Independent Sets in Line of Sight Networks

Pavan Sangha, Michele Zito

Department of Computer Science, University of Liverpool, United Kingdom
{p.sangha2,michele}@liverpool.ac.uk

March 22, 2019

Abstract

Line of Sight (LoS) networks provide a model of wireless communication which incorporates visibility constraints. Vertices of such networks can be embedded onto the cube $\{(x_1, x_2, ..., x_d) : x_i \in \{1, ..., n\}, 1 \leq i \leq d\}$ so that two vertices are adjacent if and only if their images lay on a line parallel to one of the cube edges and their distance is less than a given range parameter ω. In this paper we study large independent sets in LoS networks. We prove that the computational problem of finding a maximum independent set can be solved optimally in polynomial time for one dimensional LoS networks. However, for $d \geq 2$, the (decision version of) the problem becomes NP-complete for any fixed $\omega \geq 3$. In addition, we show that the problem is APX-hard when $\omega = n$ for $d \geq 3$. On the positive side, we show that LoS networks generalise chordal graphs. This implies that there exists a simple d-approximation algorithm for the maximum independent set problem in LoS networks. Finally, we describe a polynomial time approximation scheme for the maximum independent set problem in LoS networks for the case when ω is a constant and present an improved heuristic algorithm for the problem in the case $\omega = n$.

1 Introduction

Geometric graphs have become a popular tool for reasoning about wireless networks. Typically, wireless devices positioned in some physical space can be represented by a collection of vertices. A graph can then be constructed by representing communication between pairs of vertices by edges. The disk intersection model is one of the commonly used model for representing wireless sensor networks [4]. Sensors are modelled as vertices in some topological setting and their communication ranges are represented by circles having some prescribed radius. Overlapping circles then represent communication between pairs of vertices and makes it possible to construct a graph. Unfortunately in many real world applications of wireless networks, the environments often come with a large number of obstacles which impose line of sight constraints on vertices. These obstacles are difficult to incorporate in the geometric model described above.

Frieze et al. [11] introduced the notion of a (random) 2-dimensional Line of Sight (LoS) networks as a model of wireless networks which can incorporate visibility constraints, and they then studied connectivity problems in this setting. Since then connectivity in higher dimensions, percolation and communication problems have been analyzed [2, 6, 7] in the same model. In this paper we work with the following variation of Frieze's model. For positive integers d and n, let $\mathbb{Z}_n^d$ denote the *cube* $\{(x_1, x_2, ..., x_d) : x_i \in \{1, ..., n\}, 1 \leq i \leq d\}$. Parameter d will be occasionally referred to as the set *dimension*, whereas, with a slight *abus de langage*, we will say that n is the *size* of the given cube. In the rest of the paper, unless otherwise stated, the distance between points $\boldsymbol{x} = (x_1, x_2, ..., x_d)$ and $\boldsymbol{x}' = (x'_1, x'_2, ..., x'_d)$ in $\mathbb{Z}_n^d$, denoted by $\text{dist}(\boldsymbol{x}, \boldsymbol{x}')$, will be the sum $\sum_{i=1}^{d} |x_i - x'_i|$ (also known as the *Manhattan distance*). Note that this distance is not "wrapped around" as it was in [11] where a torus was used to simplify calculations. We say that two distinct points $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ and $\boldsymbol{x}' = (x'_1, x'_2, \ldots, x'_d)$ in $\mathbb{Z}_n^d$ *share a line of sight* if there exists $j \in \{1, \ldots, d\}$ such that $x_i = x'_i$ for all $i \in \{1, \ldots, d\} \setminus \{j\}$, moreover in this case we say that $\boldsymbol{x}$ and $\boldsymbol{x}'$ *share a j-line*. Let ω be a positive integer with $1 \leq \omega \leq n$. A graph $G = (V, E)$ is said to be a *Line of Sight (LoS) network (with parameters d, n, and ω)* if there exists an embedding $f_{G,w} : V \to \mathbb{Z}_n^d$ such that $\{u, v\} \in E$ if and only if $f_{G,\omega}(u)$ and $f_{G,w}(v)$ share a line of sight and the distance between them is strictly less than ω. The set $\mathbb{Z}_n^d$, is the network *underlying cube*. Given a LoS network $G = (V, E)$, we denote by $f_{G,\omega}(V)$ the set of points in $\mathbb{Z}_n^d$ that are the images of vertices of G under the embedding $f_{G,\omega}$. We say that a LoS network G with parameters d, n and ω is *d-dimensionally spanning* if for each $j \in \{1, 2, \ldots, d\}$ there exists an edge $\{u, v\} \in E$ such that $f_{G,\omega}(u)$ and $f_{G,\omega}(v)$ share a j-line. Let $L_{n,\omega}^d$ denote the set of all graphs G which are d-dimensionally spanning LoS networks with parameters n, d and ω and let $L_\omega^d = \cup_{n \in \mathbb{N}} L_{n,\omega}^d$.

The parameter ω, called the network *range parameter*, is used to model the usual proximity constraint in a wireless network. The line of sight visibility constraint (two vertices u and v cannot be connected by an edge unless $f_{G,\omega}(u)$ and $f_{G,\omega}(v)$ share a line of sight) is the mechanism that allows modelling of obstacles. LoS networks generalise various types of graphs. For example, LoS networks with parameter $\omega = 2$ are (subgraphs of) *grid graphs* [5]: each vertex can only share an edge with any of the vertices present in the $2d$ locations at distance one in $\mathbb{Z}_n^d$. On the other hand, the graphs admitting a d dimensional embedding, for some d, with $\omega = n$ are known as *gridline graphs* [23]. As a final remark before moving to the description of the optimization problem which is the main subject of this paper, we stress that our main concern here is to understand to what extent the particular properties of the LoS networks help solving difficult combinatorial problems. Clearly, every LoS network G that is embeddable in $\mathbb{Z}_n^d$ is also embeddable in $\mathbb{Z}_n^{d'}$, for every $d' > d$ (although G might not have a d'-dimensionally spanning embedding) and, in general, G might even have several different embeddings in $\mathbb{Z}_n^d$. In what follows, when we will need to use any information about the embedding, we will assume that $f_{G,\omega}$ is an arbitrary fixed embedding of G. (For completeness it should be mentioned that, since we started our work on this model, others [19] have investigated the problem of deciding whether a graph admits a particular line of sight embedding.) Because of this, unless ambiguity arises, we might occationally abuse notation and denote by letters like u or v both the vertices of G

and the points that are images of these via the embedding $f_{G,\omega}$. We will also omit the subscripts G and ω to simplify notations.

So far LoS networks have been studied in a randomized setting. Each point in $\mathbb{Z}_n^d$ contains a node with probability p (so that the expected size of $V(G)$ is n^2p) and one is interested in connectivity thresholds and other structural properties. The purpose of our paper is to start the study of LoS networks combinatorial and algorithmic properties from the worst-case point of view. In what follows we focus on the well known maximum independent set problem (MIS). An *independent set* in a graph is a set of vertices which are pairwise non-adjacent. Let $\alpha(G)$ be the maximum size of an independent set in G. Given a graph $G = (V, E)$, MIS asks for an independent set in G of cardinality $\alpha(G)$. Large independent sets in graphs have been the subject of significant study in various branches of Mathematics as they provide a measure of network dispersion and have a strong relation to other important graph structures such as vertex covers, cliques and colourings [9]. It is well known that finding a largest independent set in a graph is an NP-hard problem [12], and even good approximate solutions are hard to find [13]. In this paper we argue that working with LoS networks makes the problem more easily approximable in many ways. Section 2 presents all our major results. The focus is on the statements and their consequences, rather than the technicalities of their proofs. The subsequent sections contain the details of the hardness results (Section 3 and 4) and the algorithmic ones (Section 5). In what follows, if Π is a computational problem and $\mathcal{I}$ is a particular set of instances for it, then $\Pi(\mathcal{I})$ will denote the restriction of Π to instances belonging to $\mathcal{I}$. Unless otherwise stated we follow [9] for all our graph-theoretic notations. Definitions and notations related to (approximation) algorithms can be found in [14] or the more recent [10].

2 Statement of Results

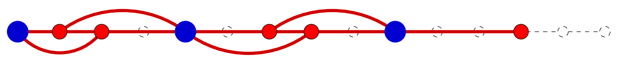

Figure 1: A 1-dimensional line of sight network ($n = 15$ and $\omega = 4$). The dashed part represents $\mathbb{Z}_{15}^1$. The big dark vertices form a maximum independent set.

One dimensional LoS networks (see Figure 1) are very simple objects indeed, for any ω. It is not difficult to see that the collection of (independent) vertices added greedily moving left to right is optimal (indeed the algorithm hinted to in the proof of Theorem 9 below returns such a set when applied to a one dimensional LoS network). At another extreme of the parameter space when $\omega \leq 2$ (for any d), the network is bipartite (Figure 2 describes a small example in three dimensions) and MIS is again easily solvable. One additional special case admits a polynomial time computable optimal solution. When $d = 2$ and $\omega = n$ all nodes mapped to the same row (column) of $\mathbb{Z}_n^2$ are connected by edges in G. In such case, as observed by Peterson [23], MIS reduces to the maximum matching problem in a bipartite graph having an edge $\{x_1, x_2\}$ for each point (x_1, x_2) in $\mathbb{Z}_n^2$ that belongs to $f(V)$.

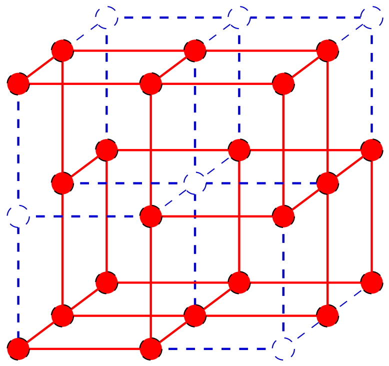
Figure 2: A small 3-dimensional line of sight network ($n = 3$ and $\omega = 2$). The dashed part represent points that are not in $f(V)$.

In higher dimensions ($d \geq 3$) or when $\omega \in \{3, \ldots, n-1\}$ the problem becomes more difficult and the overall picture is less clear cut. We prove the following (here IS is the decision version of MIS).

Theorem 1. IS(L^d_ω) *is* NP-*complete, for each fixed integer* $d \geq 2$ *and* $\omega \geq 3$.

The proof of Theorem 1 holds even for $\omega = \omega(n)$ at least if $\omega = O(n^{1-\epsilon})$ for any fixed $\epsilon \in (0, 1)$ (see the argument at the beginning of Section 4). However, it cannot be extended to cover the case of very large range parameters. A different reduction allows us to prove the following stronger result for a particular extreme case (here GRIDLINE^d is the class of all gridline graphs which admit a d-dimensional embedding, and, to avoid trivialities, are d-dimensionally spanning).

Theorem 2. MIS(GRIDLINE^d) *is* APX-*hard for each fixed integer* $d \geq 3$.

Thus, for $\omega = n$, we have a perfect dychotomy: MIS is solvable in polynomial time for $d = 2$, while it is APX-hard (and therefore NP-hard *a fortiori*) for $d > 2$.

In the final part of the paper we complement these negative results by studying the performance of various algorithmic strategies. First, we prove the following general result.

Theorem 3. *Let* d *be a positive integer. There is a polynomial time* d*-approximation algorithm for* MIS(L^d_ω) *for any positive integer* ω.

The result can be obtained by a direct analysis of a greedy algorithm based on a technique used by Marathe *et al.* [18] in the context of unit disk graphs. However, in Section 5, we will argue that in fact the proof of Theorem 3 hinges on two different structural properties of the LoS networks which may be of independent interest.

For small ranges, a second type of approximation result can also be proved.

Theorem 4. *Let* d *be a positive integer.* MIS(L^d_ω) *admits a polynomial time approximation scheme for any fixed positive integer* ω.

The algorithm used in Theorem 4 is in fact an *efficient polynomial time approximation scheme* (or EPTAS, *a la* [3]); its running time depends exponentially on the inverse of the sought approximation ratio as a function of ω, but it is fully polynomial in n, the size of the given grid, for fixed ω (see details in Section 5.2).

In the light of Theorem 2, unless P=NP, Theorem 4 cannot be extended, for $d \geq 3$, to cover the case $\omega = n$. However, the additional structural constraints on the gridline graphs enable us to prove the following improved approximation result.

Theorem 5. *Let*

$$\rho(d,t) = \frac{d - c/(d-1)^{\lfloor t/2 \rfloor}}{2 - c/(d-1)^{\lfloor t/2 \rfloor}}$$

where $c = d - (d-2)(1-(t \mod 2))$. *For each positive integers* d *and* t*, with* $d \geq 3$*, there is a* $\rho(d,t)$*-approximation algorithm for* MIS *which runs in time* $O(|V(G)|^{2t+1})$ *for any* $G \in$ GRIDLINE^d.

For completeness it is worth pointing out that our focus in this paper is on *robust* algorithms. This concept (introduced by Raghavan and Spinrad [24]) is somehow related to the possible hardness of the recognition problem for LoS networks. Informally, an algorithm is *robust on a certain graph class* $\mathcal{C}$ if it works as expected for every $G \in \mathcal{C}$ and for every other G either works or correctly certifies that $G \notin \mathcal{C}$. Approximation algorithms beating the results presented here have recently been proposed in [25], but those crucially exploit the given embedding and therefore are not robust.

3 Hardness for Short Ranges

In this section we prove Theorem 1. Membership in NP is obvious therefore we focus on the hardness result. For $d = 2$ (resp. $d \geq 3$) we describe explicit embeddings in $\mathbb{Z}_n^d$ of graphs which are subdivisions of planar graphs with maximum degree at most 4 (resp. of bounded degree graphs). We start by embedding a graph $G = (V, E)$ with the required degree bound orthogonally in $\mathbb{Z}_n^d$ (see Section 3.1). We then add further vertices to G to obtain the graph G' which is a d-dimensionally spanning LoS network. Once this is done, to complete our NP-hardness proof, we show the existence of a linear relationship between the size of an independent set in G and the size of an independent set in the resulting LoS network G'. The result follows as the independent set problem is NP-hard for both planar graphs of maximum degree four and bounded degree graphs.

3.1 Embeddings

Graph embedding has been an active research area for quite some time (the interested reader is referred to survey papers like [8, 20] for additional bibliographic details). Here we will be interested in particular embeddings of bounded degree graphs in $\mathbb{Z}_n^d$. Define a *path (in* $\mathbb{Z}_n^d$*)* to be any sequence of distinct points in $\mathbb{Z}_n^d$ such that any two consecutive points in the sequence have distance equal to one. If P is a path, $|P|$ will denote its length, the number of points in P minus one. An *orthogonal* embedding Γ_G of a graph $G = (V, E)$ in $\mathbb{Z}_n^d$ is an embedding where the vertices $v \in V$ are mapped to points in $\mathbb{Z}_n^d$ denoted $\Gamma_G(v)$ and the edges $\{u, v\} \in E$ to paths in $\mathbb{Z}_n^d$ with end-points $\Gamma_G(u)$ and $\Gamma_G(v)$, respectively. Paths representing distinct edges can only intersect at their end-points. An orthogonal embedding of G in $\mathbb{Z}_n^d$ requires $d \geq \lceil \Delta(G)/2 \rceil$ (since otherwise there is not enough directions to route all the edges out of each vertex of degree $\Delta(G)$) and, for general graphs, the bound is tight except for $\Delta(G) \leq 4$. In this work we will use the following two results, dealing with the case $d = 2$ and $d \geq 3$, respectively.

Theorem 6. [26] *Any planar graph $G = (V, E)$ with $\Delta(G) \leq 4$ and $|E| = m$ admits an orthogonal embedding in $\mathbb{Z}_{3m}^2$.*

Theorem 7. [28] *Any simple graph $G = (V, E)$ with $\Delta(G) \geq 5$ admits an orthogonal embedding in $\mathbb{Z}_{k|V|}^d$, where $d = \lceil \Delta(G)/2 \rceil$ and k is a positive integer constant.*

We point out that the embeddings in the theorems above can be constructed in polynomial time.

3.2 Full Reduction

Recall that we refer to a sequence of distinct points $P = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ as a path in $\mathbb{Z}_n^d$ if and only if $\boldsymbol{x}_i \in \mathbb{Z}_n^d$ and for all $i \in \{1, \ldots, k\}$ and $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) = 1$ for all $i \in \{1, \ldots, k-1\}$. We now describe how to transform certain graphs of bounded degree into LoS networks. In general, edges of the given graph G are independently processed one at a time. An example of the transformation applied to a single edge $\{u, v\}$ of G is described in Figure 3, while Figure 4 sketches a full example of the construction for $d = 2$.

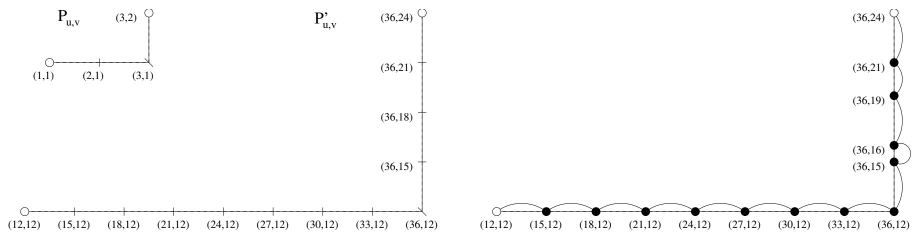

Figure 3: On the left we have the orthogonally embedded path P_{uv}, in the middle we have the stretched path P'_{uv} obtained by stretching P_{uv} by a factor $4(\omega - 1) = 12$. Finally on the right we construct a LoS path from P'_{uv} by placing an even number of sensor nodes on P'_{uv} in accordance with our method.

Depending on d, we use Theorem 6 or Theorem 7 to obtain an orthogonal embedding Γ_G of the appropriate bounded degree graph $G = (V, E)$ in $\mathbb{Z}_{k|V|}^d$, for some fixed k. We then obtain another, *stretched*, orthogonal embedding Γ'_G in $\mathbb{Z}_{4(\omega-1)\cdot k|V|}^d$, by defining $\Gamma'_G(u) = 4(\omega - 1) \cdot \Gamma_G(u)$ for each $u \in V$. Notice that we effectively stretch the length of each path P_{uv} in Γ_G by a factor $4(\omega-1)$ to obtain the corresponding path P'_{uv} in Γ'_G, hence

$$|P'_{uv}| = 4(\omega - 1) \cdot |P_{uv}|$$

for every $\{u, v\} \in E$. For any two points u' and v' on a path P_{uv}, we refer to the length of the sub-path of P_{uv} with the end-points u' and v' as the path distance from u' to v' on P_{uv}.

Next, we discuss how we place the additional vertices along each path $P'_{uv} \in \Gamma'_G$ to obtain a LoS network G' from G. Start by lexicographically ordering the d-tuples of the vertices $\Gamma'_G(u)$. Suppose without loss of generality that $\Gamma'_G(u) < \Gamma'_G(v)$ in this ordering. Consider the $4 \cdot |P_{uv}| - 1$ internal points on the path P'_{uv}, if any, whose path distance to $\Gamma'_G(u)$ on P'_{uv} are

$$(\omega - 1), 2 \cdot (\omega - 1), 3 \cdot (\omega - 1), \ldots, (4 \cdot |P_{uv}| - 1) \cdot (\omega - 1).$$

Number these internal points $1, 2, \ldots, 4 \cdot |P_{uv}| - 1$, respectively. We now place $4 \cdot |P_{uv}| - 2$ additional vertices on each path $P'_{uv} \in \Gamma'_G$ at the internal points numbered $1, 2, \ldots, 4 \cdot |P_{uv}| - 3$, and $4 \cdot |P_{uv}| - 1$ (all numbered internal points apart from $4 \cdot |P_{uv}| - 2$). Finally, we place two additional vertices at points on P'_{uv} whose distance from $\Gamma'_G(u)$ are $(4 \cdot |P_{uv}| - 3) \cdot (\omega - 1) + 1$ and $(4 \cdot |P_{uv}| - 2) \cdot (\omega - 1) + 1$, respectively (either side of the internal point numbered $4 \cdot |P_{uv}| - 2$. This ensures we have added $4 \cdot |P_{uv}|$ additional vertices to each path P'_{uv}. Furthermore notice that all bends on P'_{uv} are covered by additional vertices as the distance from any bend to $\Gamma'_G(v)$ is a multiple of $4(\omega - 1)$ and so any such bend must belong to the set of internal points numbered $1, \ldots, 4 \cdot |P_{uv}| - 1$. In this set the only point which is not covered by an additional vertex is the point numbered $4 \cdot |P_{uv}| - 2$. Such point has distance $2(\omega - 1)$ from $\Gamma_G(v)$ and so cannot contain a bend. Finally we ensure that for any vertex in $u \in V(G) \cap V(G')$ the two additional vertices in G' placed either side of u (on different paths), which share a line of sight, are at a distance at least $2(\omega - 1)$, and therefore are not at distance less than ω, thus avoiding adding any unwanted edge after the placement of the extra vertices.

3.3 Correctness

Given a graph $G = (V, E)$, let $r : E \to \mathbb{Z}^+$ be a function defined over G's edges. Let G' be a subdivision of G formed by replacing each edge $e = \{u, v\}$ in E by the path $u, w^e_1, \ldots, w^e_{2 \cdot r(e)}, v$ containing $2 \cdot r(e)$ new vertices w^e_i all of degree 2. Thus G' has the vertex set $V' = V \cup (\cup_{e \in E} V_e)$ where $V_e = \{w^e_1, \ldots, w^e_{2 \cdot r(e)}\}$ and the edge set $E' = \cup_{e \in E} \mathcal{E}_e$, where $\mathcal{E}_e = \{\{u, w^e_1\}, \{w^e_1, w^e_2\}, \ldots, \{w^e_{2 \cdot r(e)}, v\}\}$.

Lemma 1. *Let $G = (V, E)$ be a graph and $r : E \to \mathbb{Z}^+$ be a function. Then G has an independent set of size at least k if and only if G' has an independent set of size at least $k + \sum_{e \in E} r(e)$.*

Proof. The only if part is easy. An independent set S' in G' is *good* if $S' \cap V$ is an independent set in G and S' uses half of the vertices on each of the paths in G'. The

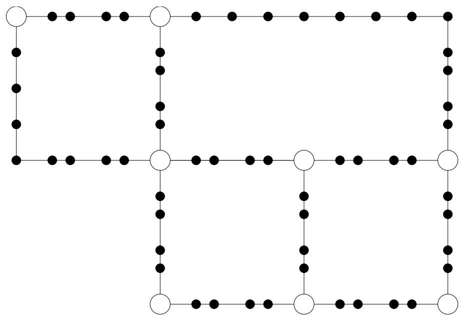

Figure 4: A 2-dimensionally spanning LoS network G' constructed from an orthogonal embedding of the planar graph G of maximum degree four with eight vertices (in white). The vertices added to complete $V(G')$ are in black.

argument is now split in two cases. If S' is good then we are done: $S' \cap V$ is an independent set in G of size $|S'| - \sum_{e \in E} r(e)$. If S' is not good, then two further cases arise. If $S' \cap V$ is an independent set in G, then it follows from the construction that its size is $|S'|$ minus the number of path vertices in S', and this number is clearly at least $|S'| - \sum_{e \in E} r(e)$. Finally, if $S' \cap V$ is not an independent set in G, some edge $e = \{u, v\}$ in E has both its end-points $S' \cap V$. But then rearranging the elements in S' belonging to $u, w_1^e, \dots, w_{2r(e)}^e, v$ so that only one of u or v is in the new S' cannot reduce the size of the resulting independent set. Thus after this is done for all edges $\{u, v\}$ in E that have both u and v in $S' \cap V$, we are left with the case when $S' \cap V$ is independent, which has already been handled. □

Note that Lemma 1 holds for any graph, it allows complete freedom in the definition of the mapping r, and it preserves the maximum vertex degree of the given graph. Therefore we can apply it to the graphs G and G' in the previous section thus completing the proof of Theorem 1.

4 APX-hardness for Gridline Graphs

The careful reader will realize that the proof of Theorem 1 goes through even when ω is not a fixed constant, but depends on n, when working with embeddings in $\mathbb{Z}_n^d$. However when, say, $\omega > n / \log n$, the size of the cube used to embed G' satisfies

$$n > k\,\omega\,|V| > k \frac{n}{\log n} |V|$$

for some fixed constant k only depending on d, assuming G' is obtained from $G = (V, E)$ via the reduction described in Section 3. But this implies that $n > \mathrm{e}^{k|V|}$. In other words G' is embedded in a cube that is exponentially bigger than G, and it is not obvious whether the reduction can be carried out in polynomial time. While we are not able to extend Theorem 1 to all values of ω, in this section we show that a different hardness proof works when $\omega = n$ showing that, in fact, MIS(GRIDLINE^d) is also hard to approximate. Let MAX 3SC-d denote the following optimization problem: given a collection C of subsets of size three of a set X, with each element of X belonging to at most d elements of C, find the largest number of disjoint subsets in C. The problem has been shown to be MAX SNP-complete for any (fixed) constant $d \geq 3$ [17]. Theorem 2 will follow once we prove that there is an approximation preserving reduction from MAX 3SC-d to MIS(GRIDLINE^d). We will use the following notion of reducibility.

Definition 1. *Let Π_1 and Π_2 be two optimisation problems and k_{Π_1} and k_{Π_2} their corresponding cost functions. There exists an L-*reduction *from Π_1* to *Π_2 (denoted $\Pi_1 \leq_L \Pi_2$) if there exist two polynomial time computable functions g and h and two constants α_1 and α_2 such that*

1. *If I is an instance of problem Π_1 then $g(I)$ is an instance of problem Π_2.*
2. *If S is a solution to $g(I)$ then $h(S)$ is a solution to I.*
3. $\mathrm{opt}_{\Pi_1}(g(I)) \leq \alpha_1 \cdot \mathrm{opt}_{\Pi_2}(I)$.
4. $|\mathrm{opt}_{\Pi_1}(I) - k_{\Pi_1}(h(S))| \leq \alpha_2 |\mathrm{opt}_{\Pi_1}(g(I)) - k_{\Pi_2}(S)|$ *for every solution S to $g(I)$.*

In this section we prove the following

Theorem 8. MAX 3SC-$d \leq_L$ MIS(GRIDLINE^d) *for each fixed integer $d \geq 3$.*

To simplify the presentation the proof is split in two parts. In the first one (Section 4.1) we reduce MAX 3SC-d to MIS in graphs of bounded degree $d+1$. In fact we obtain a family of reductions parameterized by a certain path length parameter t. Then we prove that, for a particular value of t, the graphs resulting from such reduction are in fact elements of GRIDLINE^d. This is done in two steps. In Section 4.2 we describe our main technical tools while in Section 4.3 these are applied to the appropriate graphs and the description of the embedding and its correctness is completed.

4.1 Reinventing the Wheel: APX-hardness of MIS in Bounded Degree Graphs

The outcome of this section is an alternative proof that MIS is APX-hard on bounded degree graphs. The original proof [22], involved an approximation preserving reduction from a bounded version of MAX 3SAT (the definition of this classical optimization problem can be found in standard textbooks like [12, 14]). The particular feature of our construction is that we obtain a family of L-reductions parameterized by a certain parameter t, a positive integer. Choosing t in a particular way will complete the proof of Theorem 8.

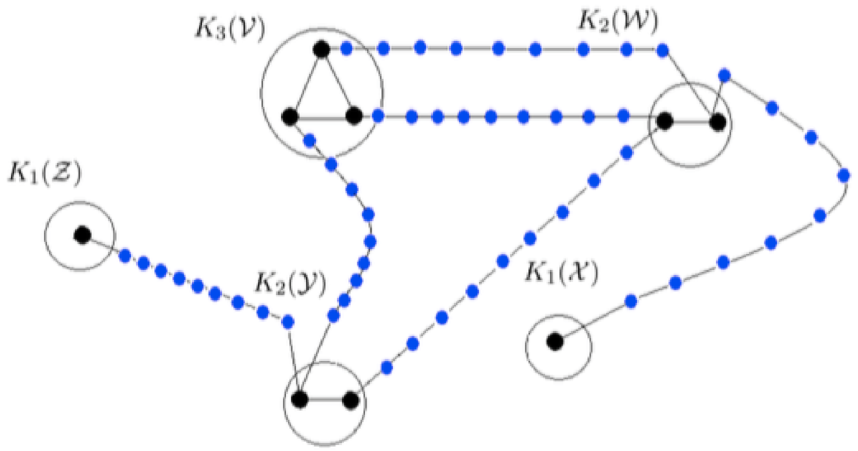

Figure 5: The graph G_I corresponding to an instance I where $X = \{\mathcal{V}, \mathcal{W}, \mathcal{X}, \mathcal{Y}, \mathcal{Z}\}$ and $C = \{\{\mathcal{V}, \mathcal{W}, \mathcal{X}\}, \{\mathcal{V}, \mathcal{W}, \mathcal{Y}\}, \{\mathcal{V}, \mathcal{Y}, \mathcal{Z}\}\}$. Here d might be any fixed value greater than two and $t = 5$.

Each instance I of MAX 3SC-d is a triple formed by d, a set X, and a collection $C = \{c_1, \ldots, c_r\}$ of three element subsets of X such that each $\mathcal{X} \in X$ is contained in $\text{occ}(\mathcal{X}) \leq d$ elements of C. We construct the graph G_I as follows. For each $\mathcal{X} \in X$, G contains the clique on $\text{occ}(\mathcal{X})$ vertices, denoted $K_{\text{occ}(\mathcal{X})}(\mathcal{X})$ (cliques corresponding to distinct elements of X are vertex disjoint). The elements of C will become paths in G_I. Assume the elements of X are ordered arbitrarily. For each set $c = \{\mathcal{X}, \mathcal{Y}, \mathcal{Z}\} \in C$ assume, without loss of generality, that $\mathcal{X} < \mathcal{Y} < \mathcal{Z}$. The graph G_I contains the path, denoted P_c, of even length which contains a vertex $u \in K_{\text{occ}(\mathcal{X})}(\mathcal{X})$, a vertex $v \in K_{\text{occ}(\mathcal{Y})}(\mathcal{Y})$, and a vertex $w \in K_{\text{occ}(\mathcal{Z})}(\mathcal{Z})$. Vertex u (resp. v) is then connected to v (resp. w) by a path of length $2t$. We refer to u, v, and w as the *base vertices* of P_c. Distinct elements of C get associated with distinct triples of base vertices and vertex disjoint paths. This complete the definition of the function g in Definition 1. Figure 5 shows a small example for $d = 3$ and $t = 5$.

Next we show that, given any independent set S in G_I, we can construct a cover of C. The elements of C correspond to paths in G_I. We say that *S covers P_c* for some $c \in C$ if the three base vertices of P_c belong to S. It is easy to see that any independent set of G_I can be converted in polynomial time in a (possibly larger) independent set S' such that for any P_c, either S' covers P_c or it contains none of the three base vertices (we call *good* such independent sets). Define $C_S \subseteq C$ to be the collection of those elements of C corresponding to paths that are covered by S'.

Consider now an optimal cover of C. Then we can construct an independent set in G_I of size $(2t + 1)\text{opt}(I) + 2t(|C| - \text{opt}(I))$ and thus $\text{opt}(G_I) \geq \text{opt}(I) + 2t|C|$. Conversely, any maximum independent set $S_{\max}$ in G_I must be good. Thus we can

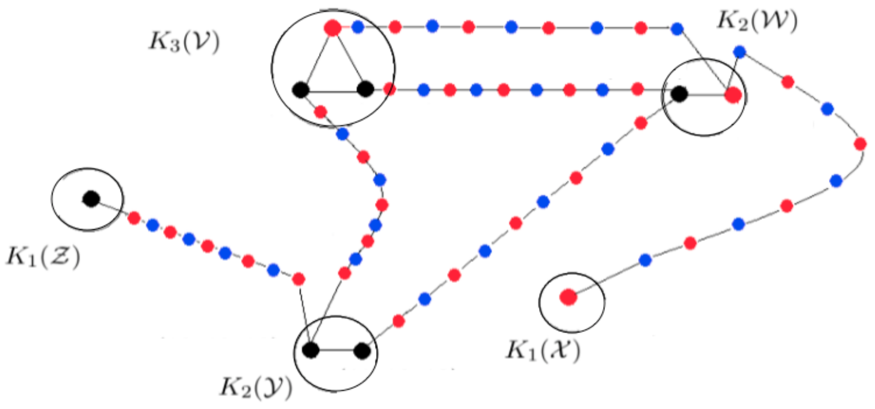

Figure 6: A largest independent set in G_I. The path contributing $2t+1=11$ vertices corresponds to the cover $\{\mathcal{V},\mathcal{W},\mathcal{X}\}$ in I.

define a cover $C_{S_{\max}}$ that satisfies:

$$\mathrm{opt}(G_I) = (2t+1)|C_{S_{\max}}| + 2t(|C| - |C_{S_{\max}}|) = |C_{S_{\max}}| + 2t|C|.$$

Hence

$$|C_{S_{\max}}| = \mathrm{opt}(G_I) - 2t|C| \le \mathrm{opt}(I)$$

and we have therefore proved

$$\mathrm{opt}(I) = \mathrm{opt}(G_I) - 2t|C|. \tag{1}$$

This equality can be used to complete the reduction. Since each element $\mathcal{X} \in X$ appears in at most d sets, it follows that $\mathrm{opt}(I) \ge \frac{|C|}{3d}$. By substituting this into equation (1), we obtain

$$\mathrm{opt}(G_I) \le (6dt+1)\cdot \mathrm{opt}(I).$$

Finally, given an instance I of MAX 3SC-d, our reduction defines G_I, and for each independent set S in such graph a cover C_S in I. Because of (1) we have

$$\mathrm{opt}(I) - |C_S| = \mathrm{opt}(G_I) - 2t|C| - |C_S| \le \mathrm{opt}(G_I) - |S|$$

as C_S is always derived from a good independent set whose size is at least $|S|$. Thus we have verified the fourth condition in Definition 1, with $\alpha_2 = 1$, and the proof is complete.

4.2 Embedding Cliques and Paths

So far we have presented a reduction showing the APX-hardness of MIS in graphs of bounded degree $d+1$. Note that the reduction goes through as long as the paths

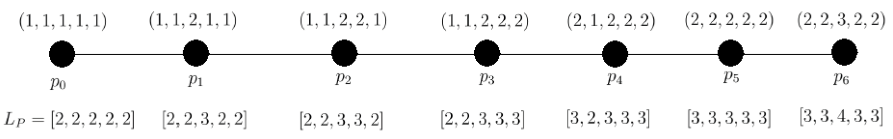

Figure 7: The path $P = p_0, p_1, p_2, p_3, p_4, p_5, p_6$ embedded using the path rotation with initial d-tuple $(1, 1, 1, 1, 1)$ assigned to p_0, list $L = [2, 2, 2, 2, 2]$, and $k = 2$. The updated lists are also displayed.

connecting the cliques are of even length. In the rest of this section we assume that $d > 2$, $2t = 3d + d \mod 2$ and we prove that for each instance $I = (d, X, C)$ of MAX 3SC-d, G_I belongs to GRIDLINE^d. Note that each G_I is the union of $|X|$ cliques and $|C|$ paths. Thus, to complete our argument, we need a systematic way to embed cliques and paths in a sufficiently large d-dimensional cube. Embedding cliques is straightforward. Namely, let CLIQUE_EMBED$(b, \boldsymbol{x})$ denote the embedding of a copy of K_b, with $b \leq d$, in a d-dimensional cube *starting* at point $\boldsymbol{x}$. Specifically, if $\boldsymbol{x} = (x_1, \ldots, x_d)$, the clique is embedded to points

$$(x_1, \ldots, x_d), (x_1 + 1, \ldots, x_d), \ldots, (x_1 + b - 1, \ldots, x_d).$$

Paths are a bit more complicated. However, their particular length plays an important role. In what follows, if u is a vertex in a given LoS network, let $f(u)_i$ denote the ith coordinate value of an embedding function $f(u)$. Given two vertices u and v which are embedded in $\mathbb{Z}_n^d$, if $f(u)$ and $f(v)$ share a line of sight, let $\oplus(u, v)$ denote the co-ordinate position $1 \leq i \leq d$ such that $f(u)_i \neq f(v)_i$.

Path Rotation. The path rotation embedding works for an arbitrary finite path $P = p_0, p_1, p_2, \ldots$ and assigns a d-dimensional point to each vertex of P sequentially, starting from a point $\boldsymbol{x}$ which is initially set to be $f(p_0)$. The embedding uses a d element tuple of positive integers (with $L[i] \neq x_i$ for every i), and an integer k. The algorithm return an embedding of P. The structure L is accessed *by reference*. This implies that any change to L during the execution of ROTATION_EMBED is permanent (this will be used further down when we will discuss embedding paths with pre-specified milestone points). Figure 7 gives an example of a short path and its rotation embedding. The idea is that P is embedded, starting from p_0, by iteratively defining $f(p_l)$ from $f(p_{l-1})$ so that for each l, $f(p_l)$ and $f(p_{l-1})$ share a line of sight but, say, $f(p_{l+1})$ and $f(p_{l-1})$ do not, as their coordinates differ in two positions. The structure L is used to make sure that the embedding (at least for the types of paths used in our reduction) is injective. Algorithm 1 formally describes the embedding process.

Path Connection. The second embedding algorithm is used to deal with paths of length d. Figure 8 gives an example. The process takes four parameters: the given path P, two points $\boldsymbol{x}$ and $\boldsymbol{y}$ used to embed the path's two end-points, and a permutation σ of $(1, \ldots, d)$. Starting from p_0 and its image $\boldsymbol{x}$, the algorithm defines the embedding

Algorithm 1 ROTATION_EMBED$(P, L, k, \boldsymbol{x}, d)$

```
1: set $f(p_0) = \boldsymbol{x}$
2: for $l = 1, \ldots, |P|$ do
3:     $f(p_l) = f(p_{l-1})$
4:     $f(p_l)_{(l+k) \mod d} = L[(l+k) \mod d]$
5:     $L[(l+k) \mod d] = L[(l+k) \mod d] + 1$
6: end for
```

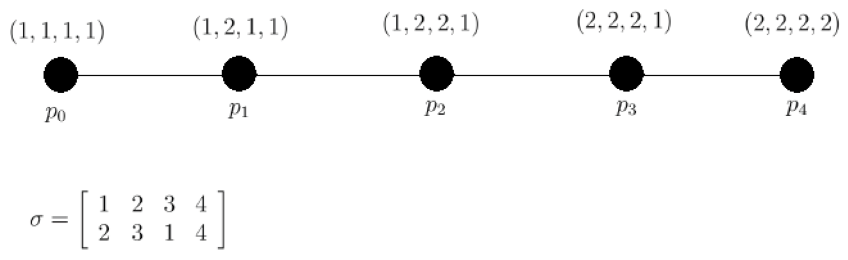

Figure 8: The path $P = p_0, p_1, p_2, p_3, p_4$ embedded using the path connection method with initial d-tuples $(1, 1, 1, 1)$ and $(2, 2, 2, 2)$ assigned to p_0 and p_4, and $\sigma = (1, 2, 3)(4)$

of vertex p_l by changing the position of $f(p_{l-1})$ from the value in $\boldsymbol{x}$ to that in $\boldsymbol{y}$. The permutation σ is used to decide which position to change. In our application of this method we will impose restrictions on σ. Note that the points used in a call of the form CONNECTION_EMBED$(P, \boldsymbol{x}, \boldsymbol{y}, \sigma, d)$ all belong to a cube of size equal to $\max |x_i - y_i|$ and only involve co-ordinate values belonging either to $\boldsymbol{x}$ or $\boldsymbol{y}$. Algorithm 2 formally describes the embedding process.

Algorithm 2 CONNECTION_EMBED$(P, \boldsymbol{x}, \boldsymbol{y}, \sigma, d)$

```
1: set $f(p_0) = \boldsymbol{x}$ and $f(p_d) = \boldsymbol{y}$
2: for $l = 1, \ldots, d-1$ do
3:     $f(p_l) = f(p_{l-1})$
4:     $f(p_l)_{\sigma(l)} = y_{\sigma(l)}$
5: end for
```

Embedding Paths With Prescribed Milestones. When constructing the appropriate embedding, we do not have complete freedom associating integer tuples to vertices, in particular, the embedding must touch certain specific points. When the length of the path is at most d, path connection is sufficient to enforce such condition, but in general we need to use a method that combines rotations and connections.

Assume that we need to embed a long path P, and that the start and end vertices must have prescribed images $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. We can then pick an index s and embed $P' = p_0, \dots, p_s$ and then $P'' = p_{|P|}, \dots, p_{s+d}$ using path rotation (we use a single list L but, in general, two distinct values for the parameter k in the rotation process). This will define $f(p_s)$ and $f(p_{s+d})$. Clearly these are two points in a d-dimensional cube, therefore at most d flips of the coordinates of $f(p_s)$ are sufficient to change that to $f(p_{s+d})$. Therefore we can complete the embedding of P using path connection to deal with $P''' = p_s, \dots, p_{s+d}$. We emphasize that when selecting the permutation σ in the final part of the embedding, σ must be chosen so that $\sigma(1) \neq \oplus(p_{s-1}, p_s)$ and $\sigma(d) \neq \oplus(p_{s+d}, p_{s+d+1})$. This is to avoid adding any unwanted edges between the pairs of vertices p_{s-1} and p_{s+1}, and p_{s+d-1} and p_{s+d+1}, respectively. The resulting process, which we call LONG_EMBED, is formally described by Algorithm 3. As already mentioned, notice that, in general, the content of the list L resulting after the execution of ROTATION_EMBED on line 1 may be different from that at the beginning of the process.

Algorithm 3 LONG_EMBED$(P, s, L, k, \boldsymbol{x}, \boldsymbol{y}, \sigma, d)$

```
1: ROTATION_EMBED$(P', L, k, \boldsymbol{x}, d)$
2: ROTATION_EMBED$(P'', L, k, \boldsymbol{y}, d)$
3: CONNECTION_EMBED$(P''', f(p_s), f(p_{s+d}), \sigma, d)$
```

It is easy to verify that in all embeddings presented so far, vertices u and v are adjacent in the given path P if and only if their embedding share a line of sight.

4.3 Overall Embedding

The main result of this section is the following:

Lemma 2. *Let $d \geq 3$ be a (fixed) integer. Given an instance $I = (d, X, C)$ of* MAX 3SC-d*, the graph G_I defined according to the reduction described in Section 4.1 for $2t = 3d + d \mod 2$ is a LoS network with parameters d, N, and $\omega = N$, where $N \leq |X| \cdot d + 10 \cdot |C|$.*

Proof. Given an instance $I = (d, X, C)$ of MAX 3SC-d, we embed G_I by first looking at its cliques $K_{\mathrm{occ}(\mathcal{X})}(\mathcal{X})$, and then at the paths P_c for all $c \in C$. Embedding the cliques is quite straightforward. For the jth clique we use a sub-cube[1] isomorphic to $\mathbb{Z}_d^d$, starting from

$$\boldsymbol{x}^j = ((j-1)d + 1, \dots, (j-1)d + 1).$$

Figure 9 gives an example.

Embedding the paths is slightly more involved. We can use rotations and connections, but we need to be careful. Note that for each $c \in C$, the path P_c has length $3d + d \mod 2 > d$, thus the path connection method on its own does not suffice. Also, each P_c involves three distinct base vertices, therefore we cannot use the path rotation method as it is as that would make it hard to control the embedding of the base vertices. However, we can rely on a combination of the two. Assume that the elements of

[1]In fact the embedding aligns the images of the vertices of the clique on a single 1-dimensional line.

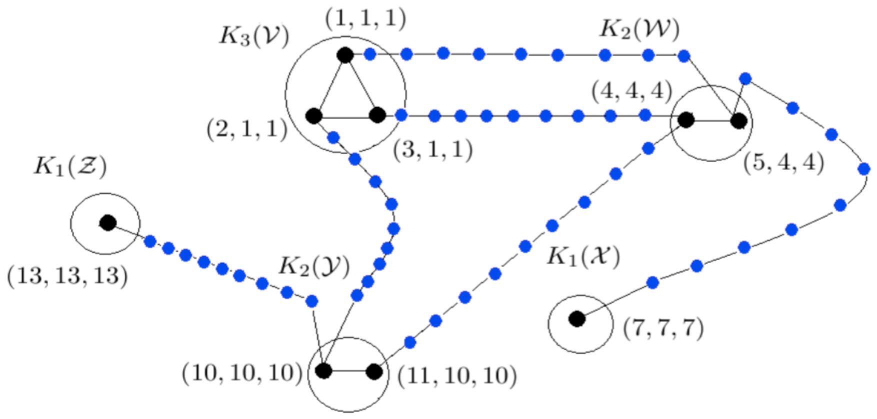

Figure 9: An embedding of the vertices of the cliques in the graph G_I, corresponding to an instance I of MAX 3SC-3.

C are ordered arbitrarily. We embed, one after the other, all paths $P_{c_1}, P_{c_2}, \ldots$ in the following manner. For each $j \in \{1, \ldots, |C|\}$, let

$$P_{c_j} = \overbrace{u, p_j^1, \ldots, p_j^{3d-1+d \mod 2}, v}^{P'_{c_j}}, \underbrace{v, p_j^1, \ldots, p_j^{3d-1+d \mod 2}, w}_{P''_{c_j}}.$$

Path P_{c_j} is embedded using lists L_{2j-1}, L_{2j} through:

$$\textsc{Long\_Embed}(P'_{c_j}, d + d \mod 2, L_{2j-1}, 0, \boldsymbol{x}, \boldsymbol{y}, \sigma, d)$$

and then

$$\textsc{Long\_Embed}(P''_{c_j}, d + d \mod 2, L_{2j}, 1, \boldsymbol{y}, \boldsymbol{z}, \sigma', d)$$

where $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ are the images of the three base vertices of P_c, permutations σ and σ' are chosen as described at the end of Section 4.2 to avoid unwanted edges, and the jth list L_j is set to

$$[|X|d + 5j - 4, \ldots, |X|d + 5j - 4]$$

for each $j \in \{1, \ldots, 2|C|\}$. Figure 10 shows a small example.

To argue that the overall embedding works note that, if N is at least $|X|d$, the distinct cliques are embedded to disjoint parts of $\mathbb{Z}_N^d$. Also, the lists L_j are *well separated* in the sense that, for every j, the largest value in L_j after a call to Long_Embed involving L_j is smaller than the smallest value in L_{j+1}. This implies that, if N is at least $|X|d + 10|C|$, the embeddings of the paths P_c (each involving four path rotation and two path connection embeddings), for different values of c, span disjont parts of $\mathbb{Z}_N^d$. To be more specific, given P_c and $P_{c'}$ for $c \neq c' \in C$, the paths' base vertices are all distinct, and they are all mapped to distinct points. Furthermore, it is easy to see

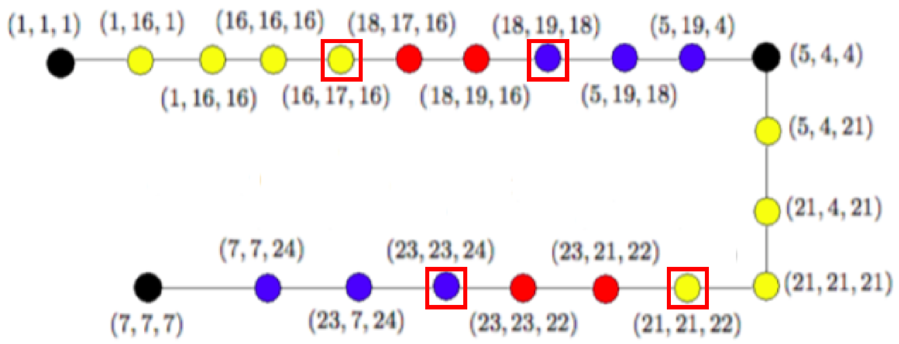

Figure 10: An embedding of the path with base vertices $(1, 1, 1)$, $(4, 4, 4)$ and $(7, 7, 7)$ for the graph G_I in Figure 5, 6, and 9. Let P' denote the path between vertices $(1, 1, 1)$ and $(4, 4, 4)$ and P'' denote the path between vertices $(4, 4, 4)$ and $(7, 7, 7)$. The yellow vertices on each path correspond to the initial path rotation embedding on P' and P'', respectively, the blue vertices correspond to the next path rotation embedding, and finally, the red vertices are those whose image is defined via the path connection embedding. Vertices surrounded by a red square are assigned an image first by path rotation, then by path connection. As $|X| = 5$, we use $L_1 = [16, 16, 16]$ and $L_2 = [21, 21, 21]$.

that any pair of intermediate vertices p and p' (one in P_c, the other in $P_{c'}$) excluding, for now, those adjacent to the base vertices, must be mapped to points that differ in at least two positions from any base vertex and, because of the well separatedness of the lists used to embed P_c and $P_{c'}$, the images of p and p' cannot share a line of sight. Finally, we mention that some care is necessary, when embedding P_c, with base vertices $u \in K_{\text{occ}(\mathcal{X})}(\mathcal{X})$, $v \in K_{\text{occ}(\mathcal{Y})}(\mathcal{Y})$ and $w \in K_{\text{occ}(\mathcal{Z})}(\mathcal{Z})$ as this may create unwanted edges between vertices on a path and any (other) vertex in $K_{\text{occ}(\mathcal{X})}(\mathcal{X})$, $K_{\text{occ}(\mathcal{Y})}(\mathcal{Y})$, or $K_{\text{occ}(\mathcal{Z})}(\mathcal{Z})$. Parameters k, σ and σ' in the calls to LONG_EMBED prevent such patological cases. $\square$

5 Approximation Algorithms

The results described so far show that in many cases MIS is not likely to be solved efficiently even if we know that the input graph is in fact a LoS network. In this section we further exploit the computational properties of MIS in LoS networks by focusing on a number of algorithmic results. We first show that LoS networks belong to classes of graphs that generalise chordal graphs. Theorem 3 can then be seen as a special case of a more general result valid for graphs in these classes. The hardness proofs of Section 3 and 4 will then be complemented by two additional algorithmic results. First, we propose a polynomial time approximation scheme that works for fixed ω. This will prove Theorem 4. Then we describe a local improvement strategy that beats the approximation guarantee of Theorem 3 for the extreme case $\omega = n$. This will lead to the proof of Theorem 5.

5.1 Clique Neighbourhoods in LoS Networks

Kammer *et al.* [16] proposed a few classes of graphs that generalise chordal graphs, and argued about the existence of good approximation algorithms for a few problems that admit exact polynomial time computable solutions on chordal graphs, when restricted to instances in these new classes. We introduce two of the complexity parameters studied by these authors, and prove that different types of LoS networks belong to the classes of graphs defined by these parameters. As a consequence we will obtain constant factor approximations results promised in Section 2.

Definition 2. *A graph G is k-*perfectly groupable *if the neighbors of each vertex v can be partitioned into k sets $S_1, \ldots, S_k$ (possibly empty) such that $G[S_h \cup \{v\}]$ is a clique for each $h \in \{1, \ldots, k\}$.*

For example, unit disk graphs are 6-perfectly groupable while graphs of maximum degree k are k-perfectly groupable.

Definition 3. *A graph G is k-*simplicial *if there is an ordering $v_{j_1}, \ldots, v_{j_{|V(G)|}}$ of the vertices of G such that for each vertex v_{j_l}, $1 \leq l \leq |V(G)|$, the set of neighbors of v_{j_l} among $v_{j_{l+1}}, \ldots, v_{j_{|V(G)|}}$ can be partitioned into k sets $S_1, \ldots, S_k$ (possibly empty) such that $G[S_h \cup \{v_{j_l}\}]$ is a clique for each $h \in \{1, \ldots, k\}$. Each neighbour of v_{j_l} among $v_{j_{l+1}}, \ldots, v_{j_{|V(G)|}}$ is called a* successor *of v_{j_l}, and the aforementioned ordering of the vertices in G is called a k-*simplicial elimination order.

Note that all k-perfectly groupable graphs are also k-simplicial (any ordering of the vertices is a k-simplicial elimination ordering) while the opposite implication is false: the n vertex tree with $n-1$ leaves is 1-simplicial but NOT k-perfectly groupable for any $k < n-1$. The following is the general algorithmic result that implies Theorem 3 and 5.

Theorem 9. [1] *MIS is k-approximable on k-simplicial graphs.*

As we pointed out in the introduction, the proof of Theorem 9 is based on a simple greedy strategy that had been successfully applied in the context of unit disk graphs (Algorithm 4 describes such strategy for the problem at hand). The merit of Kammer, and other people who have studied the parameters above (see [1, 16, 27]) was to realize that the same algorithm had much more general applicability: the resulting greedy process peels off the vertices of the given graph G following some k-simplicial elimination ordering. A recursive process ensures that a feasible solution can be retrieved as the vertices are then put back into the graph in reverse order.

To complete our argument, we will now prove two results that characterize LoS networks with respect to the two parameters above. For any vertex $u \in G$, let $f(u) = \boldsymbol{x} = (x_1, \ldots, x_d)$. Also, for each $i \in \{1, \ldots, d\}$ let $N_i(u)$ denote the subset of neighbours v of u whose d-tuples share an i-line (i.e. those vertices v for which $f(v) = (x_1, \ldots, x_{i-1}, y^v, x_{i+1}, ldots, x_d)$, for some y^v). Also, let

$$N_i^-(u) = N_i(u) \cap \{v \mid y^v < x_i\},$$

$$N_i^+(u) = N_i(u) \cap \{v \mid y^v > x_i\}.$$

Algorithm 4 GREEDY_IS_WITH_NO_EMBEDDING(G, k)

```
1: Initialise $S$ to be the empty set
2: while $V(G) \neq \emptyset$ do
3:     select $u \in V(G)$ such that $\alpha(G[N(u)]) \leq k$
4:     add $u$ to $S$
5:     remove $u \cup N(u)$ from $G$
6: end while
```

Lemma 3. *Let d, n and ω be positive integers with $1 \leq \omega \leq n$. Any graph $G \in L^d_{n,\omega}$ is $2d$-perfectly groupable (resp. d-perfectly groupable) if $\omega < n$ (resp. $\omega = n$).*

Proof. If $\omega = n$, the vertices in $N_i(u) \cup \{u\}$ all share a line of sight and there is an edge connecting any two elements of the set. Thus $G[\{u\} \cup N_i(u)]$ forms a clique for each $i \in \{1, \dots, d\}$. For $\omega < n$, for each vertex u, we consider the $2d$ sets $N_i^-(u)$ and $N_i^+(u)$, and a similar argument shows that G is $2d$-perfectly groupable. □

Taking into account Theorem 9, Lemma 3 implies Theorem 3 for $\omega = n$. When $\omega < n$ we need to work a bit harder: the graphs considered may not be d-perfectly groupable as pairs of elements in a set $N_i(u)$ for some $u \in V(G)$ might be at distance $2(\omega - 1)$. However, it is possible to define a d-simplicial elimation ordering in these graphs and Theorem 9 applies with $k = d$. In what follows a *corner* in a LoS network G is a vertex u such that $N_i(u) = N_i^-(u)$ or $N_i^+(u)$, for each $i \in \{1, \dots, d\}$.

Lemma 4. *Let d, n and ω be positive integers with $1 \leq \omega \leq n$. Any graph $G \in L^d_{n,\omega}$ is d-simplicial.*

Proof. The result follows from Lemma 3 for $\omega = n$. Here we present an argument that is valid for every ω.

Let G be a LoS network with parameters d, n, and $\omega < n$. If the points of $\mathbb{Z}^d_n$ are ordered lexicographically, the extrema (i.e. maximal or minimal elements) of such ordering that are images of elements of $V(G)$ define corner vertices. We claim that any sequence $v_{j_1}, \dots, v_{j_{|V(G)|}}$ in which v_{j_l} is a corner vertex for each l defines a d-simplicial elimination ordering. Indeed, if v_{j_l} is a corner vertex, then $N_i(v_{j_l})$, for each l is a clique, for each $i \in \{1, \dots, d\}$. □

Robustness. Note that all algorithmic results in this Section are robust, in the sense of the definition discussed in Section 2. Algorithm 4 does not require any knowledge of the embedding of the input graph. Lemma 3 and 4 imply that Algorithm 4 works corrrectly on any LoS network.

5.2 An Efficient Polynomial Time Approximation Scheme

In this section we describe an algorithm, which we call PTAS_IS, that accepts as input any d-dimensional LoS network $G = (V, E)$ with the (constant) range parameter ω and a parameter $\epsilon > 0$, and returns an independent set in G of size at least $\alpha(G)/(1 + \epsilon)$. The algorithm does not rely on a geometric representation, it accepts any graph as an input-instance. However, the statements concerning the running time depend on the assumption that the graph is a LoS network. The proposed algorithm mimics an approximation scheme [21] used to approximate MIS in *unit disk graphs*. The algorithm in this section however (see Theorem 10 below) has a running time that, for any fixed ω, is linear in the number of vertices of the input graph (although the constant hidden in the big-Oh notation depends exponentially on ϵ^{-1}). Algorithms of this type have been referred to as *efficient* polynomial time approximation schemes [3].

PTAS_IS works in iterations, by assembling an independent set as the union of partial solutions from disjoint portions of G. Namely, let r be a non-negative integer. For any $u \in V(G)$, we denote by $N^r[u]$ the so called *r-th closed neighbourhood of u*, the set of vertices at (graph) distance at most r from u. Note that $N^0[u] = u$, and in general $N^r[u]$ contains all vertices whose embedding lays within distance $r(\omega - 1)$ from the embedding of u. Starting from an arbitrary $u \in V(G)$, the algorithm computes (by complete enumeration) a maximum independent set S_r in each of $G[N^0[u]], G[N^1[u]], G[N^2[u]], \ldots$ as long as

$$|S_{r+1}| \geq (1+\epsilon)|S_r|. \tag{2}$$

Let $\bar{r}$ denote the smallest positive value of r that violates this constraint. Once the algorithm reaches $\bar{r}$, it stops, adds $S_{\bar{r}}$ to the independent set that is being built and then starts a new iteration that will work on $G' = G \setminus N^{\bar{r}+1}[u]$. The process terminates as soon as G' becomes empty.

Correctness. The set $S_{\bar{r}}$ is an independent set of $G[N^{\bar{r}}[u]]$. To achieve an independent set for the graph G, the basic process described above is iteratively applied to the graph $G' = G \setminus N^{\bar{r}+1}[u]$. The correctness of the overall process follows from an inductive proof based on the following statement.

Claim 1. *Suppose inductively that we can compute an independent set $S' \subset V \setminus N^{\bar{r}+1}[u]$ for G' of size at least $\alpha(G')/(1+\epsilon)s$. Then $S = S_{\bar{r}} \cup S'$ is an independent set for G that satisfies:*

$$\frac{\alpha(G)}{|S|} \leq 1 + \epsilon.$$

Proof. Since each $v \in V \setminus N^{\bar{r}+1}[u]$ has no neighbour in $N^{\hat{r}}[u]$ it follows that S is an independent set in G as required. Furthermore, by the definition of $\bar{r}$, it follows that

$$|S_{\bar{r}+1}| \leq (1+\epsilon) \cdot |S_{\bar{r}}|.$$

In other words

$$\alpha(G[N^{\bar{r}+1}[u]]) \leq (1+\epsilon) \cdot |S_{\bar{r}}|.$$

The result follows then from the assumption on $|S'|$ and by the sub-additivity of $\alpha(G)$:

$$\begin{array}{rcl} \alpha(G) & \leq & \alpha(G[N^{\bar{r}+1}[u]]) + \alpha(G[V \setminus N^{\bar{r}+1}[u]]) \\ & \leq & (1+\epsilon) \cdot |S_{\bar{r}}| + (1+\epsilon) \cdot |S'| \\ & = & (1+\epsilon) \cdot |S|. \end{array}$$

□

Time Complexity. To complete the proof of Theorem 4 one needs to bound the running time of the algorithm. The main statement of this part is the following theorem.

Theorem 10. *Let d, and ω be fixed positive integer constants, let n be a positive integer, and let ϵ be an arbitrary positive real number. There is a function $T_d(\omega, \epsilon)$ such that the running time of algorithm* PTAS_IS *when applied to a LoS network G with parameters d, n and ω, is bounded by $T_d(\omega, \epsilon) \cdot |V(G)|$.*

Algorithm PTAS_IS runs for at most $|V(G)|$ iterations. During each iteration, the process computes (by brute force enumeration) $\bar{r}+1$ optimal independent sets in small subsets of $V(G)$. The next two results provide the additional details needed to define $T_d(\omega, \epsilon)$.

Lemma 5. *Let d, and ω be fixed positive integer constants, let n and r be positive integers, and ϵ and arbitrary positive real number. Define*

$$b_d(\omega, r) = (2r(\omega-1)+1)^{d-1} \left\lceil \frac{2r(\omega-1)+1}{\omega} \right\rceil .$$

Let G be a LoS network with parameters d, n and ω. For any $u \in V(G)$, a maximum cardinality independent set in $N^r[u]$ can be found in time $(2r(\omega-1)+1)^{d \cdot b_d(\omega,r)}$.

Proof. It follows from the definition of $N^r[u]$ that the smallest portion of $\mathbb{Z}_n^d$ that is guaranteed to contain $N^r[u]$ is the ball

$$B^r(u) = \{\boldsymbol{y} \mid \mathrm{dist}_\infty(\boldsymbol{y}, f(u)) \le r(\omega-1)\}$$

(here $\mathrm{dist}_\infty(\boldsymbol{x}, \boldsymbol{y}) = \max |x_i - y_i|$). For each $j \in \{1, \ldots, d\}$ at most

$$\left\lceil \frac{2r(\omega-1)+1}{\omega} \right\rceil$$

independent vertices in $B^r(u)$ can share a j-line and, in d dimensions, there is $(2r(\omega-1)+1)^{d-1}$ such lines in $B^r(u)$. Hence, a crude upper bound on $|S_r|$ is

$$b_d(\omega, r) = (2r(\omega-1)+1)^{d-1} \left\lceil \frac{2r(\omega-1)+1}{\omega} \right\rceil .$$

Therefore, the time needed to find a maximum independent set in $N^r[u]$ is at most

$$\sum_{i=1}^{b_d(\omega,r)} \binom{(2r(\omega-1)+1)^d}{i}$$

□

Finally, we look at the length of each iteration. It follows from (2), the definition of $\bar{r}$, and the fact that $|S_0| = 1$ that for each $r < \bar{r}$,

$$|S_r| \ge (1+\epsilon)^r. \tag{3}$$

Furthermore, as we have seen in the proof of Lemma 5.

$$|S_r| \leq (2r(\omega-1)+1)^{d-1}\left\lceil \frac{2r(\omega-1)+1}{\omega} \right\rceil \tag{4}$$

Equations (3) and (4) along with the definition of $\bar{r}$ imply that $\bar{r}$ is bounded by the smallest integer r^* satisfying

$$(2r(\omega-1)+1)^{d-1}\left\lceil \frac{2r(\omega-1)+1}{\omega} \right\rceil < (1+\epsilon)^r. \tag{5}$$

Lemma 6. *The integer r^* defined above satisfies:*

$$r^* \leq \omega \cdot \left(\frac{2^d(2d-1)!}{(\epsilon-\epsilon^2/2)^{2d-1}} \right)^{\frac{1}{d-1}}.$$

Proof. Equation (5) is satisfied if

$$\omega^{d-1}(2r)^d < (1+\epsilon)^r.$$

Using the Taylor expansion of the exponential function we can write

$$(1+\epsilon)^r = \mathrm{e}^{r\log(1+\epsilon)} = \sum_{k=0}^{\infty} \frac{(r\log(1+\epsilon))^k}{k!} > \frac{(r\log(1+\epsilon))^t}{t!}$$

for any $t \geq 0$. Thus equation (5) is satisfied if

$$\omega^{d-1}(2r)^d < \frac{(r\log(1+\epsilon))^t}{t!}.$$

Note that, if $\epsilon \geq 2$, choosing $t = 2d-1$, and performing simple rearrangements we get that (5) is satisfied if

$$r > \left(\frac{2^d\omega^{d-1}(2d-1)!}{(\log 3)^{2d-1}} \right)^{\frac{1}{d-1}} = \omega \cdot \left(\frac{2^d(2d-1)!}{(\log 3)^{2d-1}} \right)^{\frac{1}{d-1}}.$$

On the other hand, for any $\epsilon \in (0,2)$,

$$\log(1+\epsilon) > \epsilon - \epsilon^2/2 > 0.$$

Thus, if $t = 2d-1$, equation (5) is satisfied if

$$r > \omega \cdot \left(\frac{2^d(2d-1)!}{(\epsilon-\epsilon^2/2)^{2d-1}} \right)^{\frac{1}{d-1}}$$

and the result follows. □

Robustness. Observe that neither the description of PTAS_IS at the beginning of the section nor the subsequent correctness argument use any properties of the embedding of the given LoS network. The algorithm thus always returns a $(1+\epsilon)$-approximate independent set, in any graph G. However, the running time analysis hinges on the fact that the size of the independent sets computed in each iteration is polynomially bounded in r. For a general graph, the running time may thus not be polynomial. So, during the execution of the algorithm, if an independent set of size greater than

$$\left\lceil \frac{2r(\omega-1)+1}{\omega} \right\rceil$$

can be found, this set is returned as certificate of non-membership in the given class of LoS networks.

5.3 An Improved Approximation Algorithm

In this section we describe an improved approximation strategy for MIS which works for any d-perfectly groupable graph. This improves the result stated in [16] and allow us to prove Theorem 5. Consider the following algorithm:

Algorithm 5 BETTER_IS(G,t)

```
1: set $S'$ to be any element of $V(G)$ and $S$ to be the empty set
2: while $S \neq S'$ do
3:     replace the content of $S$ with that of $S'$
4:     set $S'$ to the result of IMPROVE$(G,S,t)$
5: end while
```

where IMPROVE(G,S,t) is defined as follows:

```
IMPROVE$(G,S,t)$:
1: for $p = 1$ to $t$ do
2:     for each set $U$ with $p$ elements in $S$ do
3:         for each set $W$ with $p+1$ elements in $V(G) \setminus S$ do
4:             set $S'$ to $(S \setminus U) \cup W$
5:             if $S'$ is an independent set then
6:                 return $S'$
7:             end if
8:         end for
9:     end for
10: end for
11: return $S$
```

Observe that the larger values of the parameter t lead to better solutions at the price of a significant increase in the algorithm running time (each iteration of IMPROVE(G,S,t) runs in time $O(|V(G)|^{2t})$). The algorithm main computation loop (implemented by the function IMPROVE$(G, S, t))$ attempts to increase the size of the current independent set S by swapping p-tuples of elements in the set with $(p+1)$-tuples of "new" nodes. The process stops, in less than $|V(G)|$ iterations, if no improvement involving at most $t+1$ vertices has been found. The following general result can be used to argue about the approximation ratio of BETTER_IS.

Theorem 11. [15] *Let $E_1, \dots, E_m$ be subsets of a set T of n elements. Assume that:*

1. *Each element of T is contained in at most $k \geq 3$ of the sets $E_1, \dots, E_m$;*
2. *For any $p \leq t$, any p sets among $E_1, \dots, E_m$ cover at least p elements of T.*

Then:

$$\frac{m}{n} \leq \frac{k(k-1)^r - k}{2(k-1)^r - k} \quad \text{if } t = 2r-1, \qquad \frac{m}{n} \leq \frac{k(k-1)^r - 2}{2(k-1)^r - 2} \quad \text{if } t = 2r.$$

In our application V plays the role of T and $k = d$. Next, if $U = \{u_1, \dots, u_{\alpha(G)}\}$ is a maximum independent set in G, we define $E_i = \{v \in S : \{v, u_i\} \in E(G)\}$ for each $i \in \{1, \dots, \alpha(G)\}$. Finally, note that ondition 2in Theorem 11 is satisfied by construction when the algorithm terminates. This proves the following Theorem Note that the result is valid for any d-perfectly groupable graph, but the statement below suffices for our purposes.

Theorem 12. *Let t be a positive integer, and $d \geq 3$ be an integer. Let $G \in$ GRIDLINE^d and let S be the independent set returned by* BETTER_IS(G, t). *Then*

$$\frac{\alpha(G)}{|S|} \leq \frac{d - c/(d-1)^{\lfloor t/2 \rfloor}}{2 - c/(d-1)^{\lfloor t/2 \rfloor}}$$

where $c = d - (d-2)(1 - (t \mod 2))$.

Robustness. Note that the result works for any d-perfectly groupable graph (as per Definition 2) and it does not use the LoS network geometry.

6 Conclusion

This paper studies the worst-case complexity of the MIS problem in LoS networks with parameters d, n and ω. We show that the problem is NP-hard in many cases, and even difficult to approximate when the range parameter ω is maximized. We also describe some positive algorithmic results. Perhaps the most interesting questions left open by our work concerns the complexity of MIS on LoS networks with $\omega = n-1$. We know that MIS can be solved in polynomial time for gridline graphs (i.e. when $\omega = n$) in two dimensions, but MIS could be NP-hard for $d = 2$ when $\omega = n-1$. For $d > 2$, we know that MIS is APX-hard for gridline graphs and yet it could admit a PTAS when $\omega = n-1$.

Acknowledgement. The authors are endebted to the anonymous reviewers for reading the paper and providing numerous comments that have helped improving greately the quality of the presentation.

References

[1] K. Akcoglu, J. Aspnes, B. DasGupta, and M.-Y. Kaos. Opportunity cost algorithms for combinatorial auctions. *Computational Methods in Decision-Making, Economics and Finance*, 74:455–479, 2002.

[2] B. Bollobás, S. Janson, and O. Riordan. Line-of-sight percolation. *Combinatorics, Probability and Computing*, 18(1-2):83–106, 2009.

[3] M. Cesati and L. Trevisan. On the efficiency of polynomial time approximation schemes. *Information Processing Letters*, 64(4):165–171, 1997.

[4] S. N. Chiu, D. Stoyan, W. S. Kendall, and J. Mecke. *Stochastic geometry and its applications*. John Wiley & Sons, 2013.

[5] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete mathematics*, 86(1-3):165–177, 1990.

[6] A. Czumaj and X. Wang. Communication problems in random line-of-sight adhoc radio networks. In *International Symposium on Stochastic Algorithms*, pages 70–81. Springer, 2007.

[7] L. Devroye and L. Farczadi Connectivity for line-of-sight networks in higher dimensions. *Discrete Mathematics & Theoretical Computer Science*, 15(2):71–86, 2013.

[8] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4(5):235–282, 1994.

[9] R. Diestel. *Graph theory*. Springer, 2000.

[10] D.-Z. Du, K.-I Ko, and X. Hu *Design and Analysis of Approximation Algorithms* Springer, 2012.

[11] A. Frieze, J. Kleinberg, R. Ravi, and W. Debany. Line-of-sight networks. *Combinatorics, Probability and Computing*, 18(1-2):145–163, 2009.

[12] M. R. Garey and D. S. Johnson. *Computers and intractability: A Guide to the Theory of NP-completeness*, 1979.

[13] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[14] D. Hochbaum (ed.) *Approximation Algorithms for NP-Hard Problems* PWS, 1997

[15] C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an sdr, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics*, 2(1):68–72, 1989.

[16] F. Kammer and T. Tholey. Approximation algorithms for intersection graphs. *Algorithmica*, 68:312–336, 2014.

[17] V. Kann. Maximum Bounded 3-Dimensional Matching is MAX SNP-complete *Information Processing Letters*, 37:27–35, 1991.

[18] M. V. Marathe, H. Breu, H. B. Hunt, S. S. Ravi, and D. J. Rosenkrantz. Simple Heuristics for Unit Disk Graphs *Networks*, 25:59–68, 1995.

[19] M. Milanič, P. Muršič, and M. Mydlarz. Induced embeddings into hamming graphs. In K. G. Larsen, H. L. Bodlaender, and J.-F. Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017)*, Leibniz International Proceedings in Informatics, pages 28:1–28:15. Dagstuhl Publishing, Germany, 2017.

[20] B. Mohar and C. Thomassen. *Graphs on surfaces*, volume 2. John Hopkins University Press, 2001.

[21] T. Nieberg, J. Hurink, and W. Kern. A robust ptas for maximum weight independent sets in unit disk graphs. In J. Hromkoviš, M. Nagl, and B. Westfechtel, editors, *Graph Theoretic Concepts in Computer Science; 30th International Workshop, WG 2004*, volume 3353 of *Lecture Notes in Computer Science*, pages 214–221. Springer-Verlag, 2004.

[22] C. H. Papadimitriou and M. Yannakakis Optimization, Approximation and Complexity Classes *Journal of Computer and System Sciences*, 43:425–440, 1991.

[23] D. Peterson. Gridline graphs: a review in two dimensions and an extension to higher dimensions. *Discrete applied mathematics*, 126(2):223–239, 2003.

[24] V. Raghavan and J. Spinrad. Robust algorithms for restricted domains. In Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms, pages 460-467. Society for Industrial and Applied Mathematics, 2001.

[25] P. Sangha, P. W. H. Wong, and M. Zito Dynamic Programming Optimization in Line of Sight Networks. ArXiV preprint `arXiv:1806.01581`, submitted for journal publication (June 2018).

[26] L. G. Valiant. Universality considerations in vlsi circuits. *IEEE Transactions on Computers*, 100(2):135–140, 1981.

[27] Y. Ye and A. Borodin. Elimation graphs. In S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. Nikoletseas, and W. Thomas, editors, *Automata, Languages and Programming: 36th International Colloquium, ICALP 2009*, volume 5555 of *Lecture Notes in Computer Science*, pages 774–785. Springer Verlag, 2009.

[28] D. R. Wood. On higher-dimensional orthogonal graph drawing. In *Proc. Computing: the Australasian Theory Symp.(CATS'97)*, volume 19, pages 3–8, 1996.