ETH-Tight Algorithms for Long Path and Cycle on Unit Disk Graphs

Fedor V. Fomin
University of Bergen, Norway, fomin@ii.uib.no

Daniel Lokshtanov
University of California, Santa Barbara, USA, daniello@ucsb.edu

Fahad Panolan
Department of Computer Science and Engineering, IIT Hyderabad, India, fahad@iith.ac.in

Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India, saket@imsc.res.in

Meirav Zehavi
Ben-Gurion University, Beer-Sheva, Israel, meiravze@bgu.ac.il

Abstract

We present an algorithm for the extensively studied Long Path and Long Cycle problems on unit disk graphs that runs in time $2^{\mathcal{O}(\sqrt{k})}(n+m)$. Under the Exponential Time Hypothesis, Long Path and Long Cycle on unit disk graphs cannot be solved in time $2^{o(\sqrt{k})}(n+m)^{\mathcal{O}(1)}$ [de Berg et al., STOC 2018], hence our algorithm is optimal. Besides the $2^{\mathcal{O}(\sqrt{k})}(n+m)^{\mathcal{O}(1)}$-time algorithm for the (arguably) much simpler Vertex Cover problem by de Berg et al. [STOC 2018] (which easily follows from the existence of a $2k$-vertex kernel for the problem), *this is the only known ETH-optimal parameterized algorithm on UDGs*. Previously, Long Path and Long Cycle on unit disk graphs were only known to be solvable in time $2^{\mathcal{O}(\sqrt{k}\log k)}(n+m)$. This algorithm involved the introduction of a new type of a tree decomposition, entailing the design of a very tedious dynamic programming procedure. Our algorithm is substantially simpler: we completely avoid the use of this new type of tree decomposition. Instead, we use a marking procedure to reduce the problem to (a weighted version of) itself on a standard tree decomposition of width $\mathcal{O}(\sqrt{k})$.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Computational geometry

Keywords and phrases Optimality Program, ETH, Unit Disk Graphs, Parameterized Complexity, Long Path, Long Cycle

Lines 499

1 Introduction

Unit disk graphs are the intersection graphs of disks of radius 1 in the Euclidean plane. That is, given n disks of radius 1, we represent each disk by a vertex, and insert an edge between two vertices if and only if their corresponding disks intersect. Unit disk graphs form one of the most well studied graph classes in computational geometry because of their use in modelling optimal facility location [56] and broadcast networks such as wireless, ad-hoc and sensor networks [35, 45, 58]. These applications have led to an extensive study of NP-complete problems on unit disk graphs in the realms of computational complexity and approximation algorithms. We refer the reader to [16, 24, 38] and the citations therein for these studies. However, these problems remain hitherto unexplored in the light of parameterized complexity with exceptions that are few and far between [1, 14, 33, 42, 54].

We study the Long Path (resp. Long Cycle) problem on unit disk graphs. Here, given a graph G and an integer k, the objective is to decide whether G contains a path (resp. cycle) on at least k vertices. To the best of our knowledge, the Long Path problem is among the

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

five most extensively studied problems in Parameterized Complexity [17, 23] (see Section 1.1). One of the most well known open problems in Parameterized Complexity was to develop a $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$-time algorithm for Long Path on general graphs [52], that is, shaving the $\log k$ factor in the exponent of the previously best $2^{\mathcal{O}(k \log k)}n^{\mathcal{O}(1)}$-time parameterized algorithm for this problem on general graphs [49]. This was resolved in the positive in the seminal work by Alon, Yuster and Zwick on color coding 25 years ago [5], which was recently awarded the IPEC-NERODE prize for the most outstanding research in Parameterized Complexity. In particular, the aforementioned $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$-time algorithm for Long Path on general graphs is optimal under the Exponential Time Hypothesis (ETH).

Both Long Path and Long Cycle are known to be NP-hard on unit disk graphs [40], and cannot be solved in time $2^{o(\sqrt{n})}$(and hence also in time $2^{o(\sqrt{k})}n^{\mathcal{O}(1)}$) on unit disk graphs unless the ETH fails [19]. Our contribution is an *optimal* parameterized algorithm for Long Path (and Long Cycle) on unit disk graphs under the ETH. Specifically, we prove the following theorem.

▶ **Theorem 1.** Long Path *and* Long Cycle *are solvable in time* $2^{\mathcal{O}(\sqrt{k})}(n+m)$ *on unit disk graphs.*

Two years ago, a celebrated work by de Berg et al. [19] presented (non-parameterized) algorithms with running time $2^{\mathcal{O}(\sqrt{n})}$ for a number of problems on intersection graphs of so called "fat", "similarly-sized" geometric objects for a number of problems, accompanied by matching lower bounds of $2^{\Omega(\sqrt{n})}$ under the ETH. Only for the Vertex Cover problem does this work implies an ETH-tight parameterized algorithm. More precisely, Vertex Cover admits a $2k$-vertex kernel on general graphs [50, 17], hence the $2^{\Omega(\sqrt{n})}$-time algorithm for Vertex Cover by de Berg et al. [19] is trivially a $2^{\Omega(\sqrt{k})}n^{\mathcal{O}(1)}$-time parameterized algorithm for this problem. None of the other problems in [19] is known to admit a linear-vertex kernel, and we know of no other work that presents a $2^{\Omega(\sqrt{k})}n^{\mathcal{O}(1)}$-time parameterized algorithm for any basic problem on unit disk graphs. Thus, we present the second known ETH-tight parameterized algorithm for a basic problem on unit disk graphs, or, in fact, on any family of geometric intersection graphs of fat objects. In a sense, our work is the first time where tight ETH-optimality of parameterized algorithms on unit disk graphs is explicitly answered. (The work of de Berg et al. [19] primarily concerned non-parameterized algorithms.) We believe that our work will open a door to the realm to an ETH-tight optimality program for parameterized algorithms on intersection graphs of fat geometric objects.

Prior to our work, Long Path and Long Cycle were known to be solvable in time $2^{\mathcal{O}(\sqrt{k}\log k)}(n+m)$ on unit disk graphs [32]. Thus, we shave the $\log k$ factor in the exponent in the running time, and thereby, in particular, achieve optimality. Our algorithm is substantially simpler, both conceptually and technically, than the previous algorithm as we explain below. The main tool in the previous algorithm (of [32].) for Long Path (and Long Cycle) on unit disk graphs was a new (or rather refined) type of a tree decomposition.[1] The width of the tree decomposition constructed in [32]. is $k^{\mathcal{O}(1)}$, which on its own does not enable to design a subexponential (or even single-exponential) time algorithm. However, each of its bags (of size $k^{\mathcal{O}(1)}$) is equipped with a partition into $\mathcal{O}(\sqrt{k})$ sets such that each of them induces a clique. By establishing a property that asserts the existence of a solution (if at least one solution exists) that crosses these cliques "few" times, the tree decomposition can be exploited. Specifically, this exploitation requires to design a very tedious dynamic programming algorithm (significantly more technical than algorithms over "standard" tree

[1] We refer the reader to Section 2 for the definition of a tree decomposition.

decompositions, that is, tree decompositions of small width) to keep track of the interactions between the cliques in the partitions.

We completely avoid the use of the new type of tree decomposition of [32]. Instead, we use a simple marking procedure to reduce the problem to (a weighted version of) itself on a tree decomposition of width $\mathcal{O}(\sqrt{k})$. Then, the new problem can be solved by known algorithms as black boxes by employing either an essentially trivial $\mathtt{tw}^{\mathcal{O}(\mathtt{tw})}n$-time algorithm, or a more sophisticated $2^{\mathcal{O}(\mathtt{tw})}n$-time algorithm (of [10] or [28]). On a high level, we are able to mark few vertices in certain cliques (which become the cliques in the above mentioned partitions of bags in [32]), so that there exists a solution (if at least one solution exists) that uses only marked vertices as "portals"—namely, it crosses cliques only via edges whose both endpoints are marked. Then, in each clique, we can just replace all unmarked vertices by a single weighted vertex. This reduces the size of each clique to be constant, and yields a tree decomposition of width $\mathcal{O}(\sqrt{k})$. We believe that our idea of identification of portals and replacement of all non-portals by few weighted vertices will find further applications in the design of ETH-tight parameterized algorithms on intersection graphs of fat geometric objects.

Before we turn to briefly survey some additional related works, we would like to stress that shaving off logarithmic factors in the exponent of running times of parameterized algorithms is a major issue in this field. Indeed, when they appear in the exponent, logarithmic factors have a *critical* effect on efficiency that can render algorithms impractical even on small instances. Over the past two decades, most fundamental techniques in Parameterized Complexity targeted not only the objective of eliminating the logarithmic factors, but even improving the base c in running times of the form $c^k n^{\mathcal{O}(1)}$. For example, this includes the aforementioned color coding technique [5] that was developed to shave off the $\log k$ in a previous $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$-time algorithm, which further entailed a flurry of research on techniques to improve the base of the exponent (see Section 1.1), and the cut-and-count technique to design parameterized algorithms in time $2^{\mathcal{O}(t)} n^{\mathcal{O}(1)}$ rather than $2^{\mathcal{O}(t \log t)} n^{\mathcal{O}(1)}$ (in fact, for connectivity problems such as Long Path) on graphs of treewidth t [18]. Accompanying this active line of research, much efforts were devoted to prove that problems that have long resisted the design of algorithms without logarithmic factors in the exponent are actually unlikely to admit such algorithms [48].

1.1 Related Works on Long Path and Long Cycle

We now briefly survey some known results in Parameterized Complexity on Long Path and Long Cycle. Clearly, this survey is illustrative rather than comprehensive. The standard parameterization of Long Path and Long Cycle is by the solution size k, and here we will survey only results that concern this parameterization.

The Long Path (parameterized by the solution size k on general graphs) is arguably one of the five (or even fewer) problems with the richest history in Parameterized Complexity, having parameterized algorithms continuously developed since the early days of the field and until this day. The algorithms developed along the way gave rise to some of the most central techniques in the field, such as color-coding [5] and its incarnation as divide-and-color [15], techniques based on the polynomial method [46, 47, 57, 8], and matroid based techniques [29]. The first parameterized algorithm for this problem was an $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$-time given in 1985 by Monien [49], even before the term "parameterized algorithm" was in known use. Originally in 1994, the logarithmic factor was shaved off [5], resulting in an algorithm with running time $c^k n^{\mathcal{O}(1)}$ for $c = 2e$. After that, a long line of works that presented improvements over c has followed [46, 47, 57, 8, 29, 59, 37, 53, 15, 55], where the algorithm with the current best running time is a randomized algorithm whose time complexity is $1.66^k n^{\mathcal{O}(1)}$ [8]. Unless the ETH fails, Long Path (as well as Long Cycle) does not admit any algorithm with running time $2^{o(k)}n^{\mathcal{O}(1)}$ [39].

For a long time, the Long Cycle problem was considered to be significantly harder than Long Path due to the following reason: while the existence of a path of size at least k implies the existence of a path of size exactly k, the existence of a cycle of size at least k does not imply the existence of a cycle of size exactly k—in fact, the only cycle of size at least k in the input graph might be a Hamiltonian cycle. Thus, for this problem, the first parameterized algorithm appeared (originally) only in 2004 [34], and the first parameterized algorithm with running time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ appeared (originally) only in 2014 [29]. Further improvements on the base of the exponent in the running time were given in [60, 30]. Lastly, we remark that due to their importance, over the past two decades there has been extensive research of Long Path and Long Cycle parameterized by k above some guarantee [7, 26, 43, 27], and the (approximate) counting versions of these problems [25, 6, 2, 4, 3, 13, 9]. Both Long Path and Long Cycle are unlikely to admit a polynomial kernel [11], and in fact, are even conjectured not to admit Turing kernels [36, 44].

While Long Path and Long Cycle remain NP-complete on planar graphs, they admit $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$-time algorithms: By combining the bidimensionality theory of Demaine et al. [20] with efficient algorithms on graphs of bounded treewidth [22], Long Path and Long Cycle, can be solved in time $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$ on planar graphs. Moreover, the parameterized subexponential "tractability" of Long Path/Cycle can be extended to graphs excluding some fixed graph as a minor [21]. Unfortunately, unit disk graphs are somewhat different than planar graphs and H-minor free graphs—in particular, unlike planar graphs and H-minor free graphs where the maximum clique size is bounded by 5 (for planar graphs) or some other fixed constant (for H-minor free graphs), unit disk graphs can contain cliques of arbitrarily large size and are therefore "highly non-planar". Nevertheless, Fomin et al. [33] were able to obtain subexponential parameterized algorithms of running time $2^{\mathcal{O}(k^{0.75}\log k)}n^{\mathcal{O}(1)}$ on unit disk graphs for Long Path, Long Cycle, Feedback Vertex Set and Cycle Packing. None of these four problems can be solved in time $2^{o(\sqrt{n})}$(and hence also in time $2^{o(\sqrt{k})}n^{\mathcal{O}(1)}$) on unit disk graphs unless the ETH fails [19]. Afterwards (originally in 2017), Fomin et al. [32] obtained improved, yet technically quite tedious, $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$-time algorithms for Long Path, Long Cycle and Feedback Vertex Set and Cycle Packing. Recall that this work was discussed earlier in the introduction. Later, the same set of authors designed $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$ time algorithms for the aforementioned problems on map graphs [31]. We also remark that recently, Panolan et al. [51] proved a contraction decomposition theorem on unit disk graphs as an application of the theorem, they proved that Min-Bisection on unit disk graphs can be solved in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$.

2 Preliminaries

For a positive integer ℓ, let $[\ell] = \{1, \ldots, \ell\}$. We refer to Appendix A for standard graph theoretic terms.

Unit disk graphs. Let $P = \{p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n)\}$ be a set of points in the Euclidean plane. Let $D = \{d_1, d_2, \ldots, d_n\}$ where for every $i \in [n]$, d_i is the disk of radius 1 whose centre is p_i. Then, the unit disk graph of D is the graph G such that $V(G) = D$ and $E(G) = \{\{d_i, d_j\} \mid d_i, d_j \in D, i \neq j, \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq 2\}$.

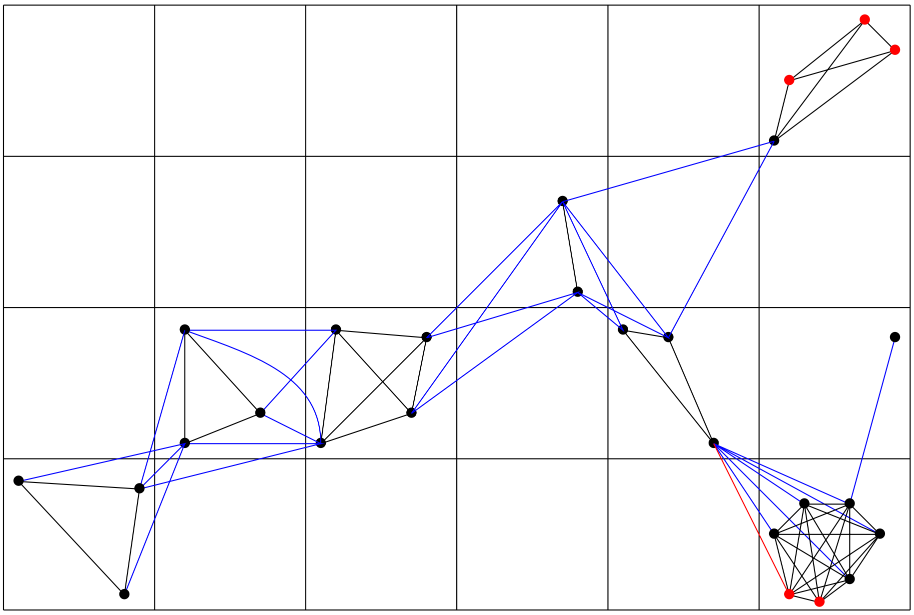


Figure 1 A clique-grid graph G. For the sake of illustration, in phase II of marking let $\mathsf{Mark}_2(v,(i',j'))$ denote a set of 5 vertices in $f^{-1}(i',j')$ that are adjacent to v in G, where if no 5 vertices with this property exist, then let $\mathsf{Mark}_2(v,(i',j'))$ denote the set of all vertices with this property. Then the good and bad edges are colored blue and red, respectively (see Definition 7). The marked vertices are colored black.

Clique-Grids. Intuitively, a clique-grid is a graph whose vertices can be embedded in grid cells (where multiple vertices can be embedded in each cell), so that the each cell induces a clique and "interacts" (via edges incident to its vertices) only with cells at "distance" at most 2 (see Figure 1).

▶ **Definition 2 (Clique-Grid).** *A graph G is a* clique-grid *if there exists a function $f : V(G) \to [t] \times [t]$ for some $t \in \mathbb{N}$, called a* representation*, such that the following conditions are satisfied.*

1. *For all $(i,j) \in [t] \times [t]$, $G[f^{-1}(i,j)]$ is a clique.*
2. *For all $\{u,v\} \in E(G)$, $|i-i'| \leq 2$ and $|j-j'| \leq 2$ where $f(u) = (i,j)$ and $f(v) = (i',j')$.*

We call a pair $(i,j) \in [t] \times [t]$ a *cell*. It is easy to see that a unit disk graph is a clique-grid, and a representation of it, can be computed in linear time. A formal proof can be found in [32] (also see [41] for a similar result). Specifically, we will refer to the following proposition.

▶ **Proposition 3** ([41, 32]). *Let G be the unit disk graph of a set of points D in the Euclidean plane. Then, G is a clique-grid, and a representation of G can be computed in linear time.*

Treewidth. The treewidth of a graph, which is a standard measure of its "closeness" to a tree, whose formal definition (not explicitly used in this paper) can be found in Appendix A. The treewidth of a graph can be approximated within a constant factor efficiently as follows.

▶ **Proposition 4** ([12]). *Given a graph G and a positive integer k, in time $2^{\mathcal{O}(k)} \cdot n$, we can either decide that $\mathtt{tw}(G) > k$ or output a tree decomposition of G of width $5k$.*

We will need the following proposition to argue that a unit disk graph *of bounded degree* contains a grid minor of dimension *linear* in its treewidth.

▶ **Proposition 5** ([33]). *Let G be a unit disk graph with maximum degree Δ and treewidth $\mathtt{tw}$. Then, G contains a $\frac{\mathtt{tw}}{100\Delta^3} \times \frac{\mathtt{tw}}{100\Delta^3}$ grid as a minor.*

3 Marking Scheme

In this section, we present a marking scheme whose purpose is to mark a constant number of vertices in each cell of a clique-grid G so that, if G has a path (resp. cycle) on at least k vertices, then it also has a path (resp. cycle) on at least k vertices that "crosses" cells only at marked vertices. Then, we further argue that unmarked vertices in a cell can be thought of, in a sense, as a "unit" that is representable by one weighted vertex. We remark that we did not make any attempt to optimize the number of vertices marked, but only make the proof simple.

Marking Scheme. Let G be a clique-grid graph with representation $f : V(G) \to [t] \times [t]$. Then, the marking scheme consists of two phases defined as follows.

Phase I. For each pair of distinct cells $(i, j), (i', j') \in [t] \times [t]$ with $|i - i'| \leq 2$ and $|j - j'| \leq 2$, let M be a maximal matching where each edge has one endpoint in $f^{-1}(i, j)$ and the other endpoint in $f^{-1}(i', j')$; if $|M| \leq 241$, then denote $\mathsf{Mark}_1(\{(i, j), (i', j')\}) = M$, and otherwise choose a subset M' of M of size 241 and let $\mathsf{Mark}_1(\{(i, j), (i', j')\}) = M'$.

For each cell $(i, j) \in [t] \times [t]$, let $\mathsf{Mark}_1(i, j)$ denote the set of all vertices in $f^{-1}(i, j)$ that are endpoints of edges in $\bigcup_{(i',j')} \mathsf{Mark}_1(\{(i, j), (i', j')\})$ where (i', j') ranges over every cell such that $|i - i'| \leq 2$ and $|j - j'| \leq 2$; the vertices that belong to this set are called *marked vertices*.

Phase II. For each ordered pair of distinct cells $(i, j), (i', j') \in [t] \times [t]$ with $|i - i'| \leq 2$ and $|j - j'| \leq 2$ and vertex $v \in \mathsf{Mark}_1(i, j)$, let $\mathsf{Mark}_2(v, (i', j'))$ denote a set of 121 vertices in $f^{-1}(i', j')$ that are adjacent to v in G, where if no 121 vertices with this property exist, then let $\mathsf{Mark}_2(v, (i', j'))$ denote the set of all vertices with this property; the vertices that belong to this set are also called *marked vertices*.

Altogether. For each cell $(i, j) \in [t] \times [t]$, let $\mathsf{Mark}^\star(i, j)$ denote the set of all marked vertices in $f^{-1}(i, j)$.

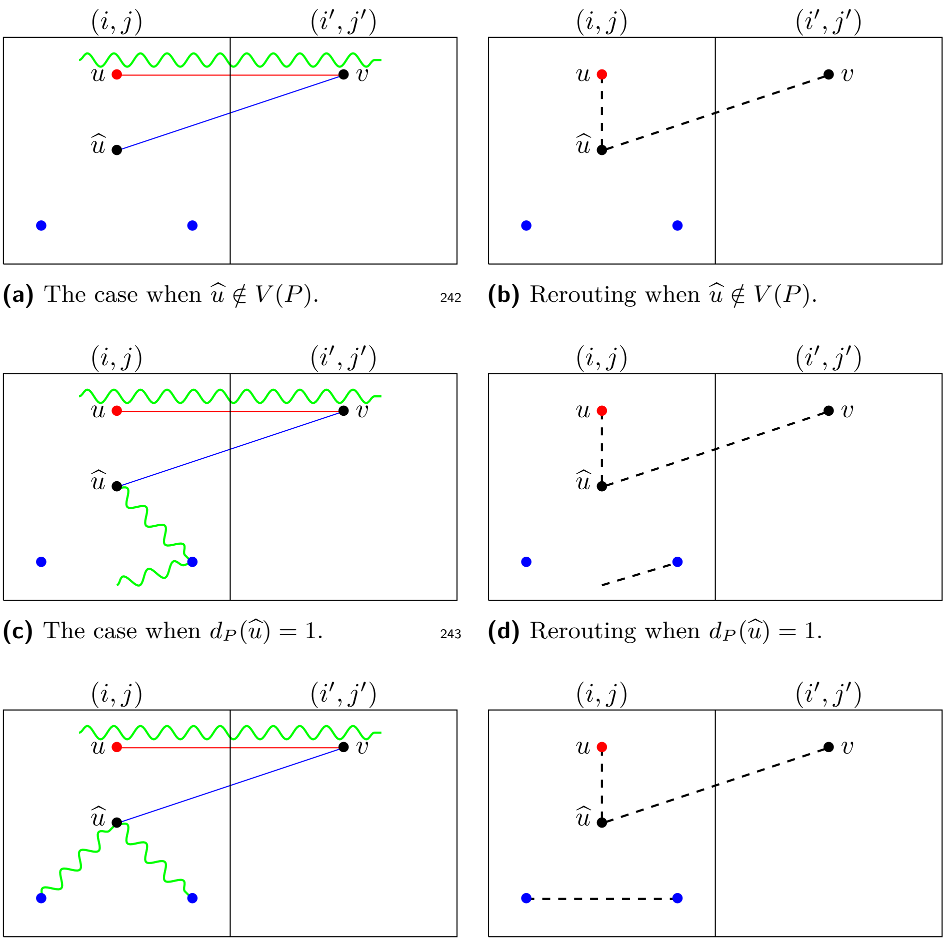
Clearly, given G and f, $\mathsf{Mark}^\star(i, j)$ is not uniquely defined. Whenever we write $\mathsf{Mark}^\star(i, j)$, we refer to an arbitrary set that can be the result of the scheme above. We have the following simple observation regarding the size of $\mathsf{Mark}^\star(i, j)$ and the computation time.

▶ **Observation 3.1.** *Let G be a clique-grid with representation $f : V(G) \to [t] \times [t]$. For each cell $(i, j) \in [t] \times [t]$, $|\mathsf{Mark}^\star(i, j)| \leq 10^{10}$. Moreover, the computation of all the sets $\mathsf{Mark}^\star(i, j)$ together can be done in linear time.*

Proof. Consider a cell $(i, j) \in [t] \times [t]$. In the first phase, at most $24 \cdot 241$ vertices in $f^{-1}(i, j)$ are marked. In the second phase, for each of the 24 cells (i', j') such that $|i - i'| \leq 2$ and $|j - j'| \leq 2$, and each of the at most $24 \cdot 241$ marked vertices in $f^{-1}(i', j')$, at most 121 new vertices in $f^{-1}(i, j)$ are marked. Therefore, in total at most $24 \cdot 241 + 24 \cdot (24 \cdot 241) \cdot 121 \leq 10^{10}$ vertices in $f^{-1}(i, j)$ are marked.

The claim regarding the computation time is immediate. ◀

As part of the proof that our marking scheme has the property informally stated earlier, we will use the following proposition.



(a) The case when $\widehat{u} \notin V(P)$. **(b)** Rerouting when $\widehat{u} \notin V(P)$.

(c) The case when $d_P(\widehat{u}) = 1$. **(d)** Rerouting when $d_P(\widehat{u}) = 1$.

(e) The case when $d_P(\widehat{u}) = 2$. **(f)** Rerouting when $d_P(\widehat{u}) = 2$.

Figure 2 Illustration of Case I in the proof of Lemma 8. The vertices colored black and red are marked and unmarked, respectively. The blue colored vertices are either marked or unmarked. Good and bad edges are colored blue and red, respectively. The curves colored green are part of the path P. The dashed lines are part of the path P_2.

▶ **Proposition 6** ([32]). *Let G be a clique-grid with representation f that has a path (resp. cycle) on at least k vertices. Then, G also has a path (resp. cycle) P on at least k vertices with the following property: for every two distinct cells (i, j) and (i', j'), there exist at most 5 edges $\{u, v\} \in E(P)$ such that $f(u) = (i, j)$ and $f(v) = (i', j')$.*

We now formally state and prove the property achieved by our marking scheme. For this purpose, we have the following definition (see Figure 1) and lemma.

▶ **Definition 7.** *Let G be a clique-grid with representation f. An edge $\{u, v\} \in E(G)$ where $f(u) \neq f(v)$ is* good *if $u \in \mathsf{Mark}^{\star}(i, j)$ and $v \in \mathsf{Mark}^{\star}(i', j')$ where $f(u) = (i, j)$ and $f(v) = (i', j')$; otherwise, it is* bad.

Intuitively, the following lemma asserts the existence of a solution (if any solution exists) that crosses different cells only via good edges, that is, via marked vertices.

▶ **Lemma 8.** *Let G be a clique-grid with representation f that has a path (resp. cycle) on at least k vertices. Then, G also has a path (resp. cycle) P on at least k vertices with the following property: every edge $\{u,v\} \in E(P)$ where $f(u) \neq f(v)$ is good.*

Proof. By Proposition 6, G has a path (resp. cycle) on at least k vertices with the following property: for every two distinct cells (i,j) and (i',j'), there exist at most 5 edges $\{u,v\} \in E(P)$ such that $f(u) = (i,j)$ and $f(v) = (i',j')$. Among all such paths (resp. cycles), let P be one that minimizes the number of bad edges. The following claim follows immediately from the choice of P and Property 2 in Definition 2.

▷ Claim 9. For each cell $(i,j) \in [t] \times [t]$, there are at most $24 \cdot 5 = 120$ vertices in $f^{-1}(i,j) \cap V(P)$ that are adjacent in P to at least one vertex that does not belong to $f^{-1}(i,j)$.

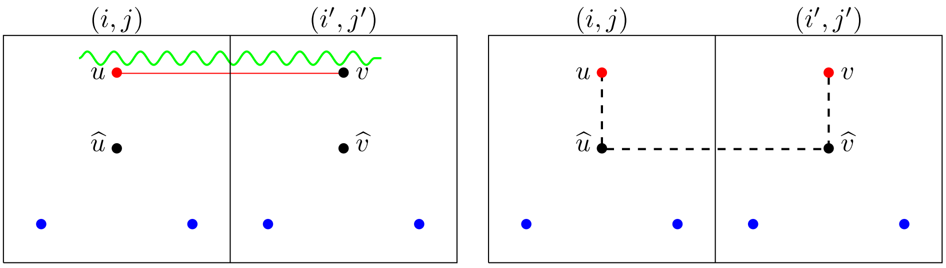
Next, we show that P has no bad edge, which will complete the proof. Targeting a contradiction, suppose that P has some bad edge $\{u,v\}$. By Definition 7, $u \notin \mathsf{Mark}^\star(i,j)$ or $v \notin \mathsf{Mark}^\star(i',j')$ (or both) where $f(u) = (i,j)$ and $f(v) = (i',j')$. Without loss of generality, suppose that $u \notin \mathsf{Mark}^\star(i,j)$. We consider two cases as follows.

Case I. First, suppose that $v \in \mathsf{Mark}_1(i',j')$. Because u is adjacent to v but it is not marked in the second phase, it must hold that $|\mathsf{Mark}_2(v,(i,j))| \geq 121$. By Claim 9, this means that there exists a vertex $\widehat{u} \in \mathsf{Mark}_2(v,(i,j)) \cap V(P)$ whose neighbors in P—which might be 0 if $\widehat{u}$ does not belong to P, 1 if it is an endpoint of P or 2 if it is an internal vertex of P—also belong to $f^{-1}(i,j)$ (see Figure 2). In case $\widehat{u} \notin V(P)$, denote $P_1 = P$. Else, by Property 1 in Definition 2, by removing $\widehat{u}$ from P, and if $\widehat{u}$ has two neighbors on P, then also making these two neighbors adjacent,[2] we still have a path (resp. cycle) in G, which we denote by P_1, whose size is at least $|V(P)| - 1$. Now, note that because $\widehat{u} \in \mathsf{Mark}_2(v,(i,j))$, we have that $\widehat{u}$ is adjacent to v in G and also $\widehat{u} \in f^{-1}(i,j)$. Because $u \in f^{-1}(i,j)$, Property 1 in Definition 2 implies that $\widehat{u}$ is also adjacent to u. Thus, by inserting $\widehat{u}$ between u and v in P_1 and making it adjacent to both, we still have a path (resp. cycle) in G, which we denote by P_2 (see Figure 2). Note that $|V(P_2)| = |V(P_1)| + 1 \geq |V(P)| \geq k$. Moreover, the only edges that appear only in one among P_2 and P are as follows.

1. If $\widehat{u}$ has two neighbors in P, then the edges between $\widehat{u}$ and these two neighbors might belong only to P, and the edge between these two neighbors belongs only to P_2. As $\widehat{u}$ and its neighbors in P belong to the same cell (by the choice of $\widehat{u}$), none of these edges is bad, and also none of these edges crosses different cells.
2. If $\widehat{u}$ has only one neighbor in P, then the edge between $\widehat{u}$ and this neighbor might belong only to P.
3. $\{u,v\} \in E(P) \setminus E(P_2)$ is a bad edge that crosses different cells by its initial choice.
4. $\{u,\widehat{u}\}$ might belong only to P_2, and it is a neither a bad edge nor an edge that crosses different cells because u and $\widehat{u}$ belong to the same cell.
5. $\{\widehat{u},v\} \in E(P_2) \setminus E(P)$ is a not a bad edge because both $\widehat{u}$ and v are marked (since $v \in \mathsf{Mark}_1(i',j')$ and $\widehat{u} \in \mathsf{Mark}_2(v,(i,j))$), but it crosses different cells.

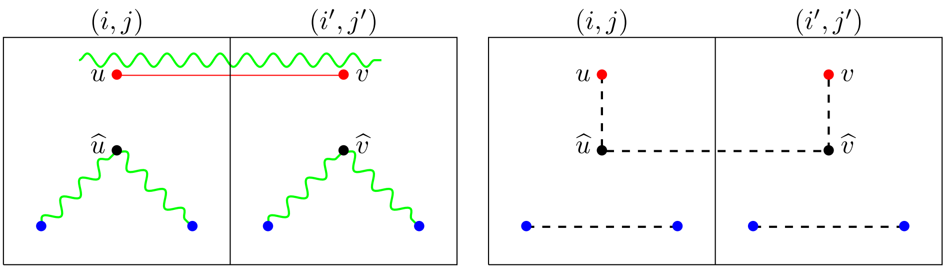
Thus, P_2 has no bad edge that does not belong to P, and P has at least one bad edge that does not belong to P_2 (specifically, $\{u,v\}$), and therefore P_2 has fewer bad edges than P. Moreover, notice that the items above also imply that P_2 has at most one edge that crosses different cells and does not belong to P (specifically, $\{\widehat{u},v\}$), and P has at least one edge that crosses the *same* cells and does not belong to P_2 (specifically, $\{u,v\}$). Therefore, P_2

[2] If $\widehat{u}$ is an endpoint of P, then only the removal of $\widehat{u}$ is performed.



(a) The case when $\widehat{u}, \widehat{v} \notin V(P)$.

(b) Rerouting when $\widehat{u}, \widehat{v} \notin V(P)$.



(c) The case when $d_P(\widehat{u}) = d_P(\widehat{v}) = 2$.

(d) Rerouting when $d_P(\widehat{u}) = d_P(\widehat{v}) = 2$.

Figure 3 Illustration of two subcases of Case II in the proof of Lemma 8. Other subcases are handled similarly to the subcases depicted here. The vertices colored black and red are marked and unmarked, respectively. The blue colored vertices are either marked or unmarked. Good and bad edges are colored blue and red, respectively. The curves colored green are part of the path P. The dashed lines are part of the path P_2.

also has the property of P that for every two distinct cells $(\widetilde{i}, \widetilde{j})$ and $(\widetilde{i}', \widetilde{j}')$, there exist at most 5 edges $\{\widetilde{u}, \widetilde{v}\} \in E(P_2)$ such that $f(\widetilde{u}) = (\widetilde{i}, \widetilde{j})$ and $f(\widetilde{v}) = (\widetilde{i}', \widetilde{j}')$. Therefore, we have reached a contradiction to the minimality of the number of bad edges in our choice of P.

Case II. Second, suppose that $v \notin \mathsf{Mark}^\star(i', j')$. Then, the addition of $\{u, v\}$ to $\mathsf{Mark}_1(i, j)$ maintains the property that it is a matching. Therefore, because this edge was not marked in the first phase, it must hold that $|\mathsf{Mark}_1(\{(i,j),(i',j')\})| = 241$. By Claim 9, there are at most 120 vertices in $f^{-1}(i,j) \cap V(P)$ that are adjacent in P to at least one vertex that does not belong to $f^{-1}(i,j)$, and notice that u (which is unmarked) is one of them. Similarly, there are at most 120 vertices in $f^{-1}(i',j') \cap V(P)$ that are adjacent in P to at least one vertex that does not belong to $f^{-1}(i',j')$, and notice that v (which is unmarked) is one of them. Therefore, because $\mathsf{Mark}_1(\{(i,j),(i',j')\})$ is a matching, it must contain at least one edge $\{\widehat{u}, \widehat{v}\}$ such that neither $\widehat{u}$ nor $\widehat{v}$ has a neighbor in P that belongs to a different cell than itself (see Figure 3)—either because $\widehat{u}$ (and in the same way $\widehat{v}$) does not belong to P, or it does and all its (one or two) neighbors belong to the same cell as itself. Define P_1' as follows: if $\widehat{u}$ does not belong to P, then $P_1' = P$, and otherwise let it be the graph obtained by removing $\widehat{u}$ from P and making its two neighbors (if both exist) adjacent. Because these two neighbors (if they exist) belong to the same cell, Property 1 in Definition 2 implies that P_1' is a path (resp. cycle) in G. Similarly, let P_1 be the path (resp. cycle) obtained by the same operation with respect to P_1' and $\widehat{v}$. Now, let P_2 be the graph obtained from P_1 by inserting $\widehat{u}$ and $\widehat{v}$ between u and v with the edges $\{u, \widehat{u}\}, \{\widehat{u}, \widehat{v}\}$ and $\{\widehat{v}, v\}$ (see Figure 3). Because of Property 1 in Definition 2, and since u and $\widehat{u}$ belong to the same cell, they are adjacent in G. Similarly, v and $\widehat{v}$ are adjacent in G. Moreover, because $\{\widehat{u}, \widehat{v}\} \in \mathsf{Mark}_1(\{(i,j),(i',j')\})$, it is an edge in G. Thus, P_2 is a path (resp. cycle) in G. Additionally, $V(P) \subseteq V(P_2)$, and therefore $|V(P_2)| \geq k$. The only edges that appear only in one among P_2 and P are as follows.

1. If $\widehat{u}$ belongs to P and has two neighbors in P, then the edges between $\widehat{u}$ and these two neighbors might belong only to P, and the edge between these two neighbors belongs only to P_2. As $\widehat{u}$ and its neighbors in P belong to the same cell (by the choice of $\widehat{u}$), none of these edges is bad, and none of them crosses different cells. The same holds for $\widehat{v}$.
2. If $\widehat{u}$ belongs to P and has only one neighbor in P, the edge between $\widehat{u}$ and this neighbor might belong only to P. The same holds for $\widehat{v}$.
3. $\{u, v\} \in E(P) \setminus E(P_2)$ is a bad edge that crosses different cells by its initial choice.
4. $\{u, \widehat{u}\}$ might belong only to P_2, and it is a neither a bad edge nor it crosses different cells because u and $\widehat{u}$ belong to the same cell. The same holds for $\{v, \widehat{v}\}$.
5. $\{\widehat{u}, \widehat{v}\} \in E(P_2) \setminus E(P)$ is a not a bad edge because both $\widehat{u}$ and $\widehat{v}$ are marked (since $\{\widehat{u}, \widehat{v}\} \in \mathsf{Mark}_1(\{(i, j), (i', j')\})$), but it crosses different cells.

Thus, P_2 has no bad edge that does not belong to P, and P has at least one bad edge that does not belong to P_2 (specifically, $\{u, v\}$), and therefore P_2 has fewer bad edges than P. Moreover, notice that the items above also imply that P_2 has at most one edge that crosses different cells and does not belong to P (specifically, $\{\widehat{u}, \widehat{v}\}$), and P has at least one edge that crosses the *same* cells and does not belong to P_2 (specifically, $\{u, v\}$). Therefore, P_2 also has the property of P that for every two distinct cells $(\widetilde{i}, \widetilde{j})$ and $(\widetilde{i}', \widetilde{j}')$, there exist at most 5 edges $\{\widetilde{u}, \widetilde{v}\} \in E(P_2)$ such that $f(\widetilde{u}) = (\widetilde{i}, \widetilde{j})$ and $f(\widetilde{v}) = (\widetilde{i}', \widetilde{j}')$. Therefore, we have reached a contradiction to the minimality of the number of bad edges in our choice of P.

In both cases we have reached a contradiction, and therefore the proof is complete. ◀

Next, we further strengthen Lemma 8 with the following definition and Lemma 13. Intuitively, the following definition says that a cell is good with respect to some path if either none of its unmarked vertices is traversed by that path, or all of its unmarked vertices are traversed by that path consecutively and can be "flanked" only by marked vertices (see Figure 4).

▶ **Definition 10.** *Let G be a clique-grid with representation f. Let P be a path (resp. cycle) in G with endpoints x, y (resp. no endpoints). We say that a cell $(i, j) \in [t] \times [t]$ is* good *if* **(i)** $V(P) = f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)$*, or* **(ii)** $V(P) \cap (f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)) = \emptyset$*, or* **(iii)** *there exist distinct $u, v \in (V(P) \cap \mathsf{Mark}^\star(i, j)) \cup (\{x, y\} \cap f^{-1}(i, j))$ (resp. not necessarily distinct $u, v \in V(P) \cap \mathsf{Mark}^\star(i, j)$) such that the set I of internal vertices of the (resp. a) subpath of P between u and v is precisely $f^{-1}(i, j) \setminus (\mathsf{Mark}^\star(i, j) \cup \{u, v\})$;*[3] *otherwise, it is* bad.

It will be convenient to have, as an intermediate step, a definition and lemma that are weaker than Definition 10 and Lemma 13. Intuitively, this definition drops that requirement that none or all the unmarked vertices of a cell should be visited by the path at hand, but only requires that those unmarked vertices that are visited, are visited consecutively and can be "flanked" only by marked vertices (see Figure 5).

▶ **Definition 11.** *Let G be a clique-grid with representation f. Let P be a path (resp. cycle) in G with endpoints x, y (resp. no endpoints). We say that a cell $(i, j) \in [t] \times [t]$ is* nice *if* **(i)** $V(P) \subseteq f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)$*, or* **(ii)** $V(P) \cap (f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)) = \emptyset$*, or* **(iii)** *there exist distinct $u, v \in (V(P) \cap \mathsf{Mark}^\star(i, j)) \cup (\{x, y\} \cap f^{-1}(i, j))$ (resp. not necessarily distinct*

[3] In other words, $I \subseteq f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)$ and $(f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)) \setminus I$ can only include endpoints of this subpath, in which case P is a path and any included endpoint is an endpoint of P as well.

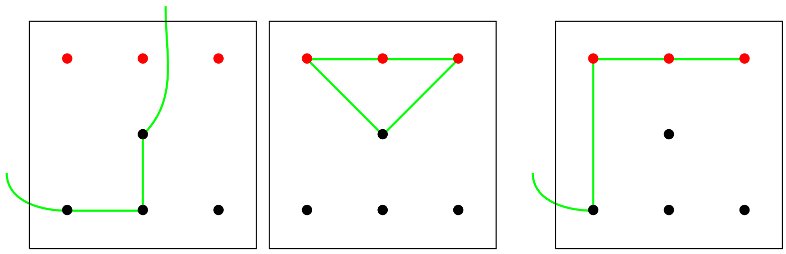


Figure 4 Illustration of good cells. The vertices colored black and red are marked and unmarked vertices, respectively. The green curve represent the path/cycle P.

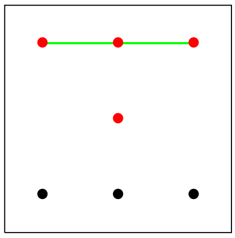


Figure 5 Illustration of a nice cell which is not good. The vertices colored black and red are marked and unmarked vertices, respectively. The green curve represents the path P.

$u, v \in V(P) \cap \mathsf{Mark}^\star(i,j)$) such that the set of internal vertices of the (resp. a) subpath of P between u and v is precisely $(V(P) \cap f^{-1}(i,j)) \setminus (\mathsf{Mark}^\star(i,j) \cup \{u,v\})$.

▶ **Lemma 12.** *Let G be a clique-grid with representation f that has a path (resp. cycle) on at least k vertices. Then, G also has a path (resp. cycle) P on at least k vertices with the following property: every cell $(i,j) \in [t] \times [t]$ is nice.*

Proof. Given a path (resp. cycle) P with endpoints x, y (resp. no endpoints) and a cell $(i,j) \in [t] \times [t]$, we say that a subpath of P is (i,j)-*nice* if there exist distinct $u, v \in (V(P) \cap \mathsf{Mark}^\star(i,j)) \cup (\{x,y\} \cap f^{-1}(i,j))$ (resp. $u, v \in V(P) \cap \mathsf{Mark}^\star(i,j)$) such that the set of internal vertices of the (resp. a) subpath of P between u and v is a subset I of $f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)$ such that if this subset I is empty, then the subpath has an endpoint in $f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)$ (which implies that P is a path and $\{u,v\} \cap \{x,y\} \cap (f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)) \neq \emptyset$); we further say that a subpath of P is *nice* if it is (i,j)-nice for some (i,j). By Lemma 8, G has a path (resp. cycle) on at least k vertices with the following property: every edge $\{u,v\}$ of that path where $f(u) \neq f(v)$ is good. Among all such paths (resp. cycles), let P be one with minimum number of nice subpaths, and let x, y be its endpoints (resp. no endpoints). (Notice that if x is unmarked, then because every edge $\{u,v\}$ of P where $f(u) \neq f(v)$ is good, it must be that x is an endpoint of a nice subpath. The same holds for y.) We next show that for every cell $(i,j) \in [t] \times [t]$, P has at most one nice (i,j)-subpath. Because either $V(P) \subseteq f^{-1}(i,j)$ or every vertex in $(V(P) \cap f^{-1}(i,j)) \setminus (\mathsf{Mark}^\star(i,j) \cup \{x,y\})$ (resp. $(V(P) \cap f^{-1}(i,j)) \setminus \mathsf{Mark}^\star(i,j)$) must be an internal vertex of a nice subpath (since every edge $\{u,v\}$ of P where $f(u) \neq f(v)$ is good), this would imply that every cell $(i,j) \in [t] \times [t]$ is nice, which will complete the proof. Targeting a contradiction, suppose that P yields some cell (i,j) such that there exist two distinct subpaths Q, Q' of P that are (i,j)-nice (see Figure 6), that is, each of them has both endpoints in $\mathsf{Mark}^\star(i,j) \cup (\{x,y\} \cap f^{-1}(i,j))$ (resp. $\mathsf{Mark}^\star(i,j)$) and the set

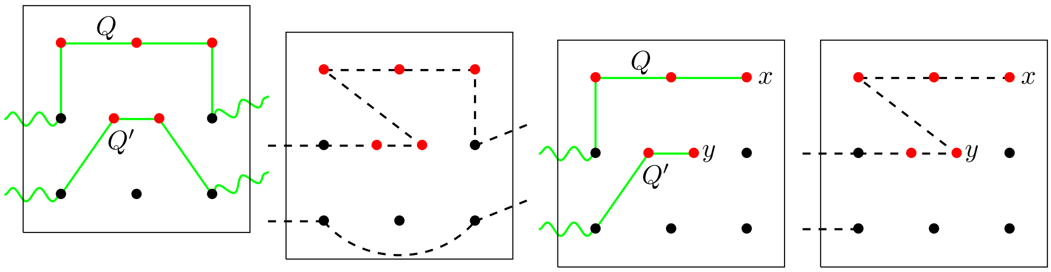


Figure 6 Illustration of the proof of Lemma 12. The vertices colored black and red are the marked and unmarked vertices in the cell, respectively. In the first figure the union of internal vertices of Q and Q' is the set of unmarked vertices in the cell, and the second figure depicts how to reroute to make the cell nice. The third figure illustrate the case when both the endpoints x and y of the path P are in the cell, and the fourth figure depicts how to reroute to make the cell nice.

of its internal vertices is a subset of $f^{-1}(i,j) \setminus \mathsf{Mark}^{\star}(i,j)$ that is either non-empty or some endpoint belongs to $\{x,y\} \cap (f^{-1}(i,j) \setminus \mathsf{Mark}^{\star}(i,j))$.

Note that if Q and Q' intersect, then they intersect only at their endpoints. Define $\widehat{P}$ by removing from P all the internal vertices of Q' as well as its endpoint in $f^{-1}(i,j) \setminus \mathsf{Mark}^{\star}(i,j)$ if such an endpoint exists (in which case P is a path and this endpoint it is also an endpoint of P), and inserting them arbitrarily between the vertices of Q (where multiple vertices can be inserted between two vertices); see Figure 6. By Property 1 in Definition 2, we have that $\widehat{P}$ is also a path (resp. cycle). Clearly, $|V(\widehat{P})| = |V(P)| \geq k$, and it is also directly implied by the construction that $\widehat{P}$ also has the property that every edge $\{u,v\} \in E(\widehat{P})$ where $f(u) \neq f(v)$ is good (since we did not make any change with respect to the set of edges that cross different cells). Notice that each subpath that is nice with respect to $\widehat{P}$ is either the subpath obtained by merging Q and Q' or a subpath that also exists in P and is therefore also a nice subpath with respect to P. Therefore, $\widehat{P}$ has one less nice subpath than P, which contradicts the minimality of P. ◀

We now state the main lemma of this section, whose proof is relegated to Appendix B.

▶ **Lemma 13.** *Let G be a clique-grid with representation f that has a path (resp. cycle) on at least k vertices. Then, G also has a path (resp. cycle) P on at least k vertices with the following property: every cell $(i,j) \in [t] \times [t]$ is good.*

4 The Algorithm

Our algorithm is based on a reduction of Long Path (resp. Long Cycle) on unit disk graphs to the weighted version of the problem, called Weighted Long Path (resp. Weighted Long Cycle), on unit disk graphs of treewidth $\mathcal{O}(\sqrt{k})$. In Weighted Long Path (resp. Weighted Long Cycle), we are given a graph G with a weight function $w : V(G) \to \mathbb{N}$ and an integer $k \in \mathbb{N}$, and the objective is to determine whether G has a path (resp. cycle) whose weight, defined as the sum of the weights of its vertices, is at least k.

The following proposition will be immediately used in our algorithm.

▶ **Proposition 14** ([10, 28]). Weighted Long Path *and* Weighted Long Cycle *are solvable in time $2^{\mathcal{O}(\mathtt{tw})}n$ where $\mathtt{tw}$ is the treewidth of the input graph.*

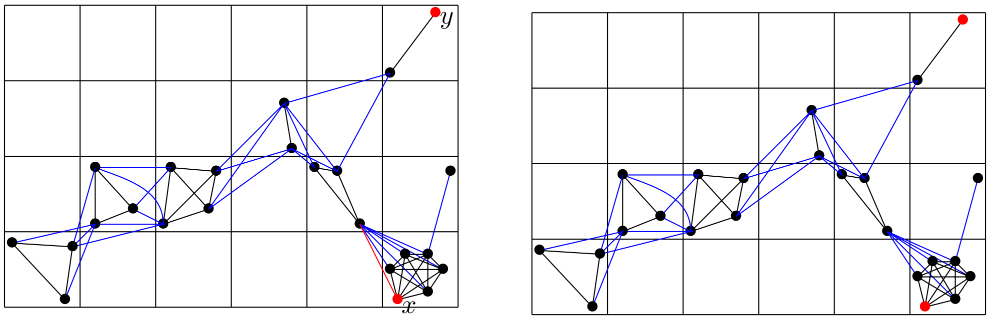


Figure 7 The graphs G' and $G^\star$ constructed from the graph G in Figure 1 are depicted on the left side and right side figures, respectively. Here, $w(x) = 2$, $w(y) = 3$, and for all $z \in V(G') \setminus \{x, y\}$, $w(z) = 1$.

Algorithm Specification. We call our algorithm ALG. Given an instance (G, k) of Long Path (resp. Long Cycle) on unit disk graphs, it works as follows.

1. Use Proposition 3 to obtain a representation $f : V(G) \to [t] \times [t]$ of G.
2. Use Observation 3.1 to compute $\mathsf{Mark}^\star(i, j)$ for every cell $(i, j) \in [t] \times [t]$. Let $\mathsf{Mark}^\star = \bigcup_{(i,j)\in[t]\times[t]} \mathsf{Mark}^\star(i, j)$.
3. Let G' be the graph defined as follows (see Figure 7). For any cell $(i, j) \in [t] \times [t]$, let $c_{(i,j)}$ denote a vertex in $f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)$ (chosen arbitrarily), where if no such vertex exists, let $c_{(i,j)} = \mathtt{nil}$. Then, $V(G') = \mathsf{Mark}^\star \cup (\{c_{(i,j)} : (i, j) \in [t] \times [t]\} \setminus \{\mathtt{nil}\})$ and $E(G') = E(G[V(G')])$. Because G' is an induced subgraph of G, it is a unit disk graph.
4. Define $w : V(G') \to \mathbb{N}$ as follows. For every $v \in V(G')$, if $v = c_{(i,j)}$ for some $(i, j) \in [t]\times[t]$ then $w(v) = |f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)|$, and otherwise $w(v) = 1$.
5. Let $G^\star$ be the graph defined as follows (see Figure 7): $V(G^\star) = V(G')$ and $E(G^\star) = E(G') \setminus \{\{c_{(i,j)}, v\} \in E(G') : (i, j) \in [t] \times [t], v \notin f^{-1}(i, j)\}$.
6. Let Δ be the maximum degree of $G^\star$. Use Proposition 4 to decide either $\mathtt{tw}(G^\star) > 100\Delta^3\sqrt{2k}$ or $\mathtt{tw}(G^\star) \leq 500\Delta^3\sqrt{2k}$.
7. If it was decided that $\mathtt{tw}(G^\star) > 100\Delta^3\sqrt{2k}$, then return Yes and terminate.
8. Use Proposition 14 to determine whether $(G^\star, w, k)$ is a Yes-instance of Weighted Long Path (resp. Weighted Long Cycle). If the answer is positive, then return Yes, and otherwise return No.

Analysis. We first analyze the running time of the algorithm.

▶ **Lemma 15.** *The time complexity of ALG is upper bounded by $2^{\mathcal{O}(\sqrt{k})}(n + m)$.*

Proof. By Proposition 3 and Observation 3.1, Steps 1 and 2 are performed in time $\mathcal{O}(n+m)$. By the definition of G', w and $G^\star$, they can clearly be computed in time $\mathcal{O}(n + m)$ as well (Steps 3, 4 and 5). Moreover, Step 7 is done in time $\mathcal{O}(1)$. By Proposition 4, Step 6 is performed in time $2^{\mathcal{O}(100\Delta^3\sqrt{2k})}n = 2^{\mathcal{O}(\Delta^3\sqrt{k})}n$. Thus, because we reach Step 8 only if we do not terminate in Step 7, we have that by Proposition 14, Step 8 is performed in time $2^{\mathcal{O}(\mathtt{tw}(G^\star))}n = 2^{\mathcal{O}(500\Delta^3\sqrt{2k})} = 2^{\mathcal{O}(\Delta^3\sqrt{k})}n$.

Thus, to conclude the proof, it remains to show that $\Delta = \mathcal{O}(1)$. Let Δ' be the maximum degree of G'. Since $G^\star$ is a subgraph of G', $\Delta^\star \leq \Delta'$. Thus, to prove $\Delta = \mathcal{O}(1)$, it is enough to prove that $\Delta' = \mathcal{O}(1)$. To this end, let $M = \max_{(i,j)\in[t]\times[t]} |(f^{-1}(i, j) \cap V(G')) \cup (\{c_{(i,j)}\} \setminus \{\mathtt{nil}\})|$. Since G' is a clique-grid, by Property 2 in Definition 2, we have that

$\Delta' \leq M^{25}$, hence it suffices to show that $M = \mathcal{O}(1)$. The definition of G' yields that $M \leq \max_{(i,j)\in[t]\times[t]} |\mathsf{Mark}^\star(i,j)|+1$. By Observation 3.1, $\max_{(i,j)\in[t]\times[t]} |\mathsf{Mark}^\star(i,j)| = \mathcal{O}(1)$, and therefore indeed $M = \mathcal{O}(1)$. ◀

Finally, we prove that the algorithm is correct.

▶ **Lemma 16.** *ALG solves* LONG PATH *and* LONG CYCLE *on unit disk graphs correctly.*

Proof. Let (G,k) be an instance of LONG PATH or LONG CYCLE on unit disk graphs. By the specification of the algorithm, to prove that it solves (G,k) correctly, it suffices to prove that the two following conditions are satisfied.

1. If $\mathtt{tw}(G^\star) > 100\Delta^3\sqrt{2k}$, then (G,k) is a Yes-instance of LONG PATH and LONG CYCLE.
2. (G,k) is a Yes-instance of LONG PATH (resp. LONG CYCLE) if and only if $(G^\star,w,k)$ is a Yes-instance of WEIGHTED LONG PATH (resp. WEIGHTED LONG CYCLE).

The proof of satisfaction of the first condition is simple and can be found in Appendix C.

Now, we turn to prove the second condition. In one direction, suppose that (G,k) is a Yes-instance of LONG PATH (resp. LONG CYCLE). Then, by Lemma 13, G has a path (resp. cycle) P on at least k vertices with the following property: every cell $(i,j) \in [t]\times[t]$ is good. Notice that every maximal subpath Q of P that consists only of unmarked vertices satisfies *(i)* $V(Q) = f^{-1}(i_Q,j_Q) \setminus \mathsf{Mark}^\star(i_Q,j_Q)$ for some cell $(i_Q,j_Q) \in [t]\times[t]$, and *(ii)* the endpoints of Q are adjacent in P to vertices in $f^{-1}(i_Q,j_Q)$ (unless $Q = P$). Obtain $P^\star$ from P as follows: every maximal subpath Q of P that consists only of unmarked vertices is replaced by $c_{(i_Q,j_Q)}$. (Notice that $c_{(i_Q,j_Q)} \neq \mathtt{nil}$ because $V(Q) \neq \emptyset$.) Because of Property *(ii)* above and Property 1 in Definition 2, we immediately have that $P^\star$ is a path (resp. cycle) in $G^\star$. Moreover, by Property *(i)* above and the definition of the weight function w (in Step 4), each subpath Q is replaced by a vertex $c_{(i_Q,j_Q)}$ whose weight equals $|V(Q)|$. Because $|V(P)| \geq k$, we have that $P^\star$ is a path (resp. cycle) of weight at least k in $G^\star$. Thus, $(G^\star,w,k)$ is a Yes-instance of WEIGHTED LONG PATH (resp. WEIGHTED LONG CYCLE).

In the other direction, suppose that $(G^\star,w,k)$ is a Yes-instance of WEIGHTED LONG PATH (resp. WEIGHTED LONG CYCLE). Then, $G^\star$ has a path (resp. cycle) $P^\star$ of weight at least k. Obtain P from $P^\star$ by replacing each vertex of the form $c_{(i,j)} \in V(P)$ for some $(i,j) \in [t]\times[t]$ by a path Q whose vertex set is $f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)$ (the precise ordering of the vertices on this path is arbitrary). Notice that because all edges in $\{\{c_{(i,j)},v\} \in E(G') : (i,j) \in [t]\times[t], v \notin f^{-1}(i,j)\}$ were removed from G' to derive $G^\star$, each vertex of the form $c_{(i,j)} \in V(P)$ for some $(i,j) \in [t]\times[t]$ is adjacent in $P^\star$ only to vertices in $\mathsf{Mark}^\star(i,j)$. Therefore, by Property 1 in Definition 2, we have that P is a path (resp. cycle) in G. Moreover, by the definition of the weight function w (in Step 4), each vertex $c_{(i,j)}$ was replaced by $w(c_{(i,j)})$ vertices. Because the weight of $P^\star$ is at least k, we have that P is a path (resp. cycle) on at least k vertices in G. Thus, (G,k) is a Yes-instance of LONG PATH (resp. LONG CYCLE). ◀

Thus, Theorem 1 follows from Lemmas 15 and 16.

References

1 Jochen Alber and Jiří Fiala. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. In *Foundations of Information Technology in the Era of Network and Mobile Computing*, pages 26–37. Springer, 2002.

2 Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and Süleyman Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. In *Proceedings 16th International Conference on Intelligent Systems for Molecular Biology (ISMB), Toronto, Canada, July 19-23, 2008*, pages 241–249, 2008. URL: `https://doi.org/10.1093/bioinformatics/btn163`, `doi:10.1093/bioinformatics/btn163`.

3 Noga Alon and Shai Gutner. Balanced hashing, color coding and approximate counting. In *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, pages 1–16, 2009. URL: `https://doi.org/10.1007/978-3-642-11269-0_1`, `doi:10.1007/978-3-642-11269-0_1`.

4 Noga Alon and Shai Gutner. Balanced families of perfect hash functions and their applications. *ACM Trans. Algorithms*, 6(3):54:1–54:12, 2010. URL: `http://doi.acm.org/10.1145/1798596.1798607`, `doi:10.1145/1798596.1798607`.

5 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. Assoc. Comput. Mach.*, 42(4):844–856, 1995.

6 Vikraman Arvind and Venkatesh Raman. Approximation algorithms for some parameterized counting problems. In *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings*, pages 453–464, 2002. URL: `https://doi.org/10.1007/3-540-36136-7_40`, `doi:10.1007/3-540-36136-7_40`.

7 Ivona Bezáková, Radu Curticapean, Holger Dell, and Fedor V. Fomin. Finding detours is fixed-parameter tractable. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 54:1–54:14, 2017.

8 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.*, 87:119–139, 2017.

9 Andreas Björklund, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Approximate counting of k-paths: Deterministic and in polynomial space. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, pages 24:1–24:15, 2019.

10 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. URL: `https://doi.org/10.1016/j.ic.2014.12.008`, `doi:10.1016/j.ic.2014.12.008`.

11 Hans L Bodlaender, Rodney G Downey, Michael R Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.

12 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016.

13 Cornelius Brand, Holger Dell, and Thore Husfeldt. Extensor-coding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 151–164, 2018. URL: `http://doi.acm.org/10.1145/3188745.3188902`, `doi:10.1145/3188745.3188902`.

14 Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46(2):178–189, 2003. URL: `http://dx.doi.org/10.1016/S0196-6774(02)00294-8`, `doi:10.1016/S0196-6774(02)00294-8`.

15 Jianer Chen, Joachim Kneis, Songjian Lu, Daniel Mölle, Stefan Richter, Peter Rossmanith, Sing-Hoi Sze, and Fenghui Zhang. Randomized divide-and-conquer: Improved path, matching, and packing algorithms. *SIAM Journal on Computing*, 38(6):2526–2547, 2009.

16 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.

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

18 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159, 2011.

19 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.

20 Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. *Journal of the ACM*, 52(6):866–893, 2005.

21 Erik D. Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51(3):292–302, 2008.

22 Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010. URL: `http://dx.doi.org/10.1007/s00453-009-9296-1`, `doi:10.1007/s00453-009-9296-1`.

23 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

24 Adrian Dumitrescu and János Pach. Minimum clique partition in unit disk graphs. *Graphs and Combinatorics*, 27(3):399–411, 2011.

25 Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, 2004.

26 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Going far from degeneracy. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, pages 47:1–47:14, 2019.

27 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Parameterization above a multiplicative guarantee. In *11th Innovations in Theoretical Computer Science, ITCS 2020 (To Appear)*, 2020.

28 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016.

29 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *Journal of the ACM*, 63(4):29, 2016. URL: `http://doi.acm.org/10.1145/2886094`, `doi:10.1145/2886094`.

30 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Long directed (s, t)-path: FPT algorithm. *Inf. Process. Lett.*, 140:8–12, 2018.

31 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Decomposition of map graphs with applications. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 60:1–60:15, 2019.

32 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Finding, hitting and packing cycles in subexponential time on unit disk graphs. *Discrete & Computational Geometry*, 62(4):879–911, 2019.

33 Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Bidimensionality and geometric graphs. pages 1563–1575. SIAM, 2012.

34 Harold N Gabow and Shuxin Nie. Finding a long directed cycle. *ACM Transactions on Algorithms (TALG)*, 4(1):7, 2008.

35 William K Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

36 Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (turing) kernelization. *Algorithmica*, 71(3):702–730, 2015. URL: `https://doi.org/10.1007/s00453-014-9910-8`, `doi:10.1007/s00453-014-9910-8`.

37 Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm engineering for color-coding with applications to signaling pathway detection. *Algorithmica*, 52(2):114–132, 2008.

38 Harry B. Hunt III, Madhav V. Marathe, Venkatesh Radhakrishnan, S. S. Ravi, Daniel J. Rosenkrantz, and Richard Edwin Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms*, 26(2):238–274, 1998.

39 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

40 Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11(4):676–686, 1982. URL: `https://doi.org/10.1137/0211056`, `doi:10.1137/0211056`.

41 Hiro Ito and Masakazu Kadoshita. Tractability and intractability of problems on unit disk graphs parameterized by domain area. In *Proceedings of the 9th International Symposium on Operations Research and Its Applications (ISORA10)*, pages 120–127, 2010.

42 Bart M. P. Jansen. Polynomial kernels for hard problems on disk graphs. In *Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 6139, pages 310–321. Springer, 2010.

43 Bart M. P. Jansen, László Kozma, and Jesper Nederlof. Hamiltonicity below dirac's condition. In *Graph-Theoretic Concepts in Computer Science - 45th International Workshop, WG 2019, Vall de Núria, Spain, June 19-21, 2019, Revised Papers*, pages 27–39, 2019.

44 Bart M. P. Jansen, Marcin Pilipczuk, and Marcin Wrochna. Turing kernelization for finding long paths in graph classes excluding a topological minor. *Algorithmica*, 81(10):3936–3967, 2019.

45 Karl Kammerlander. C 900-an advanced mobile radio telephone system with optimum frequency utilization. *IEEE journal on selected areas in communications*, 2(4):589–597, 1984.

46 Ioannis Koutis. Faster algebraic algorithms for path and packing problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, volume 5125 of *Lecture Notes in Computer Science*, pages 575–586, 2008.

47 Ioannis Koutis and Ryan Williams. Algebraic fingerprints for faster algorithms. *Commun. ACM*, 59(1):98–105, 2016. URL: `http://doi.acm.org/10.1145/2742544`, `doi:10.1145/2742544`.

48 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. *SIAM J. Comput.*, 47(3):675–702, 2018.

49 Burkhard Monien. How to find long paths efficiently. In *North-Holland Mathematics Studies*, volume 109, pages 239–254. Elsevier, 1985.

50 George L Nemhauser and Leslie Earl Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

51 Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Contraction decomposition in unit disk graphs and algorithmic applications in parameterized complexity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1035–1054, 2019.

52 Christos H. Papadimitriou and Mihalis Yannakakis. On limited nondeterminism and the complexity of the V.C dimension (extended abstract). In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 12–18, 1993.

53 Hadas Shachnai and Meirav Zehavi. Representative families: A unified tradeoff-based approach. *J. Comput. Syst. Sci.*, 82(3):488–502, 2016.

54 Warren D. Smith and Nicholas C. Wormald. Geometric separator theorems & applications. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 232–243. IEEE Computer Society, 1998.

55 Dekel Tsur. Faster deterministic parameterized algorithm for k-path. *Theor. Comput. Sci.*, 790:96–104, 2019.

56 DW Wang and Yue-Sun Kuo. A study on two geometric location problems. *Information processing letters*, 28(6):281–286, 1988.

57 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009.

58 Yu-Shuan Yeh, J Wilson, and S Schwartz. Outage probability in mobile telephony with directive antennas and macrodiversity. *IEEE journal on selected areas in communications*, 2(4):507–511, 1984.

59 Meirav Zehavi. Mixing color coding-related techniques. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 1037–1049, 2015. URL: `http://dx.doi.org/10.1007/978-3-662-48350-3_86`, `doi:10.1007/978-3-662-48350-3_86`.

60 Meirav Zehavi. A randomized algorithm for long directed cycle. *Inf. Process. Lett.*, 116(6):419–422, 2016.

A Preliminaries (Cont.)

For a graph G, let $V(G)$ and $E(G)$ denote its vertex set and edge set, respectively. When G is clear from context, let $n = |V(G)|$ and $m = |E(G)|$. For a subset $U \subseteq V(G)$, let $G[U]$ denote the subgraph of G induced by U. A graph H is a *minor* of G if H can be obtained from G by a sequence of edge deletions, edge contractions and vertex deletions. Given $a, b \in \mathbb{N}$, an $a \times b$-*grid* is a graph on $a \cdot b$ vertices that can be denoted by $v_{i,j}$ for $(i, j) \in [a] \times [b]$, such that $E(G) = \{\{v_{i,j}, v_{i+1,j}\} : i \in [a-1], j \in [b]\} \cup \{\{v_{i,j}, v_{i,j+1}\} : i \in [a], j \in [b-1]\}$.

▶ **Definition 17 (Treewidth).** *A* tree decomposition *of a graph G is a pair (T, β), where T is a tree and β is a function from $V(T)$ to $2^{V(G)}$, that satisfies the following conditions.*

- *For every edge $\{u, v\} \in E(G)$, there exists $x \in V(T)$ such that $\{u, v\} \subseteq \beta(x)$.*
- *For every vertex $v \in V(G)$, $T[\{x \in V(T)\}]$ is a tree on at least one vertex.*

The width *of (T, β) is $\max_{x \in V(T)} |\beta(x)| - 1$. The* treewidth *of G, denoted by $\mathtt{tw}(G)$, is the minimum width over all tree decompositions of G.*

B Proof of Lemma 13

Proof. By Lemma 12, G has a path (resp. cycle) P on at least k vertices with the following property: every cell $(i, j) \in [t] \times [t]$ is nice. Among all such paths (resp. cycles), let P be one with minimum number of bad cells. Next, we show that P yields no bad cell, which will complete the proof. Targeting a contradiction, suppose that P yields some bad cell $(i, j) \in [t] \times [t]$. Because this cell is not good, $(V(P) \cap f^{-1}(i, j)) \setminus \mathsf{Mark}^\star(i, j) \neq \emptyset$. Further, because (i, j) is nice, either $V(P) \subseteq f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)$ or there exist distinct $u, v \in (V(P) \cap \mathsf{Mark}^\star(i, j)) \cup (\{x, y\}) \cap f^{-1}(i, j))$ (resp. $u, v \in V(P) \cap \mathsf{Mark}^\star(i, j)$) such that the set of internal vertices of the (resp. a) subpath Q of P between u and v is precisely $(V(P) \cap f^{-1}(i, j)) \setminus (\mathsf{Mark}^\star(i, j) \cup \{u, v\})$ (see Figure 8). In the first case, notice that since $f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)$ induces a clique (by Property 1 in Definition 2) and its size is at least k (because $|V(P)| \geq k$), it is clear that G contains a path (resp. cycle) whose vertex set is $f^{-1}(i, j) \setminus \mathsf{Mark}^\star(i, j)$ and which has at least k vertices, for which every cell is trivially good. Thus, we next suppose that only the second case happens.

Notice that Q must contain a vertex from $\mathsf{Mark}^\star(i, j)$ as an endpoint, because its endpoints $u, v \in (V(P) \cap \mathsf{Mark}^\star(i, j)) \cup (\{x, y\}) \cap f^{-1}(i, j))$ (resp. $u, v \in V(P) \cap \mathsf{Mark}^\star(i, j)$) and it is not possible that $\{u, v\} = \{x, y\}$ (since then the first case happens). Because also

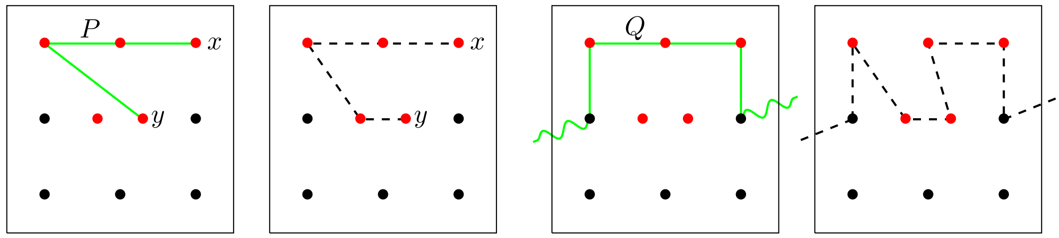


Figure 8 Illustration of the proof of Lemma 13. The vertices colored black and red are marked and unmarked vertices, respectively. In the first figure $V(P) \subseteq f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)$, and the second figure illustrates that there is a path of length at least $|V(P)|$ whose vertex set is the set of unmarked vertices in the cell. The third figure illustrates the case where P is not fully contained the cell, and the fourth figure depicts a possibility to reroute it to make the cell good.

$(V(P) \cap f^{-1}(i,j)) \setminus \mathsf{Mark}^\star(i,j) \neq \emptyset$, we know that Q contains one edge $\{a,b\}$ with both endpoints from $f^{-1}(i,j)$. Then, we derive $\widehat{P}$ from P by inserting all the vertices in $(f^{-1}(i,j) \setminus \mathsf{Mark}^\star(i,j)) \setminus V(P)$ between a and b in some arbitrary order (see Figure 8). By Property 1 in Definition 2, we still have a path (resp. cycle). Further, notice that (i,j) is a good cell with respect to $\widehat{P}$. As the adjacencies of all vertices outside the cell (i,j) are the same in P and $\widehat{P}$, we have that $\widehat{P}$ has only nice cells (because P has this property), and that every cell that is bad with respect to $\widehat{P}$ is also bad with respect to P. Thus, we obtain a path (resp. cycle) on at least k vertices with fewer bad cells than P and still with the property every cell $(i,j) \in [t] \times [t]$ is nice. This is a contradiction to the choice of P, and therefore the proof is complete. ◀

C Satisfaction of the First Condition in the Proof of Lemma 16

Proof. For the proof of satisfaction of the first condition, suppose that $\mathtt{tw}(G^\star) > 100\Delta^3\sqrt{2k}$. Then, by Proposition 5, $G^\star$ contains a $\sqrt{2k} \times \sqrt{2k}$-grid as a minor. Clearly, a $\sqrt{2k} \times \sqrt{2k}$-grid contains a cycle (and hence also a path) on k vertices. By the definition of minor, this means that G contains cycle (and hence also a path) on at least k vertices, and therefore (G,k) is a Yes-instance of Long Path and Long Cycle. ◀