Grid Recognition: Classical and Parameterized Computational Perspectives

Siddharth Gupta
Ben Gurion University of the Negev, Beer Sheva, Israel

Guy Sa'ar
Ben Gurion University of the Negev, Beer Sheva, Israel

Meirav Zehavi
Ben Gurion University of the Negev, Beer Sheva, Israel

Abstract

Grid graphs, and, more generally, $k \times r$ grid graphs, form one of the most basic classes of geometric graphs. Over the past few decades, a large body of works studied the (in)tractability of various computational problems on grid graphs, which often yield substantially faster algorithms than general graphs. Unfortunately, the recognition of a grid graph (given a graph G, decide whether it can be embedded into a grid graph) is particularly hard – it was shown to be NP-hard even on trees of pathwidth 3 already in 1987. Yet, in this paper, we provide several positive results in this regard in the framework of parameterized complexity (additionally, we present new and complementary hardness results). Specifically, our contribution is threefold. First, we show that the problem is fixed-parameter tractable (FPT) parameterized by $k + \mathsf{mcc}$ where mcc is the maximum size of a connected component of G. This also implies that the problem is FPT parameterized by $\mathtt{td} + k$ where $\mathtt{td}$ is the treedepth of G, as $\mathtt{td} \leq \mathsf{mcc}$ (to be compared with the hardness for pathwidth 2 where $k = 3$). (We note that when k and r are unrestricted, the problem is trivially FPT parameterized by $\mathtt{td}$.) Further, we derive as a corollary that strip packing is FPT with respect to the height of the strip plus the maximum of the dimensions of the packed rectangles, which was previously only known to be in XP. Second, we present a new parameterization, denoted a_G, relating graph distance to geometric distance, which may be of independent interest. We show that the problem is para-NP-hard parameterized by a_G, but FPT parameterized by a_G on trees, as well as FPT parameterized by $k + a_G$. Third, we show that the recognition of $k \times r$ grid graphs is NP-hard on graphs of pathwidth 2 where $k = 3$. Moreover, when k and r are unrestricted, we show that the problem is NP-hard on trees of pathwidth 2, but trivially solvable in polynomial time on graphs of pathwidth 1.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Mathematics of computing → Graph algorithms

Keywords and phrases Grid Recognition, Grid Graph, Parameterized Complexity

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2021.37

Related Version *Full Version*: `https://arxiv.org/abs/2106.16180`

Funding Supported in part by the United States – Israel Binational Science Foundation (BSF) grant no. 2018302 and Israel Science Foundation (ISF) individual research grant no. 1176/18.
Siddharth Gupta: Supported in part by the Zuckerman STEM Leadership Program.
Guy Sa'ar: Supported in part by the Israeli Smart Transportation Research Center (ISTRC).

1 Introduction

Geometrically, a *grid graph* is a graph that can be drawn on the Euclidean plane so that all vertices are drawn on points having positive integer coordinates, and all edges are drawn as axis-parallel straight line segments of length 1;[1] when the maximum x-coordinate is at most r and the maximum y-coordinate is at most k, we may use the term $k \times r$ *grid graph* (see Figure 2). Grid graphs form one of the simplest and most intuitive classes of geometric graphs. Over the past few decades, algorithmic research of grid graphs yielded a large body of works on the tractability or intractability of various computational problems when restricted to grid graphs (e.g., see [14, 15, 3, 37, 48, 17, 4] for a few examples). Even for problems that remain NP-hard on grid graphs, we know of practical algorithms for instances of moderate size (e.g., the Steiner Tree problem on grid graphs is NP-hard [27], but admits practical algorithms [25, 49]). Thus, the recognition of a graph as a grid graph unlocks highly efficient tools for its analysis. In practice, grid graphs can represent layouts or environments, and have found applications in several fields, such as VLSI design [45], motion planning [32] and routing [46]. Indeed, grid graphs naturally arise to represent entities and the connections between them in *existing* layouts or environments. However, often we are given just a (combinatorial) graph G – i.e., we are given entities and the connections desired to have between them, and we are to construct the layout or environment; specifically, we wish to test whether G can be embedded into a grid graph (where if it can so, realize it as such a graph). Equivalently, the recognition of a grid graph can be viewed as an embedding problem, where a given graph is to be embedded within a rectangular solid grid.

Accordingly, the problem of recognizing (as well as realizing) grid graphs is a basic recognition problem in Graph Drawing. In what follows, we discuss only recognition – however, it would be clear that all of our results hold also for realization (with the same time complexity in case of algorithms). Formally, in the Grid Embedding problem, we are given a (simple, undirected) n-vertex graph G, and need to decide whether it can be embedded into a grid graph. In many cases, taking into account physical constraints, compactness or visual clarity, we would like to not only have a grid graph, but also restrict its dimensions. This yields the $k \times r$-Grid Embedding problem, where given an n-vertex graph G and positive integers $k, r \in \mathbb{N}$, we need to decide whether G is a $k \times r$ grid graph. Notice that Grid Embedding is the special case of $k \times r$-Grid Embedding where $k = r = n$ (which virtually means that no dimension restriction is posed).

The Grid Embedding problem has been proven to be NP-hard already in 1987, even on trees of pathwidth 3 [9]. Shortly afterwards, it has been proven to be NP-hard even on binary trees [29]. On the positive side, there is research on practical algorithms for this problem [7]. The related upward planarity testing and rectilinear planarity testing problems are also known to be NP-hard [28], as well as HV-planarity testing even on graphs of maximum degree 3 [20]. We remark that when the embedding is fixed, i.e., the clockwise order of the edges is given for each vertex, the situation becomes drastically easier computationally; then, for example, a rectangular drawing of a plane graph of maximum degree 3, as well as an orthogonal drawing without bends of a plane graph of maximum degree 3, were shown to be computable in linear time in [43] and [44], respectively.

In this paper, we study the classical and parameterized complexity of the Grid Embedding and $k \times r$-Grid Embedding problems. To the best of our knowledge, this is the first time that these problems are studied from the perspective of parameterized complexity. Let

[1] Some papers in the literature use the term grid graphs to refer to *induced* grid graphs, where we require also that every pair of vertices at distance 1 from each other have an edge between them.

Π be an NP-hard problem. In the framework of parameterized complexity, each instance of Π is associated with a *parameter* k. We say that Π is *fixed-parameter tractable (FPT)* if any instance (I, k) of Π is solvable in time $f(k) \cdot |I|^{\mathcal{O}(1)}$, where f is an arbitrary computable function of k. Nowadays, Parameterized Complexity supplies a rich toolkit to design FPT algorithms as well as to prove that some problems are unlikely to be FPT [22, 16, 24]. In particular, the term para-NP-hard refers to problems that are NP-hard even when the parameter is fixed, which implies that they are not FPT unless P=NP.

Research at the intersection of graph drawing and parameterized complexity (and parameterized algorithms in particular) is in its infancy. Most (in particular, the early efforts) have been directed at variants of the classic CROSSING MINIMIZATION problem, introduced by Turán in 1940 [47], parameterized by the number of crossings (see, e.g., [30, 38, 23, 35, 36, 39]). However, in the past few years, there is an increasing interest in the analysis of a variety of other problems in graph drawing from the perspective of parameterized complexity (see, e.g., [1, 10, 2, 31, 13, 34, 6, 19, 18, 11, 21, 41, 40].

Our Contribution and Main Proof Ideas

I. Parameterized Complexity: Maximum Connected Component Size. Our contribution is threefold. First, we prove that $k \times r$-GRID EMBEDDING is FPT parameterized by $\mathsf{mcc} + k$. Here, the idea of the proof is first to recognize all possible embeddings of any choice of connected components or parts of connected components of G into $k \times \mathsf{mcc}(G)$ grids, called blocks. These blocks then serve as vertices of a new digraph, where there is an arc from one vertex to another if and only if the corresponding blocks can be placed one after the other. After that, we also guess which blocks should occur at least once in the solution, as well as a spanning tree of the underlying undirected graph of the graph induced on them. This then leads us to a formulation of an Integer Linear Program (ILP), where we ensure that each connected component is used as many times as it is in the input, and that overall we get an Eulerian trail in the graph – having such a trail allows us to place the blocks one after the other, so that every pair of consecutive blocks are compatible. The ILP can then be solved using known tools.

▶ **Theorem 1.1.** *$k \times r$-GRID EMBEDDING is FPT parameterized by $\mathsf{mcc} + k$ where mcc is the maximum size of a connected component in the input graph.*

One almost immediate corollary of this theorem concerns the 2-STRIP PACKING problem. In this problem, we are given a set of n rectangles S, and positive integers $k, W \in \mathbb{N}$, and the objective is to decide whether all the rectangles in S can be packed in a rectangle (called a *strip*) of dimensions $k \times W$. In [5], it was shown that if the maximum of the dimensions of the input rectangles, denoted by ℓ, is fixed, then the problem is FPT by k. Thus, the question whether the problem is FPT parameterized by $k + \ell$ remained open. By a straightforward reduction, we resolve this question as a corollary of our theorem.

▶ **Corollary 1.2.** *2-STRIP PACKING is FPT parameterized by $\ell + k$ where ℓ is the maximum of the dimensions of the input rectangles.*

We remark that in case k and r are unrestricted, the problem is trivially FPT with respect to mcc, since one can embed each connected component (using brute-force) individually. This implies that GRID EMBEDDING is FPT parameterized by mcc.

As a corollary of Theorem 1.1 and the above observation, we obtain that $k \times r$-GRID EMBEDDING is FPT parameterized by $\mathtt{td}+k$, and GRID EMBEDDING is FPT parameterized by $\mathtt{td}$, where $\mathtt{td}$ is the treedepth of the input graph. This finding is of interest when contrasted with the hardness of these problems when pathwidth equals 2 and $k = 3$ or unrestricted. Thus, this also charts a tractability border between pathwidth and treedepth.

II. Parameterized Complexity: Difference Between Graph and Geometric Distances. Secondly, we introduce a new parameterization that relates graph distance to geometric distance, and may be of independent interest. Roughly speaking, the rationale behind this parameterization is to bound the difference between them, so that graph distances may act as approximate indicators to geometric distances. In particular, vertices that are close in the graph, are to be close in the embedding, and vertices that are distant in the graph, are to be distant in the embedding as well. Specifically, with respect to an embedding f of G in a grid, we define the *grid distance* between any two vertices as the distance between them in f in L1 norm. Then, we define the measure of *distance approximation* of f as the maximum of the difference between the graph distance (in G) and the grid distance of two vertices, taken over all pairs of vertices in G. Here, it is implicitly assumed that G is connected. Then, the parameter a_G is the minimum distance approximation a_f of any embedding f of G in a (possibly $k \times r$) grid, defined as $|V(G)|$ if no such embedding exists. A more formal definition as well as motivation is given in Section 2.

We first prove that the problems are para-NP-hard parameterized by a_G. This reduction is quite technical. On a high level, we present a construction of "blocks" that are embedded in a grid-like fashion, where we place an outer "frame" of the form of a grid to guarantee that the boundary (which is a cycle) of each of these blocks must be embedded as a square. Each variable is associated with a column of blocks, and each clause is associated with a row of blocks. Within each block, we place two gadgets, one which transmits information in a row-like fashion, to ensure that the clause corresponding to the row has at least one literal that is satisfied, and the other (which is very different than the first) transmits information in a column-like fashion, to ensure consistency between all blocks corresponding to the same variable (i.e., that all of them will be embedded internally in a way that represents only truth, or only false). For clarity, in the full version we split the reduction into two, and use as an intermediate problem a new problem that we call the BATTERIES problem.

▶ **Theorem 1.3.** GRID EMBEDDING *(and hence also* $k \times r$-GRID EMBEDDING*) is para-NP-hard parameterized by* a_G.

When we enrich the parameterization by k, then $k \times r$-GRID EMBEDDING problem becomes FPT. (Recall that parameterized by k alone, the problem is para-NP-hard). The idea of the proof is to partition a rectangular solid $k \times r$ grid in which we embed our graph into blocks of size $k \times (a_G + k)$, and "guess" one vertex that is to be embedded in the leftmost column of the leftmost block. Then, the crux is in the observation that, for every vertex, the block in which it should be placed is "almost" fixed – that is, we can determine two consecutive blocks in which the vertex may be placed, and then we only have a choice of one among them. This, in turn, leads us to the design of an iterative procedure that traverses the blocks from left to right, and stores, among other information, which vertices were used in the previous block.

▶ **Theorem 1.4.** $k \times r$-GRID EMBEDDING *is FPT parameterized by* $a_G + k$.

Lastly, we prove that when restricted to trees, the problems become FPT parameterized by a_G alone. Here, a crucial ingredient is to understand the structure of the tree, including a bound on the number of vertices of degree at least 3 in the tree that split it to "large" subtrees. For this, one of the central insights is that, with respect to an internal vertex v and any two "large" subtrees attached to it (there can be up to four subtrees attached to it), in order not to exceed the allowed difference between the graph and geometric distances, one of the subtrees must be embedded in the "opposite" direction of the other (so, both

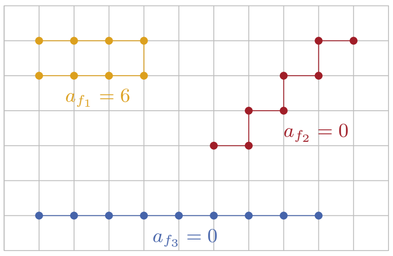

Figure 1 Example of a path P on 8 vertices with three different grid graph embeddings f_1, f_2 and f_3. Since $a_{f_2} = a_{f_3} = 0$, we get that $a_P = 0$.

are embedded roughly on the same vertical or horizontal line in opposite sides). Now, for an internal vertex of degree at least 3, there must be two attached subtrees that are not embedded in this fashion (as a line can only accommodate two subtrees), which leads us to the conclusion that all but two of the attached subtrees are small. Making use of this ingredient, we argue that a dynamic programming procedure (somewhat similar to the one mentioned for the previous theorem but much more involved) can be used.

▶ **Theorem 1.5.** *$k \times r$-Grid Embedding (and hence also Grid Embedding) on trees is FPT parameterized by a_G.*

III. Classical Complexity. Lastly, we extend current knowledge of the classical complexity of Grid Embedding and $k \times r$-Grid Embedding at several fronts. Here, we begin by developing a refinement of the classic reduction from Not-All-Equal 3SAT in [9] (which asserted hardness on trees of pathwidth 3) to derive the following result. While the reduction itself is similar, our proof is more involved and requires, in particular, new inductive arguments.

▶ **Theorem 1.6.** *Grid Embedding is NP-hard even on trees of pathwidth 2. Thus, it is para-NP-hard parameterized by* pw*, where* pw *is the pathwidth of the input graph.*

In particular, now the hardness result is *tight* with respect to pathwidth due to the simple observation that Grid Embedding is polynomial time solvable on graphs of pathwidth 1. Because Grid Embedding is a special case of $k \times r$-Grid Embedding, the above theorem has the following result as an immediate corollary: $k \times r$-Grid Embedding is NP-hard even on trees of pathwidth 2.

Additionally, we show that $k \times r$-Grid Embedding is NP-hard on graphs of pathwidth 2 even when $k = 3$. Here, we give a relatively simple reduction from 3-Partition (whose objective is to partition a set of numbers encoded in unary into sets of size 3 that sum up to the same number), where the idea is to encode "containers" by special identical connected components whose embedding is essentially fixed, and then each number as a simple path on a corresponding number of vertices.

▶ **Theorem 1.7.** *$k \times r$-Grid Embedding is NP-hard even on graphs of pathwidth 2 when $k = 3$. Thus, it is para-NP-hard parameterized by $k +$ pw, where* pw *is the pathwidth of the input graph.*

2 Preliminaries

Definitions of various standard concepts can be found in the full version. For every $k \in \mathbb{N}$, denote $[k] = \{1, 2, \dots k\}$, and for $i, j \in \mathbb{N}$, denote $[i, j] = \{i, i+1, \dots, j\}$. Given a set W of integers, $\sum W$ denotes the sum of its elements. Given a function g defined on a set W, we denote the set of images of its elements by $g(W)$. Given a graph G, we denote its vertex set and edge set by $V(G)$ and $E(G)$, respectively. For a vertex $v \in V(G)$, we denote the degree of v in G by $\deg_G(v)$. Given a set $V' \subseteq V(G)$, the subgraph of G induced by V is denoted by $G[V']$. Given $u, v \in V(G)$, the distance $d(u, v)$ between u and v in G is the length of a shortest path between them. The *pathwidth* of a graph G is denoted by $\mathtt{pw}(G)$.

We now define basic notions related to grids and grid embeddings. Let $f : V(G) \to \mathbb{N} \times \mathbb{N}$ be a function that maps each vertex v of G to a point (i, j) of an integer grid; then, i and j are also denoted as $f_{\mathsf{row}}(v)$ and $f_{\mathsf{col}}(v)$, respectively, that is, $f(v) = (f_{\mathsf{row}}(v), f_{\mathsf{col}}(v))$. Let $u, v \in V(G)$ be two vertices. The *grid graph distance of u and v induced by f*, denoted by $d_f(u, v)$, is defined to be $d_f(u, v) = |f_{\mathsf{row}}(u) - f_{\mathsf{row}}(v)| + |f_{\mathsf{col}}(u) - f_{\mathsf{col}}(v)|$. A *$k \times r$ grid graph embedding* of G is an injection $f : V(G) \to [k] \times [r]$ such that for every $\{u, v\} \in E(G)$ it follows that $d_f(u, v) = 1$. Moreover, a *grid graph embedding* of G is a $|V(G)| \times |V(G)|$ grid graph embedding of G. We say that G is a *(resp. $k \times r$) grid graph* if there exists a (resp. $k \times r$) grid graph embedding of G. Now, the $k \times r$-Grid Embedding and Grid Embedding problems are defined as follows. Given a graph G and two positive integers k, r, the $k \times r$-Grid Embedding and Grid Embedding problems ask whether G is a $k \times r$ grid graph or a grid graph, respectively.

We now define the distance approximation parameter formally and discuss some of the motivation behind it. Towards this, we first present the following simple observation. Let G be a connected grid graph with a grid embedding f, and let $u, v \in V(G)$. Then, $d_f(u, v) \leq d(u, v)$. Keeping this in mind, we drop the absolute value notation from the following definition: For any $k \times r$ grid graph embedding f of G, define $a_f = \max_{u,v \in V(G)}(d(u, v) - d_f(u, v))$. Then, if G is a $k \times r$ grid graph, let $a_G(k, r) = \min\{a_f \mid f \text{ is a } k \times r \text{ grid graph embedding of } G\}$; otherwise, $a_G(k, r) = |V(G)|$. When k and r are clear from context, we write "distance approximation parameter" and a_G rather than "$k \times r$ distance approximation parameter" and $a_G(k, r)$, respectively. When k and r are unrestricted, $a_G(k, r) = a_G(|V(G)|, |V(G)|)$. See Figure 1. We also remark that whenever G is a $k \times r$ grid graph, then $a_G(k, r) \leq |V(G)| - 2$ (because for any grid graph embedding f of G and two different vertices $u, v \in V(G)$, $d(u, v) \leq |V(G)| - 1$ and $d_f(u, v) \geq 1$).

This rationale behind this parameter makes sense in various scenarios. Suppose that vertices represent utilities, factories or organizations, or, very differently, components to be placed on a chip. On the one hand, those that are closer to each other in the graph might need to cooperate more often: they have direct and indirect (through other entities on the path) connections between them; the more "links on the chain", the less is directed interaction required. On the other hand, we may have a competitive constraint – we may want these entities to also be "as far as possible". In particular, if they are far in the graph, we will take advantage of this to place them far in the embedding (proportionally). For example, these entities may cause pollution, radiation or heat [8, 26]. Alternatively, in the case of utilities, we may want to cover as large area as we can. Recently, due to the COVID-19 pandemic, many governments around the world have introduced social distancing. Briefly, social distancing means that people should be physically away from each other, if possible. According to experts, one of the most effective ways to reduce the spread of coronavirus is social distancing [12, 33, 42]. Suppose that the vertices represent people, the edges represent social (or other) relations between them, and we want to find a seating arrangement. In order to preserve the social distancing, we would like that people who do not need to be close to each other, to be relatively far away from each other. In another example, suppose that the vertices represent some facilities that "attract" people, like stores. Placing the stores far away from each other, if possible, contributes to social distancing. More intuition is given in the full version. We remark that the embeddings that our algorithms compute satisfy the conditions of being a grid graph embedding, in particular, the embeddings are planar. Furthermore, we do not need to know the value of a_G in advance, in order to use our algorithm, as we iterate over all the potential values for a_G.

3 FPT Algorithm on General Graphs

In this section, we show that $k \times r$-Grid Embedding is FPT parameterized by $\mathsf{mcc}(G) + k$. We first give the definition of a $k \times r$ *rectangular grid graph* and some related terms. An undirected graph H is a $k \times r$ *rectangular grid graph* if there exists a bijection $f : V \to [k] \times [r]$, such that for every pair of vertices $u, v \in V(H)$, $\{v, u\} \in E(H)$ if and only if $d_f(u, v) = 1$. Given a $k \times r$ rectangular grid graph H and a corresponding bijective function f, we define the *columns* of H as follows: For every and $j \in [r]$, let $C_j(H) = \{u \in V(H) | f_{\mathsf{col}}(u) = j\}$. Clearly, $V(H) = \bigcup_{j=1}^{r} C_j(H)$. We refer to $C_1(H)$ and $C_r(H)$ as the *left boundary column* and *right boundary column*, respectively, of H.

Given a subgraph S of a $k \times r$ rectangular grid graph H, we denote by $\mathcal{FC}(S)$ the set of *fully contained* connected components of S defined as all the connected components of S that either do not intersect the boundary columns of H or intersect both boundary columns of H. Moreover, we denote by $\mathcal{LC}(S)$ ($\mathcal{RC}(S)$) the set of *left contained* (*right contained*) connected components of S defined as all the connected components of S that intersect the left (right) boundary column of H but do not intersect the right (left) boundary column of H. Note that the three sets $\mathcal{FC}(S), \mathcal{LC}(S)$ and $\mathcal{RC}(S)$ are pairwise disjoint and $S = \mathcal{FC}(S) \cup \mathcal{LC}(S) \cup \mathcal{RC}(S)$. See Figure 2. We now prove that $k \times r$-Grid Embedding is FPT.

Proof Sketch of Theorem 1.1. The FPT algorithm is based on ILP. To this end, let G be an instance of the $k \times r$-Grid Embedding problem. Due to lack of space, we only give a sketch of the algorithm. For completeness, please refer to the full version.

Algorithm. Let H be a $k \times r$ rectangular grid graph. Let $\mathcal{B} = \{B_1, B_2, \dots, B_p\}$ be an almost partition of H into blocks of size $k \times \mathsf{mcc}(G)$ such that $V(B_i) = \bigcup_{j=(i-1)(\mathsf{mcc}(G)-1)+1}^{i(\mathsf{mcc}(G)-1)+1} C_j(H)$, for each $i \in [p]$ where $p = (r-1)/(\mathsf{mcc}(G)-1)$. Here, we consider the case where $r-1$ is a multiple of $\mathsf{mcc}(G)-1$. Note that each block B_i is a $k \times \mathsf{mcc}(G)$ rectangular grid graph, and for all $i \in [p-1], B_i$ and B_{i+1} share a boundary column.

We can restate the $k \times r$-Grid Embedding problem as follows: is G a subgraph of H? As H is a planar graph, G must be planar. So, we first check if G is planar. Let $\mathsf{Comp}(G) = \{G_1, G_2, \dots, G_t\}$ be the set of all *non-isomorphic* connected components of G. For every $i \in [t]$, let $\mathsf{num}(G_i)$ be the number of times G_i appears in G. As the size of any connected component of G is at most $\mathsf{mcc}(G)$, any connected component C of G (when embedded as a subgraph of H, if possible) intersects (i) only one block B_i (in particular, it does not intersect either the right or the left boundary of B_i), or (ii) exactly two consecutive blocks B_i and B_{i+1} through the right boundary column of B_i, or (iii) exactly three consecutive blocks B_{i-1}, B_i and B_{i+1} through the left and right boundary columns of B_i.

Based on the above observation, we compute the set $\mathcal{S}$ of all the possible *snapshots* of a $k \times \mathsf{mcc}(G)$ rectangular grid graph R, i.e. the set of all the subgraphs of R, and the left and right *adjacencies* between snapshots. For every subgraph S of R, if $\mathcal{FC}(S) \subseteq \mathsf{Comp}(G)$,

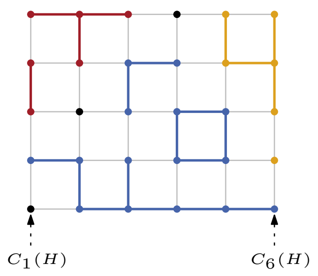

Figure 2 A 5×6 rectangular grid graph H, its boundary columns and a subgraph S of H shown by colored vertices and thick colored edges. The blue, red and orange colored connected components belong to $\mathcal{FC}(S)$, $\mathcal{LC}(S)$ and $\mathcal{RC}(S)$, respectively.

then we add S to $\mathcal{S}$. Note that, if there exists a connected component of S in $\mathcal{FC}(S)$ that does not belong to $\mathsf{Comp}(G)$, then S cannot "contribute to" a valid solution. We also find the set, denoted Source, of snapshots that may correspond to B_1 and the set, denoted Sink, of snapshots that may correspond to B_p. Note that we make this distinction, as except for blocks B_1 and B_p, all the other blocks share both boundary columns, but B_1 (B_p) only share their right (left) boundary column. So for every $S \in \mathcal{S}$, if $\mathcal{LC}(S) \subseteq \mathsf{Comp}(G)$, then add S to Source, and if $\mathcal{RC}(S) \subseteq \mathsf{Comp}(G)$, then add S to Sink. For every snapshot $S \in \mathcal{S}$ and $i \in [t]$, let $\mathsf{freq}_{\mathsf{cen}}(G_i, S)$ be the number of times G_i appears in $\mathcal{FC}(S)$. Similarly, for every snapshot $S \in \mathsf{Source}$ ($S \in \mathsf{Sink}$) and $i \in [t]$, let $\mathsf{freq}_{\mathsf{left}}(G_i, S)$ ($\mathsf{freq}_{\mathsf{right}}(G_i, S)$) be the number of times G_i appears in $\mathcal{LC}(S)$ ($\mathcal{RC}(S)$).

We now find the set $\mathsf{Adj} \subseteq \mathcal{S} \times \mathcal{S}$ of all possible adjacencies between pairs of snapshots in $\mathcal{S}$. Let R' be a $k \times (2\mathsf{mcc}(G) - 1)$ rectangular grid graph. We divide R' into two blocks B'_1 and B'_2 of size $k \times \mathsf{mcc}(G)$ such that $V(B'_1) = \bigcup_{i=1}^{\mathsf{mcc}(G)} C_i(R')$ and $V(B'_2) = \bigcup_{i=\mathsf{mcc}(G)}^{2\mathsf{mcc}(G)-1} C_i(R')$. For every $i \in \{1, 2\}$ and subgraph S' of R', let $S'_i = S'[V(S') \cap V(B'_i)]$ be the subgraph of S' in block B'_i. We look only at those subgraphs S' for which both S'_1 and S'_2 belong to $\mathcal{S}$. For every such S', we add the pair (S'_1, S'_2) to Adj if all the connected components of $S'_1 \cup S'_2 = S'$ that intersect both B'_1 and B'_2 (i.e., intersect column $C_{\mathsf{mcc}(G)}(R')$) belong to $\mathsf{Comp}(G)$. Let $\mathcal{BC}(S'_1, S'_2)$ be the set of all the connected components of $S'_1 \cup S'_2 = S'$ that intersect the column $C_{\mathsf{mcc}(G)}(R')$ but intersect neither $C_1(R')$ nor $C_{2\mathsf{mcc}(G)-1}(R')$. For every $i \in [t]$, we denote the number of times G_i appears in $\mathcal{BC}(S'_1, S'_2)$, by $\mathsf{freq}_{\mathsf{boun}}(G_i, (S'_1, S'_2))$.

For every pair of snapshots $(start, end)$ such that $start \in \mathsf{Source}$ and $end \in \mathsf{Sink}$ and a set $\mathcal{S}' \subseteq \mathcal{S}$ of snapshots, we create a directed graph D as follows. We add all the snapshots in $\mathcal{S}'$ as vertices of D, and for every pair of snapshots $S, S' \in \mathcal{S}'$, if $(S, S') \in \mathsf{Adj}$, then add an arc from S to S' in D. We then add both $start$ and end as vertices of D and for every snapshot $S \in \mathcal{S}$, if $(start, S) \in \mathsf{Adj}$ ($(S, end) \in \mathsf{Adj}$), add an arc from $start$ to S (S to end) in D. We then find the number of times $X(S, S')$, each arc (S, S') should be duplicated in D to get a new multidigraph D' such that we get a (connected) Eulerian trail in D' from $start$ to end of length p and all the connected components of G are covered by the Eulerian trail. Finally, we use the path to get the correspondence between the blocks of H and the snapshots in $\mathcal{S}'$ with a correct placement from left to right. The algorithm to find D' proceeds as follows.

- Find the set $\mathcal{T}$ of all spanning trees of the underlying undirected graph of D.
- For every spanning tree $T \in \mathcal{T}$, solve the following ILP to find $X(S, S')$ for every edge $(S, S') \in E(D)$.

$$\forall S \in V(D) \setminus \{start, end\} : \sum_{(S,S') \in E(D)} X(S, S') = \sum_{(S'',S) \in E(D)} X(S'', S). \tag{1a}$$

$$\sum_{(start,S)\in E(D)} X(start,S) = 1. \tag{1b}$$

$$\sum_{(S,end)\in E(D)} X(S,end) = 1. \tag{1c}$$

$$\sum_{(S,S')\in E(D)} X(S,S') = p-1. \tag{1d}$$

$$\forall i \in [t] : \mathsf{freq}_{\mathsf{left}}(G_i, start) + \sum_{(S,S')\in E(D)} X(S,S') \cdot \mathsf{freq}_{\mathsf{boun}}(G_i,(S,S'))+$$

$$\sum_{S\in V(D)} \Bigg(\sum_{(S,S')\in E(D)} X(S,S') \Bigg) \cdot \mathsf{freq}_{\mathsf{cen}}(G_i,S) + \mathsf{freq}_{\mathsf{right}}(G_i,end) = \mathsf{num}(G_i). \tag{1e}$$

$$\forall (S,S') \in E(T) : X(S,S') + X(S',S) \geq 1. \tag{1f}$$

$$\forall (S,S') \in E(D) \setminus E(T) : X(S,S') \geq 0. \tag{1g}$$

- If the ILP returns a feasible solution, then return Yes.

Recall that we run the algorithm for every possible D. If for none of them we return Yes, we eventually return No. Equation 1f ensures that the digraph D' is connected, and, in this context, recall that we go over all the possible spanning trees to check all the different possible connectivities between the vertices of D'. Equations 1a, 1b and 1c ensure that there exists an Eulerian trail in D from *start* to *end*. Equation 1d ensures that the total number of edges in D' is $p-1$, which in turn means that the Eulerian trail from *start* to *end* in D' is of length p, which is equal to the number of required blocks. Given a multidigraph D', each connected component of G can "contribute to" only one set out of $\mathcal{LC}(start)$, $\mathcal{BC}(S',S'')$, $\mathcal{FC}(S)$ and $\mathcal{RC}(end)$, for $S,S',S'' \in \mathcal{S}'$ such that $(S',S'') \in E(D)$, as there exists no $S \in \mathcal{S}'$ such that $(S,start) \in E(D)$ or $(end,S) \in E(D)$. So, Equation 1e ensures that all the connected components of G are covered by the path exactly once. ◀

4 Distance Approximation Parameter

In this section, we consider the distance approximation parameter, and sketch the proofs of Theorems 1.3 and 1.4. The proof of Theorem 1.5 is deferred to the full version.

4.1 Para-NP-hardness with Respect to a_G on General Graphs

We show a reduction from SAT to Grid Embedding where a_G is upper bounded by a constant if the output is a Yes instance. For this purpose, we present the *battery gadget* (see Figure 3), composed of a 13×9 rectangle. It has a *positive side* and a *negative side*, two *wire vertices* and six *synchronization vertices* attached to the top and bottom sides of the rectangle. On the top and bottom halves of the gadget, it has an optional extra edge, called the *positive voltage* and the *negative voltage*, respectively. We describe the battery gadget by a boolean pair $H = (x_1, x_2), x_1, x_2 \in \{0,1\}$ where $x_1 = 1$ (resp. $x_2 = 1$) if and only if we added the positive voltage edge (resp. negative voltage edge).

Given an instance π of SAT with variables $x_1, \ldots, x_n$ and clauses $\mu_1, \ldots, \mu_m$, the reduction output is $\mathsf{reduc}(\pi) = G_\pi$ where G_π defined as follows. G_π is composed of $m \cdot n$ battery gadgets ordered in a "matrix shape", i.e. the battery gadget $H_{i,j}$ is located at the i-th "row" and j-th "column" (see Figure 4). Every column $j \in [n]$ of gadgets corresponds to the variable x_j, and every row $i \in [m]$ of gadgets corresponds to the clause μ_i. Now, for each $i \in [m]$ and $j \in [n]$ we set $H_{i,j} = (x_1^{i,j}, x_2^{i,j})$ where $x_1^{i,j} = 0$ (resp. $x_2^{i,j} = 0$) if and only if

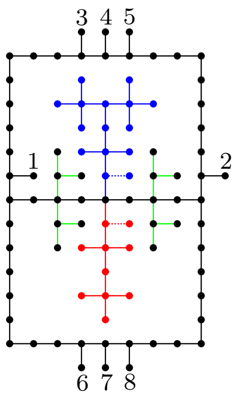

Figure 3 The battery gadget. The positive side is in blue and the negative side is in red. The positive voltage is in dashed blue and the negative voltage is in dashed red. The wire vertices are numbered $1, 2$. The synchronization vertices are numbered 3 to 8.

the literal x_j (resp. $\overline{x}_j$) appears in μ_i. We add the positive (resp. negative) voltage edge to the gadget $H_{i,j}$ if and only if the literal x_j (resp. $\bar{x}_j$) does not appear in μ_i. In addition, the "matrix" of battery gadgets is encircled by an $m \times n$-*grid frame* (see Figure 4). Lastly, we delete the "redundant" topmost and bottommost synchronization edges, i.e. those not attached to a side shared by two rectangles. More precisely, for every $j \in [n]$, we delete the three edges attached to the top side of the rectangle of $H_{1,j}$ and the three edges attached to the bottom side of the rectangle of $H_{m,j}$. Observe that each synchronization vertex is common to two battery gadgets, that is, for every $i \in [m-1]$ and $j \in [n]$, the synchronization vertices $3, 4, 5$ of $H_{i+1,j}$ are the synchronization vertices $6, 7, 8$ of $H_{i,j}$.

For the correctness of the reduction, we distinguish between "parts" (of the graph) that must have a fixed embedding and "parts" that might have several embeddings. We show that the embedding of the $m \times n$-grid frame is "almost fixed". More precisely, the embedding is fixed once we choose an embedding of three specific vertices of it. Intuitively, the "shape" of the $m \times n$-grid frame is fixed in every embedding, up to "rotation" of the frame in 90, 180 or 270 degrees, or "movement" (shifting) to another "location", which are immaterial for our purposes. In addition to the $m \times n$-grid frame, the embeddings of the sides of the rectangles, as well as the "middle crossing lines", of the battery gadgets are also fixed once we choose an embedding for the aforementioned three vertices.

Now, observe that given a battery gadget, we might be able to choose to embed the positive or the negative side on the top of the gadget. For a battery gadget H with a grid graph embedding f, we set $p_f(H) = +$ if the positive side of the gadget is embedded to the top of the gadget; otherwise the negative side is embedded to the top of the gadget, and we set $p_f(H) = -$. In addition, we denote by $V_f(H)$ the voltage of the side of the battery gadget that f embeds at the top of H. That is, given $H = (x_1, x_2)$, if $p_f(H) = +$, then $V_f(H) = x_1$, and if $p_f(H) = -$, then $V_f(H) = x_2$.

Next, we show that for any $j \in [n]$, the gadgets in the j-th column are "synchronized": for every $1 \leq i, i' \leq m$, $p_f(H_{i,j}) = p_f(H_{i',j})$. For intuition, observe that the six synchronization vertices in the battery gadget maybe embedded inside or outside the rectangle and consider two adjacent battery gadget in the same column, $H_{i,j}$ and $H_{i+1,j}$. If $p_f(H_{i+1,j}) = +$, then vertices 1 and 3 of $H_{i+1,j}$ (see Figure 3) must be embedded outside $H_{i+1,j}$, so they are embedded inside $H_{i,j}$. Then, in $H_{i,j}$ the positive side cannot be embedded at the bottom, and hence $p_f(H_{i,j}) = +$. Similarly, if $p_f(H_{i+1,j}) = -$, then vertex 4 must be embedded outside $H_{i+1,j}$, so it is embedded inside $H_{i,j}$. Then, in $H_{i,j}$ the negative side cannot be embedded at the bottom, and hence $p_f(H_{i,j}) = -$. Formally, we prove the following.

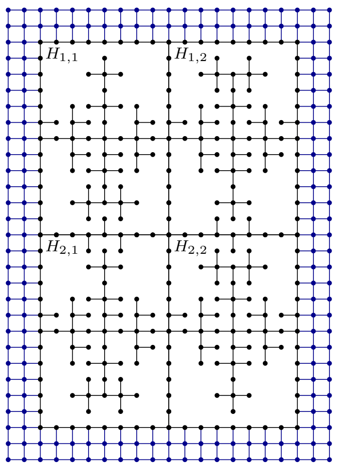

Figure 4 Construction of G_π where $\pi = (\bar{x}_1 \vee x_2) \wedge (x_1 \vee x_2)$. The 2×2-grid frame is in blue.

▶ **Lemma 4.1 (*).** *Let π be an instance of* SAT *with n variables and m clauses. Let f be a grid graph embedding of G_π. For every $j \in [n]$, there exists $p_j \in \{+,-\}$ such that for every $i \in [m]$ it follows that $p_f(H_{i,j}) = p_j$.*

Next, we show that for every $i \in [m]$, there exists $j \in [n]$ such that $V_f(H_{i,j}) = 0$. As intuition, note that if vertex 1 of $H_{i,j}$ is embedded inside $G_{i,j}$ and $V_f(G_{i,j}) = 1$, then it must be that vertex 2 is embedded outside $G_{i,j}$. So, since vertex 1 must be embedded inside $G_{i,1}$ and vertex 2 must be embedded inside $G_{i,n}$, there must be a j such that $V_f(H_{i,j}) = 0$.

▶ **Lemma 4.2 (*).** *Let π be an instance of* SAT *with n variables and m clauses. Let f be a grid graph embedding of G_π. For every $i \in [m]$, there exists $j \in [n]$ such that $V_f(H_{i,j}) = 0$.*

We are ready to prove the reverse direction of the correctness of the reduction. Due to lack of space, we omit proof of the forward direction.

▶ **Lemma 4.3.** *Let π be an instance of* SAT *with n variables and m clauses. Then G_π is a grid graph if and only π is a* Yes *instance of* SAT*.*

Partial proof. Let f be a grid graph embedding of G_π. By Lemma 4.1, for every $j \in [n]$, there exists $p_j \in \{+,-\}$ such that for every $i \in [m]$, $p_f(H_{i,j}) = p_j$. We define $s : \{x_1, \dots, x_n\} \to \{T, F\}$ as follows. For $j \in [n]$, $s(x_j) = T$ if $p_j = +$; otherwise, $s(x_j) = F$. We show that s is a satisfying assignment for π. Let $i \in [m]$. By Lemma 4.2, there exists $j \in [n]$ such that $V_p(i,j) = 0$. If $p(i,j) = p_j = +$, then $x_1^{(i,j)} = 0$, and by the definition of G_π, x_j appears in μ_i. By the definition of s, since $p_j = +$, we get that $s(x_j) = T$, therefore μ_i is satisfied. Similarly, if $p(i,j) = p_j = -$, then $x_2^{(i,j)} = 0$, and by the definition of G_π, $\bar{x}_j$ appears in μ_i. By the definition of s, since $p_j = -$, we get that $s(x_j) = F$, therefore μ_i is satisfied. So, π is a Yes instance of SAT. ◀

It only remains to show is that the distance approximation of G_π is bounded by a constant if G_π is a grid graph. Due to lack of space, we omit this proof.

▶ **Lemma 4.4 (*).** *Let π be an instance of* SAT*. If G_π is a grid graph, then for every grid graph embedding f of G_π it follows that $a_f \leq 234$.*

Using Lemma 4.3 and Lemma 4.4, we can conclude Theorem 1.3.

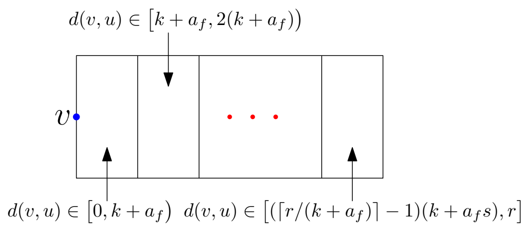
$d(v,u) \in [k + a_f, 2(k + a_f))$
v
$d(v,u) \in [0, k + a_f)$
$d(v,u) \in [(\lceil r/(k + a_f)\rceil - 1)(k + a_f s), r]$

Figure 5 Sorting the vertices u of a $k \times r$ grid into small rectangles.

4.2 $k \times r$-Grid Embedding is FPT with Respect to $k + a_G$

We present an FPT algorithm with respect to $k + a_G$ for $k \times r$-Grid Embedding. The idea is as follows. We guess a leftmost vertex v in the $k \times r$ grid (i.e. satisfying $f_{\mathsf{col}}(v) = 0$). Then, going from left to right, we divide the $k \times r$ grid into $\left\lceil \frac{r}{k+a_G} \right\rceil$ "small rectangles" of size $k \times (a_G + k)$, except the last which might be smaller. After this, we sort the vertices into the small rectangles: We put each vertex u, having graph distance $d(v,u)$ from v, in the $\left\lceil \frac{d(v,u)}{k+a_G} \right\rceil$-th rectangle; if no such rectangle exists, we put u in the $(\left\lceil \frac{d(v,u)}{k+a_G} \right\rceil - 1)$-th rectangle, if this rectangle does not exist as well (as $\left\lceil \frac{d(v,u)}{k+a_G} \right\rceil - 1$ is too large), we can conclude that we have a no-instance. In particular, we show that in every $k \times r$ grid graph embedding f of G with $a_f = a_G$ where v is a leftmost vertex, every u is embedded either in its sorted rectangle or the previous one. For intuition, observe that u cannot be embedded into a farther rectangle as then its shortest path(s) from v cannot be embedded. On the other hand, if u is embedded into a closer rectangle, then the embedding does not respect the distance approximation. After we know the "approximate location" of each vertex, we try to find a $k \times r$ grid graph embedding of G using an iterative algorithm.

We start by proving the correctness of the "location approximation" of each vertex. To this end, for any $0 \le s \le t$, we define $C_f(s,t) = \{u \in V \mid s \le f_{\mathsf{col}}(u) \le t\}$ and $D_v(s,t) = \{u \in V \mid s \le d(v,u) \le t\}$. See Figure 5.

▶ **Lemma 4.5 (*).** *Let $G = (V,E)$ be a $k \times r$ grid graph, and let f be a $k \times r$ grid graph embedding of G. Let $v \in V$ such that $f_{\mathsf{col}}(v) = 0$. Let $u \in V$ be a vertex. Then $u \in C_f(d(u,v) - a_f - k, d(u,v))$, and $u \in D_v(f_{\mathsf{col}}(u), f_{\mathsf{col}}(u) + a_f + k)$.*

Therefore, every vertex must be embedded either in its sorted rectangle or in the one to its left. In light of this, we try to find a $k \times r$ embedding of G by iteration on the small rectangles from left to right. At each step, we seek $k \times (a_f + k)$ embeddings for the vertices in the current rectangle and for a subset U of the vertices of the next rectangles. For each such embedding, we only need to store the following information: the subset U and the "right column" of the embedding (so as to "glue" embeddings of adjacent rectangles properly). In the next rectangle, we try to embed (using brute-force) the vertices we did not embed in the previous rectangle (those outside U) and some of the vertices of its next rectangle, such that the left column of the current embedding "agrees" with the right column of the embedding of the previous rectangle. Note that at each step we only store "FPT amount" of information, and so we use only "FPT runtime". A formal description of the algorithm can be found in the full version. Here, we directly proceed to state the correctness.

▶ **Lemma 4.6 (*).** *There exists an algorithm that given a graph and $k, r \in \mathbb{N}$ runs in time $\mathcal{O}(|V|^2 (k a_G)^{\mathcal{O}(k a_G + k^2)})$ and returns "*Yes *instance" if and only if G is a $k \times r$ grid graph.*

Using Lemma 4.6, we can conclude Theorem 1.4.

References

1 Parameterized complexity in graph drawing (Dagstuhl Seminar 21062). URL: https://www.dagstuhl.de/21062.

2 Akanksha Agrawal, Grzegorz Guspiel, Jayakrishnan Madathil, Saket Saurabh, and Meirav Zehavi. Connecting the Dots (with Minimum Crossings). In Gill Barequet and Yusu Wang, editors, *Symposium on Computational Geometry (SoCG 2019)*, volume 129 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2019.7.

3 Eric Allender, Tanmoy Chakraborty, David A Mix Barrington, Samir Datta, and Sambuddha Roy. Grid graph reachability problems. In *21st Annual IEEE Conference on Computational Complexity (CCC'06)*, pages 15–pp. IEEE, 2006.

4 Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Sándor P. Fekete, Joseph S. B. Mitchell, and Saurabh Sethia. Optimal covering tours with turn costs. *SIAM J. Comput.*, 35(3):531–566, 2005.

5 Pradeesha Ashok, Sudeshna Kolay, Syed Mohammad Meesum, and Saket Saurabh. Parameterized complexity of strip packing and minimum volume packing. *Theor. Comput. Sci.*, 661:56–64, 2017.

6 Michael J. Bannister, Sergio Cabello, and David Eppstein. Parameterized complexity of 1-planarity. *J. Graph Algorithms Appl.*, 22(1):23–49, 2018.

7 Moritz Beck and Sabine Storandt. Puzzling grid embeddings. In *2020 Proceedings of the Twenty-Second Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 94–105. SIAM, 2020.

8 Arnold D Bergstra, Bert Brunekreef, and Alex Burdorf. The effect of industry-related air pollution on lung function and respiratory symptoms in school children. *Environmental Health*, 17(1):1–9, 2018.

9 Sandeep N Bhatt and Stavros S Cosmadakis. The complexity of minimizing wire lengths in vlsi layouts. *Information Processing Letters*, 25(4):263–267, 1987.

10 Sujoy Bhore, Robert Ganian, Fabrizio Montecchiani, and Martin Nöllenburg. Parameterized algorithms for book embedding problems. In *Graph Drawing and Network Visualization - 27th International Symposium, GD 2019, Prague, Czech Republic, September 17-20, 2019, Proceedings*, pages 365–378, 2019.

11 Thomas Bläsius, Marcus Krug, Ignaz Rutter, and Dorothea Wagner. Orthogonal graph drawing with flexibility constraints. *Algorithmica*, 68(4):859–885, 2014.

12 CDC. Social distancing. URL: https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/social-distancing.html, November 2020.

13 Hubert Y. Chan. A parameterized algorithm for upward planarity testing. In *European Symposium on Algorithms (ESA 2004)*, volume 3221 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 2004. doi:10.1007/978-3-540-30140-0_16.

14 Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. Almost polynomial hardness of node-disjoint paths in grids. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1220–1233, 2018.

15 Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. Improved approximation for node-disjoint paths in grids with sources on the boundary. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 38:1–38:14, 2018.

16 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

17 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for eth-tight algorithms and lower bounds in geometric intersection graphs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 574–586, 2018.

18 Emilio Di Giacomo, Giuseppe Liotta, and Fabrizio Montecchiani. Sketched representations and orthogonal planarity of bounded treewidth graphs. In *Graph Drawing and Network Visualization (GD 2019)*, Lecture Notes in Computer Science. Springer, 2019. To appear. `arXiv:1908.05015`.

19 Walter Didimo and Giuseppe Liotta. Computing orthogonal drawings in a variable embedding setting. In *Algorithms and Computation (ISAAC 1998)*, volume 1533 of *Lecture Notes in Computer Science*, pages 79–88. Springer, 1998.

20 Walter Didimo, Giuseppe Liotta, and Maurizio Patrignani. On the complexity of hv-rectilinear planarity testing. In *International Symposium on Graph Drawing*, pages 343–354. Springer, 2014.

21 Walter Didimo, Giuseppe Liotta, and Maurizio Patrignani. HV-planarity: Algorithms and complexity. *J. Comput. Syst. Sci.*, 99:72–90, 2019.

22 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. `doi:10.1007/978-1-4471-5559-1`.

23 Vida Dujmovic, Michael R. Fellows, Matthew Kitching, Giuseppe Liotta, Catherine McCartin, Naomi Nishimura, Prabhakar Ragde, Frances A. Rosamond, Sue Whitesides, and David R. Wood. On the parameterized complexity of layered graph drawing. *Algorithmica*, 52(2):267–292, 2008. `doi:10.1007/s00453-007-9151-1`.

24 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.

25 Joseph L Ganley. Computing optimal rectilinear steiner trees: A survey and experimental evaluation. *Discrete Applied Mathematics*, 90(1-3):161–171, 1999.

26 Javier García-Pérez, Nerea Fernández de Larrea-Baz, Virginia Lope, Antonio J Molina, Cristina O'Callaghan-Gordo, María Henar Alonso, Marta María Rodríguez-Suárez, Benito Mirón-Pozo, Juan Alguacil, Inés Gómez-Acebo, et al. Residential proximity to industrial pollution sources and colorectal cancer risk: A multicase-control study (mcc-spain). *Environment International*, 144:106055, 2020.

27 Michael R Garey and David S. Johnson. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

28 Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM Journal on Computing*, 31(2):601–625, 2001.

29 Angelo Gregori. Unit-length embedding of binary trees on a square grid. *Information Processing Letters*, 31(4):167–173, 1989.

30 Martin Grohe. Computing crossing numbers in quadratic time. *J. Comput. Syst. Sci.*, 68(2):285–302, 2004. `doi:10.1016/j.jcss.2003.07.008`.

31 Magnús M. Halldórsson, Christian Knauer, Andreas Spillner, and Takeshi Tokuyama. Fixed-parameter tractability for non-crossing spanning trees. In *Algorithms and Data Structures (WADS 2007)*, volume 4619 of *Lecture Notes in Computer Science*, pages 410–421. Springer, 2007.

32 Dan Halperin, Oren Salzman, and Micha Sharir. Handbook of discrete and computational geometry. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 50, pages 1311–1342. CRC Press LLC, Boca Raton, FL, 2017.

33 Cone Health. Social distancing faq: How it helps prevent covid-19 (coronavirus) and steps we can take to protect ourselves. `https://www.conehealth.com/services/primary-care/social-distancing-faq-how-it-helps-prevent-covid-19-coronavirus-/`, May 2020.

34 Patrick Healy and Karol Lynch. Two fixed-parameter tractable algorithms for testing upward planarity. *Int. J. Found. Comput. Sci.*, 17(5):1095–1114, 2006. `doi:10.1142/S0129054106004285`.

35 Petr Hlinený and Marek Dernár. Crossing number is hard for kernelization. In *Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *LIPIcs*, pages 42:1–42:10. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

36 Petr Hlinený and Abhisekh Sankaran. Exact crossing number parameterized by vertex cover. In *Graph Drawing and Network Visualization (GD 2019)*, Lecture Notes in Computer Science. Springer, 2019. To appear. `arXiv:1906.06048`.

37 Alon Itai, Christos H Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

38 Ken-ichi Kawarabayashi and Bruce A. Reed. Computing crossing number in linear time. In *Symposium on Theory of Computing (STOC 2007)*, pages 382–390. ACM, 2007.

39 Fabian Klute and Martin Nöllenburg. Minimizing crossings in constrained two-sided circular graph layouts. In *Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *LIPIcs*, pages 53:1–53:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

40 Giordano Da Lozzo, David Eppstein, Michael T. Goodrich, and Siddharth Gupta. Subexponential-time and FPT algorithms for embedded flat clustered planarity. In *Graph-Theoretic Concepts in Computer Science – 44th International Workshop, WG 2018, Cottbus, Germany, June 27-29, 2018, Proceedings*, pages 111–124, 2018. `doi:10.1007/978-3-030-00256-5_10`.

41 Giordano Da Lozzo, David Eppstein, Michael T. Goodrich, and Siddharth Gupta. C-planarity testing of embedded clustered graphs with bounded dual carving-width. In *14th International Symposium on Parameterized and Exact Computation, IPEC 2019, September 11-13, 2019, Munich, Germany*, pages 9:1–9:17, 2019. `doi:10.4230/LIPIcs.IPEC.2019.9`.

42 Lisa Lockerd Maragakis. Coronavirus, social and physical distancing and self-quarantine. URL: `https://www.hopkinsmedicine.org/health/conditions-and-diseases/coronavirus/coronavirus-social-distancing-and-self-quarantine`, July 2020.

43 Md Saidur Rahman, Shin-ichi Nakano, and Takao Nishizeki. Rectangular grid drawings of plane graphs. *Computational Geometry*, 10(3):203–220, 1998.

44 Md. Saidur Rahman, Takao Nishizeki, and Mahmuda Naznin. Orthogonal drawings of plane graphs without bends. *J. Graph Algorithms Appl.*, 7(4):335–362, 2003. `doi:10.7155/jgaa.00074`.

45 Sadiq M Sait and Habib Youssef. *VLSI physical design automation: theory and practice*, volume 6. World Scientific Publishing Company, 1999.

46 Nathan R Sturtevant. Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):144–148, 2012.

47 Paul Turán. A note of welcome. *Journal of Graph Theory*, 1(1):7–9, 1977. `doi:10.1002/jgt.3190010105`.

48 Christopher Umans and William Lenhart. Hamiltonian cycles in solid grid graphs. In *Proceedings 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 496–505. IEEE, 1997.

49 Martin Zachariasen. A catalog of hanan grid problems. *Networks: An International Journal*, 38(2):76–83, 2001.