 \setminus \{e \in E_1 \mid \exists f \in E_2, e \text{ crosses } f\}$. We note that the graph H we will be working with throughout the article will be a subgraph of an auxiliary graph. Hence it will always come with a drawing.

▶ **Definition 8.** *Let G be a grid graph and H be a vertex induced subgraph of $\mathsf{Aux}_\alpha(G)$ with h vertices. Let $f : \mathbb{N} \to \mathbb{N}$ be a function. A subgraph C of H is said to be an $f(h)$-*PseudoSeparator *of $\mathsf{Aux}_\alpha(G)$ if the size of every connected component in $\mathsf{cc}(H \diamond C)$ 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 $\mathsf{Aux}_\alpha(G)$, an $f(h)$-PseudoSeparator is a subgraph C of H that has the property that, if we remove the vertices as well as all the edges that cross one of the edges of the PseudoSeparator, the graph gets disconnected into small pieces. Moreover for every edge e in H, if there exists distinct sets U_1 and U_2 in $\mathsf{cc}(H \diamond C)$ such that one of the endpoints of e is in U_1 and the other is in U_2, then there exists an edge f in C such that e crosses f. Hence any path which connects two vertices in different components, must either contain a vertex of C or must contain an edge that crosses an edge of C. We divide the graph using this PseudoSeparator and give an algorithm which recursively solves each subgraph and then combines their solution efficiently.

4.1 Constructing a Pseudoseparator

We briefly comment on how to construct a PseudoSeparator of a vertex induced subgraph H of $\mathsf{Aux}_\alpha(G)$. First, we pick a maximal subset of edges from H so that no two edges cross (see Defintion 9). Then we triangulate the resulting graph. This can be done in logspace. Next, we use Imai et al.'s algorithm to find a separator of the triangulated graph. Call the triangulated graph as $\widehat{H}$ and the separator vertices as S. The vertex set of PseudoSeparator of H will contain 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 will contain all the 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.

▶ **Definition 9.** *Let G be a grid graph and H be a vertex induced subgraph of $\mathsf{Aux}_\alpha(G)$. We define $\mathsf{planar}(H)$ as a subgraph of H. The vertex set of $\mathsf{planar}(H)$ is same as that of H. For an edge $e \in H$, let l be the block to which e belongs and let w be the lowest indexed vertex in that block. Then $e = (c_l^i(w), c_l^j(w))$ is in $\mathsf{planar}(H)$ if there exists no other edge $f = (c_l^x(w), c_l^y(w))$ in H such that $\min(x, y) < \min(i, j) < \max(x, y) < \max(i, j)$.*

In Lemma 10 we show that the graph $\mathsf{planar}(H)$ is indeed planar and prove a simple yet crucial property of this graph, that would help us to construct the PseudoSeparator.

▶ **Lemma 10.** *Let G be a grid-graph and H be a vertex induced subgraph of $\mathsf{Aux}_\alpha(G)$. No two edges of $\mathsf{planar}(H)$ cross each other. Moreover, for any edge e in H that is not in $\mathsf{planar}(H)$, there exists another edge in $\mathsf{planar}(H)$ that crosses e.*

Proof. Let l be a block of $\mathsf{Aux}_\alpha(G)$ and w be the smallest index vertex of l. Let $e = (c_l^p(w), c_l^q(w))$ and $f = (c_l^r(w), c_l^s(w))$ be two edges of H that cross. We have, by lemma 4, that either $\min(p,q) < \min(r,s) < \max(p,q) < \max(r,s)$ or $\min(r,s) < \min(p,q) < \max(r,s) < \max(p,q)$. Hence, by our construction of $\mathsf{planar}(H)$, atmost one of e and f belongs to it. Thus no two edges of $\mathsf{planar}(H)$ can cross.

For the second part, we will prove by contradiction. Let us assume that there exists edges in H which is not in $\mathsf{planar}(H)$ and also not crossed by an edge in $\mathsf{planar}(H)$. We pick edge $e = (c_l^p(w), c_l^q(w))$ from them such that $\min(p,q)$ of that edge is minimum. Since this edge is not present in $\mathsf{planar}(H)$, we have by definition, an edge $f = (c_l^r(w), c_l^s(w))$ such that $\min(r,s) < \min(p,q) < \max(r,s) < \max(p,q)$. We pick the edge f for which $\min(r,s)$ is minimum. Now, since this edge f is not present in $\mathsf{planar}(H)$, we have another edge $g = (c_l^i(w), c_l^j(w))$ in $\mathsf{planar}(H)$ such that $\min(i,j) < \min(r,s) < \max(i,j) < \max(r,s)$. We pick g such that $\min(i,j)$ is minimum and break ties by picking one whose $\max(i,j)$ is maximum. Now, we have the following cases:

Case 1 ($i < r < j < s$): In this case, the edge $c_l^i(w), c_l^s(w)$ will be present in H. Since $i < p < s < q$, and $i < \min(r,s)$, this will contradict the way in which edge f was chosen.

Case 2 ($i < s < j < r$): In this case, the edge $(c_l^r(w), c_l^j(w))$ will be present in H. This edge will cross e and hence not be present in $\mathsf{planar}(H)$. Thus, we have an edge $g' = (c_l^{i'}(w), c_l^{j'}(w))$ in $\mathsf{planar}(H)$ such that $\min(i',j') < j < \max(i',j') < r$. We will thus have two subcases.

- **Case 2a ($i < \min(i',j')$):** Here, we will have $i < \min(i',j') < j < \max(i',j')$. Hence this edge will cross g giving a contradiction to the first part of this lemma.
- **Case 2b ($\min(i',j') \leq i$):** Here, this edge should have been chosen instead of g contradicting our choice of g.

The analysis of two remaining cases where $j < s < i < r$ and $j < r < i < s$ are similar to Cases 1 and 2 respectively. ◀

We next describe how to compute the triangulated planar graph $\widehat{H}$. Given H as an input, we first find $\mathsf{planar}(H)$ using Lemma 10. We then triangulate $\mathsf{planar}(H)$ by first adding edges in the boundary of each block as follows: let l be a block and v be a vertex in l. Let p be the smallest positive integer such that the vertex $c_l^p(v)$ is present in H. If the edge $(v, c_l^p(v))$ is not present in $\mathsf{planar}(H)$, we add this in $\widehat{H}$. This procedure does not result in a non-planar graph since no edge of $\mathsf{planar}(H)$ goes from one block to another. Every edge of l is now inside the boundary cycle. Finally, we triangulate the rest of the graph and add the triangulation edges to $\widehat{H}$. Note that this process can be done in logspace.

We will be using the following lemma which was proven by Imai et al.

▶ **Lemma 11** ([9])**.** *For every $\beta > 0$, there exists a polynomial time and $\tilde{O}(h^{1/2+\beta/2})$ space algorithm that takes a h-vertex planar graph P as input and outputs a set of vertices S, such that $|S|$ is $O(h^{1/2+\beta/2})$ and removal of S disconnects the graph into components of size $O(h^{1-\beta})$.*

For a subgraph H of $\mathsf{Aux}_\alpha(G)$, we construct the graph $\mathsf{psep}(H)$ in the following way.

(i) Construct $\widehat{H}$ from H.

(ii) Find a set S of vertices in $\widehat{H}$ which divides it into components of size $O(n^{1-\beta})$ by applying Lemma 11.

(iii) Add each vertex of S to the set $V(\mathsf{psep}(H))$ and each edge of $\widehat{H}(S)$ which is also in H to $E(\mathsf{psep}(H))$.

(iv) Let $e = (v, w)$ be a triangulation edge present in block l of $\widehat{H}$ whose both endpoints are in S. Let p and q be integers such that $w = c_l^p(v)$ and $v = c_l^q(w)$. We add the following set of at most four vertices and four edges to $\mathsf{psep}(H)$.

1. Let $p_1 < p$ be the largest integer such that an edge e_1 with endpoints v and $c_l^{p_1}(v)$ exists.
2. Let $p_2 > p$ be the smallest integer such that an edge e_2 with endpoints v and $c_l^{p_2}(v)$ exists.
3. Let $q_1 < q$ be the largest integer such that an edge e_3 with endpoints $c_l^{q_1}(w)$ and w exists.
4. Let $q_2 > q$ be the smallest integer such that an edge e_4 with endpoints $c_l^{q_2}(w)$ and w exists.

Note that the above edges could be directed either way. We add the vertices $c_l^{p_1}(v)$, $c_l^{p_2}(v)$, $c_l^{q_1}(w)$ and $c_l^{q_2}(w)$ to $V(\mathsf{psep}(H))$. For $i = 1, 2, 3, 4$, we add the edges e_i to $E(\mathsf{psep}(H))$. We call these four edges as *shadows* of e.

In Lemma 12 we show that the set $\mathsf{psep}(H)$ is a PseudoSeparator of H.

▶ **Lemma 12.** *Let G be a grid graph and H be a vertex induced subgraph of $\mathsf{Aux}_\alpha(G)$. The graph $\mathsf{psep}(H)$ is a $h^{1-\beta}$-*PseudoSeparator *of H.*

To prove Lemma 12, we first show a property of triangulated graphs that we use in our construction of PseudoSeparator. It is known that a simple cycle in a planar embedding of a planar graph divides the plane into two parts. We call these two parts the two *sides* of the cycle.

▶ **Lemma 13.** *Let G be a triangulated planar graph and S be a subset of its vertices. For every pair of vertex u, v which belong to different components of $G \setminus S$, there exists a cycle in $G[S]$, such that u and v belong to different sides of this cycle.*

Proof. To prove the lemma, we first need some terminology. We call a set of faces an *edge-connected region* if it can be constructed in the following way:

- A set of a single face is an edge-connected region.
- If a set F is an edge-connected region and f is a face that shares an edge with one of the faces of F, then $F \cup \{f\}$ is an edge-connected region.

We can orient the edges of an undirected planar simple cycle to make it a directed cycle. This can help us identify the two *sides* of the cycle as *interior* (left-side) and *exterior* (right-side).

Let C be a component of $G \setminus S$ and S' be the set of vertices of S which are adjacent to at least one of the vertices of C in G. Let F be the set of triangle faces of G to which at least one vertex of C belongs. We first observe that for any face f of F, the vertices of f will either belong to C or S'. We see that F is a region of edge-connected faces. Miller proved that we could write the boundary of the region of edge-connected faces as a set of vertex-disjoint simple cycles with disjoint exteriors [11]. These cycles will contain only the vertices of S'. Hence the lemma follows. ◀

Proof of Lemma 12. Let $C = \mathsf{psep}(H)$. Let S be the set of vertices obtained from $\widehat{H}$ by using Lemma 11. We claim that if any two vertices u and v belong to different connected components of $\widehat{H} \setminus S$, then it belongs to diffent components of $\mathsf{cc}(H \diamond C)$. We prove this by contradiction. Let us assume that it is not true. Then there is an edge from e in H and two

(a) The $s-t$ path takes a vertex of the separator. **(b)** The $s-t$ path crosses an edge of the separator.

Figure 2

distinct sets U_1 and U_2 of $\widehat{H} \setminus S$ such that one of the end point of e is in U_1 and the other is in U_2. This edge e does not cross any of the edge of $\mathsf{psep}(H)$. Without loss of generality, let $e = (v, c_l^p(v))$, where $v \in U_1$ and $c_l^p(v)$ is not in U_1. (we pick the edge e such that p is minimum) Due to Lemma 13, it follows that there exists a triangulation edge f such that $f = ((c_l^q(v)), c_l^r(v))$ and that e crosses f. We orient the triangulation edge so that $q < p < r$. Now, since e is not present in $\mathsf{planar}(H)$, by Lemma 10 there exists at least one edge that crosses it and is present in $\mathsf{planar}(H)$. Let $g = (c_l^s(v), c_l^t(v))$ be one such edge such that $t - s$ is maximum. We thus have the following cases:

Case 1 ($s < q < p < r < t$): In this case, since g crosses e, by Lemma 6, we have that the edge $e' = (c_l^s(v), c_l^p(v))$ is also present in H. e' also crosses f. Since $p - s < p$, existence of e' contradicts our choice of e.

Case 2 ($q < t < p < s < r$): In this case, since g crosses e, by Lemma 6, we have that the edge $e' = (v, c_l^t(v))$ is also present in H. e' also crosses f. Since $t < p$, existence of e' contradicts our choice of e.

Case 3 ($q < s < p < t < r$): In this case, since g crosses e, by Lemma 6, we have that the edge $e' = (v, c_l^t(v))$ is present in H. e' also crosses f and hence e' was not present in $\mathsf{planar}(H)$. Thus there will exists an edge in $\mathsf{planar}(H)$ which crosses e' by Lemma 10. Let $g' = (c_l^{s'}(v), c_l^{t'}(v))$ be the edge in $\mathsf{planar}(H)$ that crosses e' such that $t' - s'$ is maximum. We see that $t' < q$ and $s' > r$, for otherwise, existence of g' will contradict the way g is chosen. Now, since the edges g and g' both cross e and g' is closer to v than g, by Lemma 7, the edge $e'' = (c_l^{s'}(v), c_l^t(v))$ will also be present in H. e'' will cross f. Now, any edge present in $\mathsf{planar}(H)$ that crosses e'' will contradict our choice of g or g'.

Case 4 ($t < q < p < r < s$): In this case, the edge $e' = (c_l^s(v), c_l^p(v))$ will also be present in H. e' will cross f and hence will not be present in $\mathsf{planar}(H)$. Any edge present in $\mathsf{planar}(H)$ that also crosses e' will contradict the way g is chosen.

In other cases, if g is picked such that one of its vertices is common with f, then e will cross a *shadow* edge of f giving a contradiction. If g is picked such that g cross f, then it contradicts the fact that both of them are present in $\widehat{H}$. ◀

Summarizing Lemmas 11 and 12 we have Theorem 14.

▶ **Theorem 14.** *Let G be a grid graph and H be a vertex induced subgraph of $\mathsf{Aux}_\alpha(G)$ with h vertices. For any constant $\beta > 0$, there exists an $\tilde{O}(h^{1/2+\beta/2})$ space and polynomial time algorithm that takes H as input and outputs an $h^{1-\beta}$-*PseudoSeparator *of size $O(h^{1/2+\beta/2})$.*

5 Algorithm to Solve Reachability in Auxiliary Graph

In this section, we discuss the grid graph reachability algorithm. Let G be a grid graph having $\widetilde{n}$ vertices. By induction we assume that we have access to a vertex induced subgraph H of $\mathsf{Aux}_\alpha(G)$, containing h vertices. Below we describe a recursive procedure $\mathsf{AuxReach}(H, x, y)$ that outputs true if there is a path from x to y in H and outputs false otherwise.

5.1 Description of the Algorithm AuxReach

First we construct a $h^{1-\beta}$-PseudoSeparator C of H, using Theorem 14. We also ensure that x and y are part of C (if not then we add them). Let $I_1, I_2, \ldots, I_l$ be the connected components of $H \diamond C$.

We maintain an array called visited of size $|C|$ to mark vertices or edges of the PseudoSeparator C. Each cell of visited corresponds to a distinct vertex or edge of C. For a vertex v in C, we set visited$[v] := 1$ if there is a path from x to v in H, else it is set to 0. For an edge $e = (u, v)$ in C, we set visited$[e] := u'$ if (i) there is an edge $f = (u', v')$ that crosses e, (ii) there is a path from x to u' in H and (iii) f is the closest such edge to u. Else visited$[e]$ is set to NULL. Initially, for all vertex $v \in C$, visited$[v] := 0$ and for all edges $e \in C$, visited$[e] :=$ NULL. We say that a vertex v is *marked* if either visited$[v] := 1$ or visited$[e] := v$ for some edge e.

First set visited$[x] := 1$. We then perform an outer loop with h iteration and in each iteration update certain entries of the array visited as follows. For every vertex $v \in C$, the algorithm sets visited$[v] := 1$ if there is a path from a marked vertex to v such that the internal vertices of that path all belong to only one component I_i. Similarly, for each edge $e = (u, v)$ of C, the algorithm sets visited$[e] := u'$ if (i) there exists an edge $f = (u', v')$ which crosses e, (ii) there is a path from a marked vertex to u' such that the internal vertices of that path all belong to only one component I_i and, (iii) f is the closest such edge to u. Finally we output true if visited$[y] = 1$ else output false. We use the procedure AuxReach recursively to check if there is a path between two vertices in a single connected component of $H \diamond C$. A formal description of AuxReach is given in Algorithm 1.

5.2 Proof of Correctness of AuxReach

Let P be a path from x to y 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 in the following two ways only:

i. The path exits I_{σ_j} through a vertex w of PseudoSeparator as shown in Figure 2a. In this case, Algorithm 1 would mark the vertex w.
ii. The path exits I_{σ_j} through an edge (u, v) whose other endpoint is in $I_{\sigma_{j+1}}$. By Lemma 7, this edge will cross an edge $e = (x', y')$ of the PseudoSeparator. In this case, Algorithm 1 would mark the vertex u', such that there is an edge (u', v') that crosses e as well and (u', v') is closer than (u, v) to x' and there is a path in I_{σ_j} from a marked vertex to u'. By Lemma 7, the edge (u', v) would be present in H as well.

Thus after the j-th iteration, AuxReach 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 would be marked after L iterations if and only if there is a path from s to t in H. We give a formal proof of correctness in Lemma 15. For a path $P = (u_1, u_2, \ldots u_t)$, we define tail$(P) := u_1$ and head$(P) := u_t$.

▶ **Lemma 15.** *Let G be a grid graph and H be a vertex induced subgraph of $\mathsf{Aux}_\alpha(G)$. Then for any two vertices x, y in H, there is a path from x to y in H if and only if AuxReach(H, x, y) returns true.*

Proof. Firstly observe that a vertex is marked only if there is a path from some other marked vertex to that vertex in H. Hence if there is no path from x to y then y is never marked by AuxReach and hence AuxReach returns false.

Now let P be a path from x to y in H. We divide the path into subpaths $P_1, P_2, \ldots, P_l$, such that for each i, all vertices of P_i belong to $U \cup V(C)$ for some connected component U in cc$(H \diamond C)$ and either (i) head$(P_i) =$ tail(P_{i+1}), or (ii) $e_i =$ (head(P_i), tail(P_{i+1})) is an edge

Algorithm 1 AuxReach(H, s, t).

Input: A vertex induced subgraph H of $\mathsf{Aux}_\alpha(G)$ and two vertices x and y in H (let G be an $m \times m$ grid graph and $h = |V(H)|$)
Output: true if there is a path from x to y in H and false otherwise

```
1  if h ≤ m^{1/8} then Use DFS to solve the problem;   /* m is a global variable
     where G is an m × m grid graph */
2  else
3      Compute a h^{1−β}-PseudoSeparator C of H using Theorem 14;
4      C ← C ∪ {x, y};
5      foreach edge e in C do visited[e] ← NULL;
6      foreach vertex v in C do visited[v] ← 0;
7      visited[x] ← 1;
8      for i = 1 to |H| do
9          foreach edge e = (u, v) ∈ C do
10             if ((∃ marked vertex w) · (∃U ∈ cc(H ⋄ C)) · (∃f = (u′, v′) such that f
                 crosses e and f is closest to u) · (AuxReach(H[U ∪ {w, u′}], w, u′) = true))
                 then
11                 visited[e] ← u′
12             end
13         end
14         foreach vertex v ∈ C do
15             if ((∃ marked vertex w) · (∃U ∈
                 cc(H ⋄ C)) · (AuxReach(H[U ∪ {w, v}], w, v) = true)) then
16                 visited[v] ← 1
17             end
18         end
19     end
20     if visited[y] = 1 then return true;
21     else return false;
22 end
```

that crosses some edge $f_i \in C$. By Definition 8, we have that if condition (i) is true then $\mathsf{head}(P_i)$ is a vertex in C, and if condition (ii) is true then $\mathsf{head}(P_i)$ and $\mathsf{tail}(P_{i+1})$ belong to two different components of $\mathsf{cc}(H \diamond C)$ and e_i is the edge between them.

We claim that after i-th iteration of loop in Line 8 of Algorithm 1, either of the following two statements hold:

- **(I)** $\mathsf{head}(P_i)$ is a vertex in C and $\mathsf{visited}[\mathsf{head}(P_i)] = 1$.
- **(II)** There exists an edge $f_i = (u_i, v_i)$ of C such that the edge $e_i = (\mathsf{head}(P_i), \mathsf{tail}(P_{i+1}))$ crosses f_i and there is an edge $g_i = (u_i', v_i')$ which crosses f_i as well, such that g_i is closer to u_i than e_i and $\mathsf{visited}[f_i] = u_i'$.

We prove the claim by induction. The base case holds since x is marked at the beginning. We assume that the claim is true after the $(i-1)$-th iteration. We have that P_i belongs to $U \cup V(C)$ for some connected component U in $\mathsf{cc}(H \diamond C)$.

Case 1 **($\mathsf{head}(P_{i-1}) = \mathsf{tail}(P_i) = w(\mathsf{say})$):** By induction hypothesis w was marked after the $(i-1)$-th iteration. If $\mathsf{head}(P_i)$ is a vertex in C then it will be marked after the i-th iteration in Line 15. On the other hand if $e_i = (\mathsf{head}(P_i), \mathsf{tail}(P_{i+1}))$ is an edge that

crosses some edge $f_i = (u_i, v_i) \in C$ then in the i-th iteration in Line 10, the algorithm marks a vertex u'_i such that, $g_i = (u'_i, v'_i)$ is the closest edge to u_i that crosses f_i and there is a path from w to u'_i.

Case 2 ($e_{i-1} = (\mathsf{head}(P_{i-1}), \mathsf{tail}(P_i))$ is an edge that crosses some edge $f_{i-1} = (u_{i-1}, v_{i-1}) \in C$): By induction hypothesis, there is an edge $g_{i-1} = (u'_{i-1}, v'_{i-1})$ which crosses f_{i-1} as well, such that g_{i-1} is closer to u_{i-1} than e_{i-1} and $\mathsf{visited}[f_{i-1}] = u'_{i-1}$. By Lemma 7 there is an edge in H between u'_{i-1} and $\mathsf{tail}(P_i)$ as well. Now if $\mathsf{head}(P_i)$ is a vertex in C then it will be marked after the i-th iteration in Line 15 by querying the graph $H[U \cup \{u'_{i-1}, \mathsf{head}(P_i)\}]$. On the other hand if $e_i = (\mathsf{head}(P_i), \mathsf{tail}(P_{i+1}))$ is an edge that crosses some edge $f_i = (u_i, v_i) \in C$ then in the i-th iteration in Line 10, AuxReach queries the graph $H[U \cup \{u'_{i-1}, u'_i\}]$ and marks a vertex u'_i such that, $g_i = (u'_i, v'_i)$ is the closest edge to u_i that crosses f_i and there is a path from u'_{i-1} to u'_i. ◀

Our subroutine would solve reachability in a subgraph H (having size h) of $\mathsf{Aux}_\alpha(G)$. We do not explicitly store a component of $\mathsf{cc}(H \diamond C)$, 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 $H \diamond C$ to determine if a vertex is present in that component. We require $\widetilde{O}(h^{1/2+\beta/2})$ space to compute a $h^{1-\beta}$-PseudoSeparator by Theorem 14. We can potentially mark all the vertices of the 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 $\widetilde{O}(h^{1/2+\beta/2})$ space. The algorithm recurses on a graph with $h^{1-\beta}$ vertices. Hence the depth of the recursion is at most $3/(\log(1-\beta)^{-1})$, which is a constant.

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 would remain a polynomial in the number of vertices since the recursion depth is constant.

▶ **Lemma 16.** *Let G be an $m \times m$ grid graph and H be a vertex induced subgraph of $\mathsf{Aux}_\alpha(G)$ with h vertices. For every $\beta > 0$, AuxReach runs in $\widetilde{O}(h^{1/2+\beta/2})$ space and polynomial time.*

Proof. Since the size of a component U in $\mathsf{cc}(H \diamond C)$ might be too large, we will not explicitly store it. Instead we identify a component by the lowest index vertex present in it and use Reingold's algorithm on $H \diamond C$ to determine if a vertex is present in U. Let $S(m,h)$ and $T(m,h)$ denote the space and time complexity functions respectively of AuxReach, where G is an $m \times m$ grid graph and h is the number of vertices in the graph H. As noted earlier the depth of the recursion is at most $d := 3/(\log(1-\beta)^{-1})$.

Consider $S(m,h)$ for any $h > m^{1/8}$. By Theorem 14, we require $\widetilde{O}(h^{1/2+\beta/2})$ space to execute Line 3. We can potentially mark all the vertex of C and for each edge e of C we store at most one additional vertex in $\mathsf{visited}[e]$. Since the size of C is at most $O(h^{1/2+\beta/2})$, we require $\widetilde{O}(h^{1/2+\beta/2})$ space to store C. By induction, a call to AuxReach in line 10 and 15 requires $S(m, h^{1-\beta})$ space which can be subsequently reused. Hence the space complexity satisfies the following recurrence. Then,

$$S(m,h) = \begin{cases} S(m, h^{1-\beta}) + \widetilde{O}(h^{1/2+\beta/2}) & h > m^{1/8} \\ \widetilde{O}(h) & \text{otherwise.} \end{cases}$$

Solving we get $S(m,h) = \widetilde{O}(h^{1/2+\beta/2} + m^{1/4})$.

Next we measure the time complexity of AuxReach. Consider the case when $h > m^{1/8}$. The total number of steps in AuxReach is some polynomial in h, say p. Moreover AuxReach makes q calls to AuxReach, where q is some other polynomial in h. Hence $q(h) \leq p(h)$. Then,

$$T(m,h) = \begin{cases} q \cdot T(h^{1-\beta}) + p & h > m^{1/8} \\ O(h) & \text{otherwise.} \end{cases}$$

Solving the above recurrence we get $T(m,h) = O(p \cdot q^d + m^{1/4}) = O(p^{2d} + m^{1/4})$. ◀

6 Solving Grid Graph

Let G be an $m \times m$ grid graph. As mentioned in the introduction, our objective is to run Algorithm 1 on the graph $\mathsf{Aux}_\alpha(G)$. By definition of $\mathsf{Aux}_\alpha(G)$, for every pair of vertices x, y in $\mathsf{Aux}_\alpha(G)$, there is a path from x to y in $\mathsf{Aux}_\alpha(G)$ if and only if there is a path from x to y in G. Hence it is sufficient to work with the graph $\mathsf{Aux}_\alpha(G)$. However, we do not have explicit access to the edges of $\mathsf{Aux}_\alpha(G)$. Note that we can obtain the edges of $\mathsf{Aux}_\alpha(G)$ by solving the corresponding subgrid of G to which that edge belongs. If the subgrid is small enough, then we use a standard linear space traversal algorithm. Otherwise, we use our algorithm recursively on the subgrid. Algorithm 2 outlines this method.

Algorithm 2 $\mathsf{GridReach}(\widehat{G}, \widehat{s}, \widehat{t}, m)$.

Input: A grid graph $\widehat{G}$ and two vertices $\widehat{s}$, $\widehat{t}$ of $\widehat{G}$ and a positive integer m
Output: true if there is a path from s to t in G and false otherwise
if $\widehat{G}$ *is smaller than* $m^{1/8} \times m^{1/8}$ *grid* **then**
  Use Depth-First Search to solve the problem;
end
else
  Use $\mathsf{ImplicitAuxReach}(\mathsf{Aux}_\alpha(G), \widehat{s}, \widehat{t})$ to solve the problem;
  /* ImplicitAuxReach executes the same way as AuxReach except for the case when it queries an edge (u, v) in a block B of $\mathsf{Aux}_\alpha(G)$. In this case, the query is answered by calling $\mathsf{GridReach}(B, u, v, m)$ where B is the subgrid in which edge (u, v) might belong. */
end

Consider an $\widehat{m} \times \widehat{m}$ grid graph $\widehat{G}$. Let $S(\widehat{m})$ be the space complexity and $T(\widehat{m})$ be the time complexity of executing GridReach on $\widehat{G}$. Note that the size of $\mathsf{Aux}_\alpha(\widehat{G})$ is at most $\widehat{m}^{1+\alpha}$. For $\widehat{m} > m^{1/8}$, the space required to solve the grid-graph would be $S(\widehat{m}) = S(\widehat{m}^{1-\alpha}) + \widetilde{O}((\widehat{m}^{1+\alpha})^{1/2+\beta/2})$. This is because, a query whether $(u, v) \in \widehat{G}$ would invoke a recursion which would require $S(\widehat{m}^{1-\alpha})$ space and the main computation of ImplicitAuxReach could be done using $\widetilde{O}((\widehat{m}^{1+\alpha})^{1/2+\beta/2})$ space. Hence we get the following recurrence for space complexity.

$$S(\widehat{m}) = \begin{cases} S(\widehat{m}^{1-\alpha}) + \widetilde{O}((\widehat{m}^{1+\alpha})^{1/2+\beta/2}) & \widehat{m} > m^{1/8} \\ \widetilde{O}(\widehat{m}^{1/4}) & \text{otherwise} \end{cases}$$

Similar to the analysis of AuxReach, for appropriate polynomials p and q, the time complexity would satisfy the following recurrence:

$$T(\widehat{m}) = \begin{cases} q(\widehat{m}) \cdot T(\widehat{m}^{1-\alpha}) + p(\widehat{m}) & \widehat{m} > m^{1/8} \\ O(\widehat{m}) & \text{otherwise.} \end{cases}$$

Solving we get $S(m) = \widetilde{O}(m^{1/2+\beta/2+\alpha/2+\alpha\beta/2})$ and $T(m) = \mathsf{poly}(m)$. For any constant $\epsilon > 0$, we can chose α and β such that $S(m) = O(m^{1/2+\epsilon})$.

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. Eric Allender and Meena Mahajan. The complexity of planarity testing. *Information and Computation*, 189(1):117–134, 2004. doi:10.1016/j.ic.2003.09.002.
3. 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.
4. 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.
5. 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.
6. 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.
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. Gary L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences*, 32(3):265–279, 1986.
12. Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):17, 2008.
13. Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.
14. D. Stolee and N. V. Vinodchandran. Space-Efficient Algorithms for Reachability in Surface-Embedded Graphs. In *Proceedings of the 27th Annual Conference on Computational Complexity (CCC 2012)*, pages 326–333, 2012.
15. 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.